UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Development and assessment of a one-dimensional CFD solver for boiling flows in bubbly regimes

June 23, 2020

Author: Consuelo Gómez-Zarzuela Quel

Advisor: D. Rafael Miró Herrero

RESUMEN

El presente trabajo de doctorado tiene por objetivo el desarrollo de un *solver* unidimensional capaz de resolver sistemas de fluidos monofásicos y bifásicos. La novedad de este proyecto reside en el uso de una plataforma *CFD* de código abierto, llamada OpenFOAM®, como marco para el desarrollo de la nueva herramienta.

Para el desarrollo del nuevo *solver*, se han analizado las ecuaciones de conservación basadas en Navier-Stokes (tridimensionales) y se han reducido a una dimensión. Para la parte bifásica del *solver*, se utiliza el método *Two Fluid Model*. Además, se han incluido todos los modelos empíricos necesarios como ecuaciones de cierre del sistema.

El *solver* final incluye una serie de requerimientos que refuerzan sus capacidades. Entre ellas, destacan, por un lado, el uso de una segunda malla que represente el sólido y tenga en cuenta el calor transmitido al fluido por conducción a través de un sólido. Por otro lado, se ha tenido en cuenta la posible transferencia de masa entre fases en fluidos bifásicos. Igualmente, se ha implementado un modelo de ebullición subenfriada que tiene en cuenta la posible generación de vapor cerca de la pared mientras el centro del fluido se mantiene por debajo de la temperatura de saturación.

Finalmente, este trabajo presenta la verificación y validación del *solver*. La verificación se ha realizado principalmente con el código de sistema *TRACE*. Para la validación, se cuenta con los resultados de dos casos experimentales que permiten demostrar la validez física de la nueva aplicación desarrollada.

La implementación del nuevo *solver* en esta plataforma abierta permite un futuro acoplamiento mucho más directo entre mallas unidimensionales y tridimensionales, obteniendo una mayor optimización del cálculo.

ABSTRACT

The present PhD thesis aims at the development of a one-dimensional solver capable of solving single- and two-phase flow fluid systems. The novelty of this project lies in the use of an open source CFD platform, called OpenFOAM®, as a development framework for the new tool.

For the new solver development, the conservation equations based on Navier-Stokes (three-dimensional system) have been analyzed and reduced to one dimension. For the two-phase simulations, the *Two Fluid Model* method was used as base. In addition, a series of empirical models have been selected as closing equations of the system.

The final solver includes a series of requirements that reinforce their capabilities. Among them, the use of a second mesh that represents the solid and takes into account the heat transmitted to the fluid by conduction through a solid, stands out. On the other hand, the possible transfer of mass between phases in two-phase fluids has been taken into account. Similarly, a subcooled boiling model has been implemented which takes into account the possible generation of vapor near the wall while the bulk is kept below saturation temperature.

Finally, this paper presents the verification and validation of the solver. The verification has been carried out mainly with the system code *TRACE*, whose validation has been demonstrated in numerous works and its use is very extended in the scientific community. For the validation, we have the results of two experimental cases that allow us to demonstrate the physical validity of the new application developed.

The use of this platform allows for a much more direct coupling between one- and three-dimensional domains, obtaining a better optimization of the calculation.

RESUM

El present treball de doctorat té per objectiu el desenvolupament d'un nou *solver* unidimensional capaç de solucionar sistemes amb fluids monofàsics i bifàsics. La novetat d'aquest projecte resideix en l'ús d'una plataforma *CFD* de codi obert, anomenada OpenFOAM® com a marc de desenvolupament de la nova eina.

Per al desenvolupament del nou *solver*, s'han analitzat les equacions de conservació basades en Navier-Stokes (tridimensionals) i s'han reduït a una dimensió. Per a la part bifàsica del *solver* s'utilitza el mètode *Two Fluid Model*. A més, s'han inclòs tots els models empírics necessaris com a equacions de tancament del sistema.

El *solver* final inclou una sèrie de requeriments que reforcen les seues capacitats. Entre elles, destaquen, d'una banda, l'ús d'una segona malla que represente el sòlid i es tinga en compte la calor transmesa al fluid per conducció a través d'un sòlid. D'altra banda, s'ha tingut en compte la possible transferència de massa entre fases en fluids bifàsics. Igualment, s'ha implementat un model d'ebullició subrefredada que té en compte la possible generació de vapor prop de la paret mentre el centre del fluid es manté per davall de la temperatura de saturació.

Finalment, aquest treball presenta la verificació i validació del *solver*. La verificació s'ha realitzat principalment amb el codi de sistema *TRACE*, la validació del qual s'ha demostrat en nombrosos treballs i el seu ús està molt estés en la comunitat científica. Per a la validació, es compta amb els resultats de dos casos experimentals que permeten demostrar la validesa física de la nova aplicació desenvolupada.

L'ús d'esta plataforma permiteix un futur acoblament més directe, entre elements unidimensionals i tridimensionals, obtenint una major optimització del càlcul.

ACKNOWLEDGEMENTS

Now that I have reached the end of an stage in my life, I cannot help looking back and realizing that the list of people who helped me at some point during this path is far larger than I initially thought. Not everybody was necessary involved in the work, but they contributed in the achievement of this thesis.

First of all, I would like to thank my director, Prof. Rafael Miró for giving me this opportunity and for his suggestions and help during the way. I am also very grateful to Dr. Sergio Chiva and Dr. Carlos Peña-Monferrer for their guidance during last stage of this thesis. Without our invaluable meetings this thesis would not be complete.

This adventure also gave me the opportunity of learning from the best research groups abroad, and I would like to thank my advisors, Prof. Analissa Manera and Prof. Maria Avramova for their warm welcome and their always productive meetings and suggestions. I cannot forget the inestimable assistance of Dr. Agustín Abarca, specially during the two months that I stayed in Raleigh.

My colleagues in the Department of Chemical and Nuclear Engineering at UPV, who with I shared pains and joys of the PhD have been a fundamental part these years. Special thanks to my already friends Antonella, María, Amanda and Sergio, but I cannot forget of Patricio, Sete, Javier or Aina. These are only a part of the amazing PhD student group that this department has.

And last, but no least, to my family, who always supported me and believed in me even when I did not. And to you, Rubén, who suffered when I suffered and laughed when I laughed. You are the only reason this thesis is finished.

CONTENTS

LIST OF SYMBOLS

Acronyms

1D	One-dimensional
3D	Three-dimensional
BC	Boundary Condition
BSD	Bubble Size Distribution
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy number also known as Courant number
CHT	Conjugate Heat Transfer
CT	Churn-Turbulent flow regime
CV	Control Volume
DCA	Drag Coefficient Approach model
DCA*	Drag Coefficient Approach considering Sauter diameter
DCA**	Drag Coefficient Approach with specific drag closure and bubble size distribution
FEM	Finite Element Method

FVM	Finite Volume Method
KI	Kataoka-Ishii model (referring to bubbly-slug flow regime)
LDA	Laser Doppler Anemometry
MULES	Multi-dimensional Universal Limiter with Explicit Solution
NEA	Nuclear Energy Agency
NRC	Nuclear Regulatory Commission
NRS	Nuclear Reactor Systems
PIMPLE	Combination of SIMPLE and PISO
PISO	Pressure-Implicit with Splitting of Operators
PVM	Parallel Virtual Machine
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
TFM	Two-Fluid Model

Other Symbols

∇s	Gradient of the scalar field s
$\nabla \cdot v$	Divergence of the vector field v

Greek Symbols

α_k	Void fraction of phase k
β	Fluid thermal expansion
Γ_k	Mass gained by phase k
Γ_{ki}	Mass transferred from phase i to phase k
Γ_{sub}	Mass transfer term due to subcooled boiling
$\gamma_{k,i}$	Interfacial heat transfer coefficient
γ_{PB}	Pool Boiling heat transfer coefficient
γ_{sub}	Wall heat transfer coefficient due to subcooled boiling
γ_w	Wall heat transfer coefficient

$\Delta T_{ONB,sat}$	Wall superheated temperature necessary for the onset of nucleate boiling
ΔT_{sub}	Subcooling temperature
ϵ	Wall roughness
κ	Thermal conductivity
κ^{eff}	Effective conductivity
λ	Solid conductivity
μ	Fluid dynamic viscosity
μ_m	Mixture viscosity
ν	Kinematic viscosity
ρ	Density
σ	Superficial tension
τ	Shear stress term
τ_k	Phase k shear stress term
τ_k^t	Phase k turbulent shear stress term
ϕ	Contact angle

Symbols

a_w	Wall heat transfer area per unit of volume
a_i	Interfacial area
$angF$	Correction factor dependent on the material
C_D	Drag momentum term
C_d	Drag coefficient
C_{NB}	Correction term for liquid wall friction coefficient in two-phase flows
C_p	Heat capacity
$C0$	Distribution parameter of drift flux model
D_h	Hydraulic diameter

D_h^*	Dimensionless hydraulic diameter
d_{32}	Sauter mean diameter
d_b	Mean bubble diameter
Eo	Eotvös number
f	Filonenko friction factor
F_w	Wall friction term
f_w	friction factor coefficient
$f_{wl,\Phi}$	Wall friction coefficient for two-phase flow
f_{sub}	Fraction of the heat transfer from the wall to the liquid that generates evaporation near the wall when the bulk liquid temperature is subcooled
G	Mass flux
Gr	Grashof number
g	Gravity
h	Enthalpy
h_k	Phase k enthalpy
$h_{k,sat}$	Phase k enthalpy at saturation conditions
h_{ld}	Enthalpy at which bubbly detachment occurs
$h_{l,sat}$	Liquid enthalpy at saturation conditions
$h_{v,sat}$	Vapor enthalpy at saturation conditions
j	Volumetric flux of a mixture
K_w	Wall drag coefficient
k	Phase of the fluid (continuous or dispersed)
M_k	Momentum transfer term of phase k
Nu	Nusselt number
$N_{\mu l}$	Liquid viscosity number

Pe	Peclet number
Pr	Prandtl number
P_r	Reduced pressure
p	Pressure
q'''	Volumetric heat source
q''_{BI}	Boiling Initiation heat flux
q''_{cond}	radial conduction term
q''_{conv}	Convective heat flux
q''_k	Thermal heat flux in phase k
q'''_k	Volumetric heat source in phase k
$q_{k,i}$	Interfacial heat transfer in phase k
q''_{NB}	Nucleate boiling heat flux
q''_{ONB}	Heat flux at the Onset Nucleate Boiling
q''_{PB}	Pool Boiling heat flux
Re	Reynolds number
r_k	bubble radius
St	Stanton number
T	Temperature
T_{ONB}	Temperature at the Onset Nucleate Boiling
T_{sat}	Saturation temperature
U	Velocity
U_k	Phase k velocity
U_r	Relative velocity
V	Fluid volume
v_{gj}	Local drift velocity term
$\bar{v_{gj}}^+$	Weighted drift velocity
We	Webber number

CHAPTER 1

INTRODUCTION

In an growing energy demanding era, which is facing the need of reducing greenhouse gas emissions, nuclear power stands as a support for renewable energy. Its constant generation regardless the climatology conditions along with its minimum CO_2 emissions, makes it ideal to work as a base load energy. In particular in Spain, where the National Plan of Energy and Climate (PNIEC, IDAE, 2019) aims to reduce CO_2 emissions up to a 40% from levels of 1990 by 2030, nuclear energy should be considered as a strong candidate to participate in the electricity mix of the new energy generation plan.

In order to make it real and keep nuclear power as base load energy, it is necessary to hold the safety of these facilities as the primary milestone. This fact has been highlighted along time by different international institutions as the International Atomic Energy Agency (IAEA), the Nuclear Energy Agency (NEA) or the United States Nuclear Regulatory Commission (USNRC). Especially since Fukushima, significant improvements have been undertaken in every nuclear power plant to reinforce safeguard measures. In Spain, this is reflected in Almaraz-I NPP, which was considered the safest nuclear power plant in Europe in 2018 according to ranking elaborated by WANO (World Association of Nuclear Operators, www.wano.info). This achievement was the result of high investment in this NPP to improve safety.

Safety enhancing in nuclear power plants requires of a very deep knowledge of the physical phenomena that happens inside the nuclear reactor, so that the system works always within the safety limits. In order to get this knowledge, significant amounts of resources have been devoted for the better understanding of nuclear phenomena, the heat generated and its transfer to water for the past 50 years. Specially attention has been drawn to the instabilities in Boiling Water Reactors. The feedback between fission and bubbles can get delayed very fast if determined conditions are met, due to density-wave oscillations and turning the reactor unstable. Hence, two-phase flow has always been a matter of deep research, because it is imperative to understand the generation of bubbles and the forces that affect them.

One-dimensional thermal-hydraulic system codes coupled to kinetic codes have been an advantageous tool to understand the overall phenomena and feedback between kinetics and thermal-hydraulics. However, due to the inherent behavior of the two-phase flow, there is a lack of information when using these tools. Since computational fluid dynamic (CFD) codes fill the gap of information where system codes do not reach, they have become a fundamental tool to carry out detailed analysis in the fluid. There are a wide variety of systems or events where two-phase flow is of primary importance, such as reactor heat removal system (RHRS) or Loss-Of-Coolant-Accident (LOCA's). Furthermore, not only nuclear industry is focused on two-phase flow, but also chemical industry in different processes such as wastewater treatment plants or oil and petroleum industries due to their production system are very interested in accurate results for this field.

This work deals with the development, verification and validation of a new one-dimensional two-phase flow solver in a CFD package called $OpenFOAM$. One-dimensional (1D) simulations allow us to have a general perspective about what is going on in the system by getting averaged values of the main properties, such as pressure, velocity or enthalpy in a short period of time. On the other hand, 3D CFD simulations give a very detailed information about how the fluid behaves in the system and how it interacts between phases or with different components (in pumps, valves, spacer grids...), but they request amounts of computational time.

The current trend is coupling both 1D and 3D simulations, using the main advantages of both strategies. The philosophy on this strategy is to perform a 1D calculation of the general system, leaving the 3D computing for particular components or areas of the system where accurate results are needed. Following this strategy, the information obtained is more complete, while the computational time is still reasonable. Generally, this coupling is undertaken between a

1D system code and a 3D CFD code, where a wide range of coupling possibilities in the type of codes, code feedback or coupling approach. Examples of different types of coupling will be given in chapter 2. This strategy is currently used for single-phase flow simulations, where the validation of the codes is extended, but its application to two-phase flows is very limited yet, since the level of accuracy of these codes is on continuous improvement. The margin of advancement for these kind of simulations is still high.

1.1 Research objectives

The aim of this thesis is the development, verification and validation of a new one-dimensional solver for both single-phase and two-phase flows within a CFD platform. The final idea is to apply CFD advantages to 1D simulations, as well as to facilitate a future coupling between this 1D solver and the 3D platform. Different system codes and even CFD codes were used in order to verify the new solver. In addition, various benchmarks and a set of experimental results were used to validate the capability of this solver for one-phase and two-phase flow simulations. In this way, a deep analysis can be undertaken about the appropriate numerical methods that are applied for different conditions as well as the adequacy of the empirical correlations applied to represent phenomena which are not governed by any mechanistic model yet.

To achieve these objectives, the development was performed in the CFD code $OpenFOAM$. This software is meant as an open-sourced set of libraries with capability to resolve any fluid dynamic problem. $OpenFOAM$ was chosen basically for two reasons: First, for the fact that it is free and open source, which allows the modification of every line so that the user can adapt the theoretical background of the problem that is going to be solved. Then, because it is modular, and its structure may include future contributions of different models or equations. Both statements here presented allows that the implementations carried out in this CFD can be sustained, optimized and improved by the scientific community. Focused on two-phase flows, the current empirical correlations implemented could lead to future mechanistic models as for example the interfacial forces or the subcooled boiling model.

Furthermore, this work is intended as a future 1D/3D coupling methodology where no external coupling tools are needed and all the capabilities already compiled in the CFD can be used. This new solver must include the following features:

- One-dimensional solver
- Able two work with single-phase and two-phase flow
- Limited to bubbly/slug flow regimes
- Mass transfer between phases capability
- Subcooled boiling simulations
- Conjugate heat transfer with solid

To fairly compare the simulation results between different codes, it should be required a relevant knowledge on the numerical methods applied and their similarities and differences, and the empirical correlations implemented and again which conditions have been validated. Consequently, this work not only concerns the development and validation, but also an extensive study about the discretization technique used and the analysis of empirical correlation validations.

1.2 Thesis outline

This work is composed by the following chapters. First chapter after this introduction, Chapter 2, presents the evolution of the computational fluid dynamics tools, why they appeared, how they evolved, their strengths and weaknesses and which are the current trends in their use. CFD are the following step in computational thermal-hydraulic simulations. They started working with single-phase flows, where they have currently become a basic tool for any kind of situation, regardless the type of fluid or regime. Nowadays CFD are also used for two-phase flow simulations, but in this field there is still margin to improve. Special mention is given for the 1D/3D coupling methodology, a solution that allows the use of CFD potential with a good balance in computational time. This chapter also relies in the necessity of collecting experimental data in order to validate the different programs, otherwise there is no way of improving applications or even the understanding of the phenomena. Therefore, different experimental installations where two-phase flow tests are performed are presented in the chapter.

Chapter 3 deepen in the development of the mass, momentum and energy conservation equations and the required rearrangements to avoid instabilities and guarantee conservation during the simulations. System codes work generally with a semi-implicit method that avoid divergences and numerical instabilities by averaging time, space or void fraction. The numerical methods in CFD codes varies so that possible numerical instabilities are avoided. The new solver de-

parts from a built-in CFD application were the system of three-dimensional equations is replaced by a set of one-dimensional equations, in both single- and two-phase flow cases. Additionally, one-dimensional closure models are also implemented, in order to solve the system properly. The developed solver considers also the heat transfer through a solid, so then, the formulation for the conduction equation through a solid is also implemented. This chapter presents the conservation equations for single-phase and two-phase flows. In the former case, the study was focused on wall friction and wall heat transfer coefficient models. This closure equations are also presented in order to get a primary idea of the solver. Closure equations for two-phase flow closes the chapter. This second part of the chapter introduces the different one-dimensional two-phase flow closure models that have been incorporated to the code. In both cases the model departs from a mechanistic model to represent the phenomena, but these models include an empirical constant that needs to be defined empirically. A study of different correlations is performed. Through the chapter, an analysis is presented on the modeling of the diverse interfacial forces. Since this work is focused on one-dimensional bubbly and slug flows, turbulence effects can be simplified, considering these effects within the wall friction model. Lift and wall turbulent interfacial forces are also neglected, leaving an important role to the drag force. This force along with the bubble diameter calculation is a decisive influence in the behavior of the bubbly flow simulated. Therefore an analysis of several empirical correlations of these models and their validation for different fluid conditions is carried out.

Chapter 4 presents a brief introduction about OpenFOAM. In this chapter, the main features of this software are described. One of the best-known properties is the similarity of the way of writing equations with reality. This means that the symbols used in the program represent same mathematical operation as in books. CFD codes commonly work with Finite Element Method (FEM) or Finite Volume Method (FVM) for three dimensions. OpenFOAM approach is based on the latter. The differences between techniques are presented, as well as the simplification to reach the one-dimensional equations. Furthermore, a brief introduction about the general structure of the code as well as the concept of *class* and *object*, which are basis for programming in this code, is done. In addition to this, the chapter pass through the different modification which have been undertaken and the external tools that have been required to become the solver in a powerful tool.

In relation to the former, chapter 5 describes the general structure of the final solver. The application was developed in two stages: first, a simple trial for one-dimensional single-phase flow simulations, which helped to understand the

platform, was performed. Then, the main effort was put to generate a two-phase flow solver, which were able to simulate also single-phase and boiling phenomena. In this chapter, one can see the final result after including all the modifications explained in chapter 4.

The testcase and different simulations performed so far are shown in chapter 6. Results can be divided into two parts. First, simulations for single-phase flows are presented. In this case, a verification was performed between the new one-dimensional solver developed in OpenFOAM and the well-known system code TRACE. The comparison showed a very good agreement between codes. The verification of this solver allowed having the base for the two-phase flow, which was built over this one, so that the final implementation could work for single-phase and two-phase flows. Next, an adiabatic two-phase flow is presented where the closure models for the interfacial forces where validated against an experimental set of results and the developed solver. Two-phase flow interfacial closure models were also verified against TRACE, completing the recommended steps for a proper CFD simulation. Lastly, the final solver including boiling phenomena and heat transfer through a solid is added to all of the above. Verification and validation was performed against TRACE system codes and a set of experimental results found in the literature.

Chapter 7 presents some conclusions, recommendations and a description of future steps are drawn. These are the final remarks obtained through the work of this thesis and future assignments that could be done to continue this work.

CHAPTER 2

STATE OF THE ART

This chapter presents the evolution of CFD codes and the different trends for optimizing their capabilities up to nowadays. CFD codes are widely used in single-phase flow, but their use in two-phase flow is still under validation. In order to get more accurate solvers in this field, it is also necessary to collect experimental data with different focus which allow the validation of two-phase flows simulations.

2.1 Computational Fluid Dynamics

Traditionally, one-dimensional thermal-hydraulic system codes have been used to model nuclear reactor systems. These codes provide good information of the primary circuit in a low computational time and under a wide range of conditions, but they are not able to acquire local phenomena, specially when three dimensional effects play the main part. There are different components where phenomena in the three directions is relevant, such as inlet plena. It is also significant when singular accidents happen, like main steam line break, boron dilutions or temperature stratification. This lack of accuracy induced the need of three dimensional software capable of providing accurate results in particular areas, which led to the next generation of thermal hydraulic codes, known as Computational Fluid Dynamics codes (CFD codes).

The emergence of these codes was favored by the evolution of the computer technologies, on the one hand, and the trend of a financial optimization of the plants,

on the other. A tendency to uprate the power in operating reactors or new designs are different arguments that motivated the development of CFD's.

The first group of Computational Fluid Dynamics (CFD) codes dates from the 1970's, when different laboratories developed their own codes to represent elements which cannot be defined as one-dimensional (Mahaffy et al., 2007), such as upper and lower plenums and downcomers. In this group one can find the American code COMMIX (Rivard and Torrey, 1977) or its European version, called FLUTAN (Willerding and Baumann, 1996). Soon afterwards, the capability of these codes was accepted and extended to Nuclear Reactor Safety (NRS) analysis. Thanks to the evolution of computational technology, CFD's evolved to the calculation of complex three-dimensional systems.

The application of CFD codes provides new and real benefits specially in single-phase flow analysis, where the application of this model in many different fields (aerodynamics, chemical engineering, turbo-machinery...) allows a preliminary validation. Due to this wide range of applications and the scientific acceptance of these codes, different commercial codes (Ansys-CFX, FLUENT, Star-CD...) were developed. One of their main advantages is that the simulated geometries are of high complexity, providing great flexibility and new perspectives (FLOW3D).

Particularly in Nuclear Reactor Systems (NRS), the application of CFD is advantageous in several applications, e.g erosion, corrosion and deposition, pressurized thermal shock (PTS), induced breaks or thermal fatigue. In this context, Jeong and Han, 2008 analyzed the flow distribution in the downcomer and lower plenum of Korean nuclear power plants using the real geometry in the analysis. A few years later, Lee et al., 2014 modeled the internal structure of a reactor on the accuracy of prediction for the scaled-down APR+(Advanced Power Reactor Plus) flow distribution, also applying real geometry.

However, until recently, there were still drawbacks that avoided the extended use of these codes in cases of design or license tasks, for instance. Furthermore, the verification and validation of these codes is, actually, on going yet. There are only a few accurate numerical solutions to be used as a reference for verification besides a analytic solution, although the continuing research in this field allows the emergence of new models to simulate particular events. And regarding to the validation, a similar situation takes place, since there is not enough tests that meet the requirements needed and there is not an appropriate PIRT (Smith et al., 2015). The PIRT (*Phenomena Identification and Ranking Table*) is a methodology developed for a particular event in order to systematically identify main aspects when dealing with specific phenomena of a primary relevance so that further studies can use the obtained results. In order to collect and share

all international efforts in validation of CFD's in NRS, the NEA organized the CFD4NRS Workshop in 2006, which has been celebrated every two years since then up to 2016, when last workshop took place in Boston (http://cfdr.mit.edu/).

Along with the problems in verification and validation, the other main disadvantage in CFD simulation is the computational time. Even though the computational time has improved with the evolution of the computer technology, it is not competitive enough yet. In order to overcome this problem, the current strategy consists in the coupling of a 1-D system code with a 3-D CFD code. This coupling is characterized by the match of the boundary conditions at the interface of both domains. Figure 2.1 presents an example of coupling.

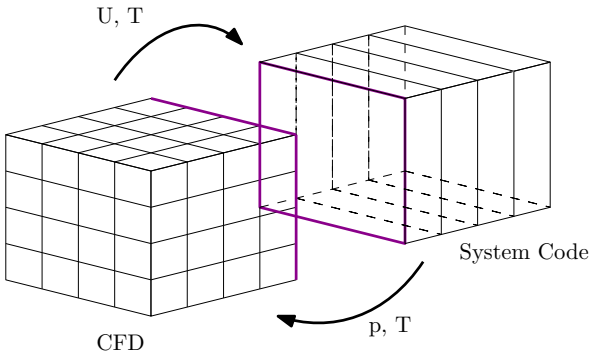


Figure 2.1: 3D-1D communication.

With this aim, the coupling of CFD codes with best-estimate thermal-hydraulic system codes is one of the strategies where more resources are being devoted. This is due to the fact that in this coupling one can encompass the power uprates necessity of the industry with the computational time reduction while the fidelity of the simulations is increased, fulfilling safety requirements given by the international regulations. Table 2.1 includes a summary of the couplings undertaken between the main system codes and CFD's.

The general procedure followed by both system codes and CFD codes to solve a thermal-hydraulic problem consists in discretizing in the spatial directions the linear algebraic equation system that represents the physical phenomena. The applied discretization will depend on the selected method and it will lead to a matrix system that represents the final system to be solved. In 1D - 3D couplings, one can use two different spatial decomposition method, according to the literature. These are known as the domain decomposition and the domain overlapping approaches.

Table 2.1: Examples of coupled codes found in the literature.

System Code	CFD	References
RELAP5	COBRA-TF	Jeong et al., 1997
RELAP5	CFX	Aumiller, Tomlinson, and Bauer, 2001
RELAP5	FLUENT	Li et al., 2014, Toti, Vierendeels, and Belloni, 2017
RELAP5	GOTHIC	Grgic et al., 2002
RELAP5	OpenFOAM	Godino and S. F. Corzo, 2018
TRACE	COBRA-TF	Ivanov and Avramova, 2007
TRACE	CFX	Bertolotto et al., 2009
TRACE	Star-CCM+	Grunloh and Manera, 2016
ATHLET	CFX	Kliem et al., 1999, Papukchiev et al., 2009
ATHLET	OpenFOAM	Huber, 2017
CATHARE	TRIO_U	Baviere et al., 2014

The domain decomposition approach splits the geometry to be modeled into two different domains. One domain is solved by the system code and another by the CFD. Both subsystems are interdependent and they exchange thermal hydraulic parameters through their boundary patches. Therefore, the calculated value in the boundary in CFD can be transferred to the first cell center in the system code and vice-versa. Figure 2.2 presents an scheme of domain decomposition. When information is tranferred from 3D domain to 1D, the values will be averaged to one only cell. If the values are transmitted towards 3D domain, they will be transformed according an appropriated statistical distribution and each cell will received the proper corresponding value.

The main advantage of using this approach is the simplicity and the robustness of the implementation. However, it might lead to numerical instabilities that end up in a divergence of the case.

Diverse efforts have been undertaken using this approach. Anderson, Hassan, and Schultz, 2008 applied the domain decomposition to the thermal mixing of the coolant at the outlet of the core into the outlet plenum in a Very High Temperature Reactor (VHTR). Bertolotto et al., 2009 applied it in order to analyze

single phase mixing phenomena. Toti, Vierendeels, and Belloni, 2016 used this methodology to demonstrate the validity of RELAP5-FLUENT coupling to reproduce a forced-to-natural convection transient event strongly affected by stratification.

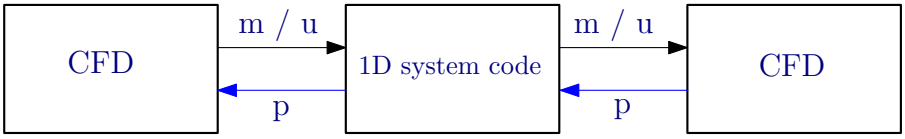


Figure 2.2: Domain decomposition scheme.

The domain overlapping methodology (figure 2.3) calculates the whole domain by means of the system code and only the regions which require special analysis, due to the strong 3D influence of the zone, are simulated in the CFD code. Hence, the codes are volumetrically coupled in the overlapping region. When undertaking the coupling, the system code region with strong 3D effects is internally corrected using information from the CFD, which is assumed to be more accurate that the system code. Due to these corrections, it is a primary requisite a proper validation of the CFD model.

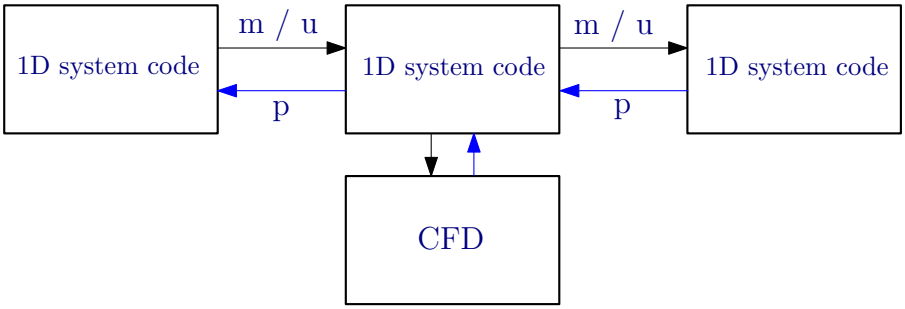


Figure 2.3: Domain overlapping scheme.

As a favorable opportunity of this methodology, the mass and momentum conservation equations are solved by the system code. This fact limits inconsistencies that can lead to numerical instabilities in the CFD.

The domain overlapping approach has been applied with different purposes. Fanning, Thomas, et al., 2010 utilized this strategy in order to identify critical safety aspects in advanced reactors. Kööp, 2018 applied this methodology to validate the STH-CFD coupling methodology for safety analysis in Generation IV sys-

tems. Using domain overlapping approach, they avoided numerical instabilities at low flows and possible discontinuities between codes.

Different studies have been carried out to compare both approaches. Papukchiev et al., 2015 used both methodologies for the simulation of the experimental facility TALL-3D. Different codes were used in each simulation, so there is not a clear conclusion about the comparison. Later, Grunloh and Manera, 2016 used also both methodologies to validate the coupling of TRACE system code with Star-CCM+, obtaining slightly better results with the domain overlapping approach. They also developed a domain overlapping method which corrects non-inertial pressure drops in the momentum equation of the 1D code on-the-go, giving better convergence that the decomposition domain approach.

Regardless the selected approach, this type of coupling requires two system of matrices and a platform to translate the information from a code to the other. This strategy can be known as *partitioned solution* (Angelucci et al., 2017). The issue of the platform is resolved in different ways: One of the codes can be adapted in order to be coupled (Weaver, Tomlinson, and Aumiller, 2002), external applications can be developed to transfer information between codes (Toti, Vierendeels, and Belloni, 2017) or an external PVM machine can be used as a link between codes (Bertolotto et al., 2009). The latter method sets one of the codes as the master, while the other remains as the slave. In all cases every program solves its system of matrices each time step passing the new solution to the other program. If each system is solved only once every time-step, the algorithm is called explicit implementation. One can use also an implicit algorithm, where every system is solved several time steps until the value at the interface reach an specific convergence. In both cases, the system memory has to be large enough to admit the matrix systems and the transfer between them. A compromise solution between implicit and explicit is the use of a semi-implicit algorithm, reducing slightly the computational cost.

There is a different alternative to deal with the complexity of the system, which consists in solving 1D and 3D domains in the same program. This is called a *monolithic approach*. In this approach, no platforms to link codes nor external applications are necessary. The information is transferred by the same code. Generally, codes are not prepared by default to work with 1D-3D couplings, so one of the conditions of this methodology is the access to the source. In addition to this, and as in previous coupling strategies, it is necessary to define properly the conversion of the variables from 1D to 3D (Cadinu, Kozlowski, and Dinh, 2007).

Therefore, if the code is the same for both domains, it has to be capable of solving 1D and 3D calculations. Some system codes can perform 3D simulations, such as COBRA-TF or TRACE for specific components. However, as mentioned before, the capabilities of the latter do not reach 3D effects as, for instance, boron dilution or turbulence. Hence, the philosophy in this work is the opposite. In this work, all the calculations take place in a CFD. Nowadays, CFD codes are capable of performing 1D or 2D (axisymmetric) simulations. It remains to adapt the code so that it can simulate 1D and 3D regions in the same case.

This work, focused in the latter strategy, has the objective of developing a 1-D module within a 3-D framework, simplifying the transfer of information between both systems.

2.1.1 Two-phase flow in CFD's

Hitherto, the presented review applies mainly to three-dimensional single-phase flow simulations, which are broadly extended and validated. However, when it comes to the simulation of two-phase flow in CFD's, this field is still under validation, even though the improvements constantly achieved allow the user to apply them in diverse fields.

Along with the single-phase flow CFD simulations group, the OECD/NEA/CSNI created a working group only to focus on two-phase flow, so that the best practices guidelines were established (Bestion et al., 2014). This group listed the safety issues where two-phase CFD calculations can provide benefit, described the recommended methodology and provided the progress made so far. According to this guidelines, there are four main steps that have to be clear before carrying out a work about two-phase flow in CFD's: Identification of the flow processes involved, determination of a basic model, selection of the closure models and verification & validation.

Regarding to the flow processes, it was already mentioned in 2.1. There are special phenomena, such as Pressure Thermal Shock (PTS), Dry-out and DNB investigations or pool heat exchanger, where two-phase CFD simulations can bring more detailed information, giving new perspectives and the possibility of define more accurate margins in safety limits.

Two-phase simulations can be modeled following different approaches according with the degree of accuracy than one wants to achieve and the computational resources that are available. The different models range from the single homogeneous fluid that can be found in system codes to the two-phase flows modeled with interfacial models, which are nowadays impracticable yet. Peña-Monferrer,

2017 undertakes an study about the degree of accuracy that can be achieved with each model and the amount of information lost from one model to the next simpler.

Among these approaches, there is one that has been most extensively used than the rest. This is the Two-fluid Model (TFM, Drew and Segel, 1971). This model is based on an Eulerian-Eulerian approach, which consists in the average of the equations of each phase. The TFM was well-established in (Ishii and Mishima, 1984) and it was formulated by using each phase separately. The fields in each phase are time and volume averaged and it is used in both system codes (1D-TFM) and CFD codes (3D-TFM), according to the closure laws that are used (Ishii and Hibiki, 2011).

Interfacial transfer modeling is a primary matter regardless the selected basic model. Most of the interfacial model are empirical equations based on different experiments that allow their justification. This has been the way so far to explain the different interfacial forces like lift force (Antal, Lahey Jr, and Flaherty, 1991, Hibiki and Ishii, 2007), wall heat transfer (Podowski et al., 1997) or break up and coalescence (Yao and Morel, 2004, Hibiki and Ishii, 2002). With the current technology for experiments and simulations, one could get a more mechanistic model to use in CFD codes, avoiding dependencies from numerical coefficients that exist in every empirical model. The need of more experiments and, consequently, more investment in this area, is a constraint factor in order to be less dependent of empirical correlations.

Finally, the recommended methodology ends up with the verification and validation of the model. Demonstrating the mathematical correctness of the model so that it can be applied for different situations as well as showing a physical behavior that corresponds to reality in different scenarios is fundamental to verify and validate the capabilities of a new methodology.

There is a wide range of investigations using TFM approach in CFD and most of these have been done focusing on specific phenomena, like bubble area calculation or subcooled boiling. Kurul presented a preliminary study for void fraction prediction using TFM in CFD and including subcooled boiling. Anglart and Nylund applied similar methodology for rod bundles, getting a good agreement between experimental measurements and averaged axial void at each axial point. Tu and Yeoh focused their research on low-pressure subcooled boiling flows and pointing out that validated high-pressure models were inefficient in low-pressure cases. They also incorporated the population balance model to their analysis with and without heat transfer (Yeoh and Tu, 2006), finding good agreement in void fraction distribution and liquid velocity, but failing in the vapor velocity

prediction, probably due to the assumption that all bubbles types have same velocity. Krepper, Konar, and Egorov applied TFM including subcooled boiling to investigate the capability of CFD to contribute to the design of a fuel assembly. They concluded that calculating the temperature of the rod surface one can identify the region where the departure from nucleate boiling could lead to a potential damage in the surface of the pin.

From 2010, the application of TFM in CFD including subcooled boiling starts to spread to different codes and aims. Michta, 2011, Michta et al., 2012, Ghione, 2011 and Fu and Anglart, 2017 applies the subcooled boiling and phase change models in the open-source CFD OpenFOAM in adiabatic cases. Drzewiecki et al. carried out a sensitivity study of the parameters of subcooled boiling and two-phase flow models, and they demonstrated that the bubble diameter is the most influent parameter to the void fraction distribution. Corzo et al. presented a one-dimensional TFM application implemented in Octave, which based the governing equations from a solver taken from OpenFOAM code. Rzehak proposed some guidelines to work with turbulent bubbly flows. He also presented a comparison between the extended licensed code Ansys-CFX and the open-sourced OpenFOAM, but without heat transfer (Rzehak and Kriebitzsch, 2015). The results showed that significant differences are found in the near-wall turbulence, due to the implementation of the turbulent wall functions. The need of an improved multiphase turbulence model is highlighted in this paper, regardless the code.

Colombo and Fairweather undertook their research trying to avoid tunable models and replacing it by mechanistic models and simulating 20 different experiments in vertical pipes with subcooled boiling. They confirmed the capability of CFD to provide detail predictions of boiling flows, but they also pointed out the lack of mechanistic models for different empirical closure correlations that weaken the accuracy of the predictions. On the other hand, Rollins developed a TFM CFD solver with phase change that included a wide range of different closure correlations for the diverse interfacial forces and it was verified and validated for different experiments, showing good agreement.

Therefore, TFM in CFD codes is a field which is growing in interest and needs, where there is still margin to improve and provide new possibilities that help to increase accuracy while keeping or reducing computational time.

2.1.2 Two-phase flow experimental facilities

Boiling flow is one of the most extended phenomenon in the nuclear field and it is fundamental in order to evaluate the safety in the nuclear reactor. Boiling flow affects the reactivity and therefore the output power of the reactor. Moreover, it affects the criticality of the equipment, being directly involved in the heat dissipation capability of the installation. In two-phase flows where the Critical Heat Flux (CHF) is exceeded, the flow turns into Departure from Nucleate Boiling (DNB) state, which leads to a fast increase of the wall temperature, that may alter the integrity of the materials of the rod and clad.

Thus, it is necessary to analyze the convective flow and its micro-scale circumstances. Different experiments has been performed along years with the technology of each time. One extended benchmark is the experiment carried out by Bartolomej in 1967 (Bartolomej G., 1967). This work aimed at the true vapor content study in a pipe of circular section under forced convection conditions. The facility consisted of an open loop with contact heaters and a vertical pipe of 2 meters height which simulated a fuel element by a internal source located in the middle of the tube. The heat was generated by an alternating current through the tube. In order to acquire information, the facility had in place different thermocouples to measure flow temperature at the inlet and outlet and the external wall surface. The vapor content was determined by irradiating the tube with a wide pencil of γ rays in two sections simultaneously. Datasets provided by this experiment contributed to the validation of empirical correlations that fitted different ranges of the two-phase flow regimes and have been implemented in one-dimensional thermal hydraulic codes. Due to this way of working, there is a wide variety of correlations in the field of two fluid modeling with a huge dependency on tuning factors, as recently addressed by Lucas et al. (2016).

Nowadays, experiments are still going on, but with improved techniques in order to get the results. One of the experiments used in this thesis for validation purposes was performed in a facility located at the Fluid Mechanics Laboratory of the Universitat Jaume I in Castelló. This facility consists of an upward flow experimental loop where four sensor conductivity probes are located at different heights. The technique used in order to extract the information was Laser Doppler Anemometry (LDA) (Peña-Monferrer, 2017). Thanks to the high speed cameras they can obtain liquid velocity, turbulent intensity in single-phase flows and, in addition to the previous, void fraction, bubble diameter, frequency and velocity and interfacial area concentration in two-phase flows. All these experiments are performed in adiabatic conditions. The current trend is providing enough data to develop mechanistic models which are independent of geometry or fluid properties.

However, the reproduction of the conditions in which a two-phase flow is involved within the reactor to validate radial profiles and different numerical contributions requires high investment. Therefore it has been necessary to look for alternatives. One alternative is the use of refrigerants due to the similarities of the properties with the water, but using lower pressure conditions. This option was chosen in the DEBORA test facility (Garnier, Manon, and Cubizolles, 2001). This facility consisted of a vertical pipe of 19.2 mm diameter and 3.5 m heigth through which dichlorodifluoromethane (R12) flowed upwards while it was heated. Optical probes were used to determine the gas volume fraction radial profile and gas velocity. Thermocouples obtained radial and axial liquid temperatures and wall temperature.

An important range of fluid properties and operation conditions could be covered if the number of experiments and their database values were great enough, but they were still conditioned to a particular geometry. As it was mentioned before, the use of CFD programs is required to get independence from geometry or fluid properties.

CHAPTER 3

GOVERNING EQUATIONS

This chapter presents an study of the governing equations used in the solver. The analysis departs from the three-dimensional Navier-Stokes equations for single-phase and two-phase flow and goes through its simplification to get the necessary one-dimensional form. Then, the solid conduction equation is introduced. The solving system is not complete without the closure equations, which allows to fulfill the necessary number of unknowns and equations. These closure models vary according to the system and the regime, and in this case the number of required models will be reduced due to the one-dimensional simplification.

3.1 Introduction

CFD codes are based on Navier-Stokes equations. These equations allows the depiction of the behavior of a fluid, by using them as the mathematical model to evaluate a change in the fluid properties due to a dynamic or thermal interaction. The Navier-Stokes equations consist of the conservation laws of mass, momentum and energy, although some authors specify the expression of Navier-Stokes equations merely for momentum equation.

There are mainly two mathematical models to investigate the fluid dynamics, and these are the Lagrangian and the Eulerian methods. The basis of each model depends on the position of a virtual *observer*. This can be located in a fixed position or he can be moved with the flow. Furthermore, in order to study de fluid, it is necessary to define a minimum particle which is big enough for the observer to

notice a change in its properties. If the observer selects some particles and follows them along time, the method of study is Lagrangian. Otherwise, if the observer is located at a fixed position and he observes all the particles that crosses his view, the method of study is Eulerian. The solvers used and developed in this work are based on the Eulerian method.

According to the flow conditions, the Navier-Stokes equations can be rearranged to provide accurate solution. The new arrangement can either increase or decrease the complexity of the problem. Compressibility, turbulence or multiphase flow are constraints that determine the type of problem to deal with.

Following sections presents the departing conservation equations and the assumptions taken in order to reduce the initial complexity. Figure 3.1 presents the main choices that should be taken to define a system that is going to be solved. Blue dashed path highlights the options selected in this work.

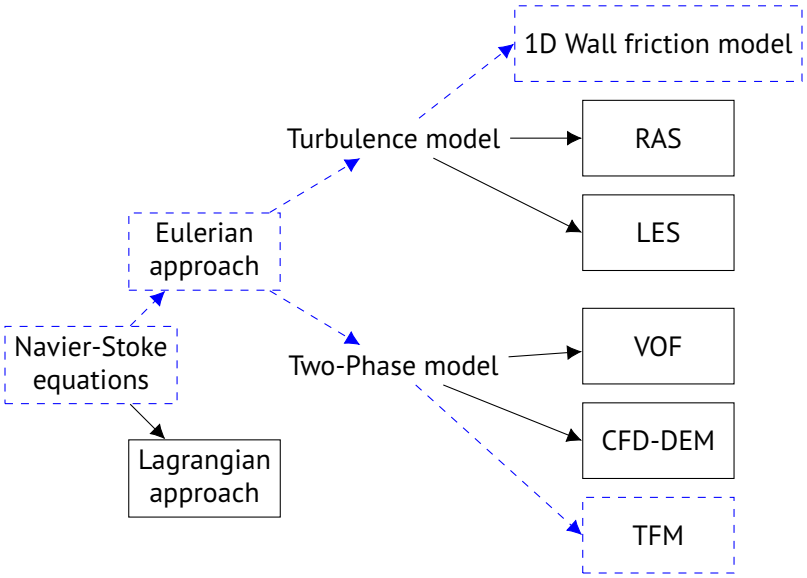


Figure 3.1: Selected models for the development of the one-dimensional solver

It is worthy to remind that the aim of this work is developing a one-dimensional solver, which works for single and two-phase flows. Summarizing very briefly the undertaken work, one could say that there were three main stages: First, a one-dimensional single-phase solver was developed departing from a CFD solver. In this step, the system of equation originally implemented in the CFD application was replaced by a one-dimensional set of equations. This step was carried

out in order to get familiarized with the language and structure of the CFD, since starting with one-dimensional single-phase flows allows easier and simpler simulations. Next, similar task was performed but dealing with two-phase flows: A CFD solver was chosen and its set of equations was replaced by the required one-dimensional set of equations. Additionally, different one-dimensional closure models were implemented, in this case they were added to the collection, without replacing any original model. Once this second stage was also overcome and the solver was able to perform two-phase flow simulations, it was adapted so that it could also simulate single-phase cases and the same solver involved any kind of flow.

The solver developed in this work was implemented within the OpenFOAM framework (Weller et al., 1998). OpenFOAM was meant as a set of libraries that allows to solve the fluid-dynamic equations. Section 4.1 presents a detailed introduction about this program. It is composed of a variety of solvers, where each one is meant to solve particular conditions. In this case, the selected solver for single-phase flows was buoyantPimpleFoam, which is a *solver for turbulent flows of compressible fluids in transient simulations* (OpenCFD-Ltd, 2018). Once the single-phase solver was transformed and adapted to satisfy requirements, the implementation was moved to the two-phase solver. The two-phase flow analysis is based on the solver called twoPhaseEulerFoam (OpenCFD-Ltd, 2018), which is a widely used solver for Eulerian simulations based on the TFM approach. Finally, the latter was adapted so that it was able to work with single-phase and two-phase flows.

The single-phase version of the new solver presented in this work must be able to work with both liquid and gas states, this is the reason because a compressible solver was taken as a reference to start the work, instead of an incompressible one.

In order to undertake this work, the general structure of buoyantPimpleFoam remained unchanged, but the equations were replaced by those used in the one-dimensional reference code, which in this case, the system code TRACE was selected. Therefore, the final solver keeps the structure of the original CFD solver, but the equation corresponds to a one-dimensional system, which is the main purpose of this work.

The built-in solver buoyantPimpleFoam is characterized by its ability to simulate in 3D flows that are:

- compressible

- turbulent
- able to consider convective heat
- in transient conditions

The substitution of the three dimensional Navier-Stokes equations by the TRACE one-dimensional system of equations implies the adaptation of the features listed above. Regarding to the compressibility, the new system of equations is used also for compressible fluids, so this feature is conserved. In turn, 3D turbulence is not considered in the new 1D solver implemented in this work. Instead, a wall friction model is considered in order to take into account the axial contribution of the diffusion term. The wall friction model included is also taken from TRACE system code, in order to ensure the correct performance of the new code when inter-comparison of results with the former is applied. The ability of heat transfer is also considered in the new solver by including the one-dimensional energy equation. Finally, the transient condition is remained, since the set of equations used as reference is time-dependent.

Up here, the new solver is summarized as a solver based on the built-in buoyantPimpleFoam but with one-dimensional conservation equations. Once this structure was assessed, new features were also implemented in the new 1D solver. In particular, the addition of a new domain to consider the solid wall and the integration of the corresponding wall heat transfer model were also carried out. Again, both solid conduction equation and wall heat transfer correlation were based on the reference system code equations in order to avoid discrepancies in the simulation comparisons performed afterwards. Section 3.2 presents the equations implemented in the solver and the main differences with respect the original set.

Similar methodology was followed to implement the 1D two-phase solver. In this case, the OpenFOAM solver used as a basis was twoPhaseEulerFoam as mentioned before. Again, the set of original equations of this solver was replaced by the 1D system of conservation equations found in TRACE. In this case, twoPhaseEulerFoam in fluids with the following features:

- with 2 compressible phases, where one is dispersed in the other
- turbulent flows
- able to transfer heat between phases
- in transient conditions

Again, in this case, when the substitution of the equations is applied, these properties are adapted. Regarding to the Eulerian model, this is conserved since the 1D system of equations is also based on the Eulerian approach. On the other hand, and similarly to the previous case, the 3D turbulence is replaced by a 1D wall friction model. And finally, both heat transfer between phases and transient conditions are remained since the 1D energy equation considers the interfacial heat transfer and all the implemented equations are time-dependent. In this case, the different closure correlations integrated in the new solver were selected due to its contribution in the reference system code used in this work. Following same methodology as in the single-phase stage, once the fluid equations were assessed, the solid domain was applied. Section 3.4 presents the new implemented Eulerian system of equations, as well as the different closure models included in the new 1D solver.

Once the modified 1D two-phase flow solver was evaluated and its capability for two-phase flow simulations tested, in order to merge both single and two-phase solvers, this last solver was adapted to work with single-phase simulations, assuming that a fluid is single-phase when the void fraction, which is the ratio of presence of dispersed phase respect to the continuous one, is so low that it can be neglected. The final solver is called `my1DTPFoam`.

The structure of this chapter is the following: the single-phase equations are described first, showing the original and final system and which terms are not longer necessary and which ones appear with the new system. Next, solid conduction equation is presented. Then, the two-phase flow equations will be developed in detail following similar procedure, and similarly with the closure models, which are not replacing any built-in model, but only added to the solver. For the sake of clarity, the single-phase closure models are presented immediately after the single-phase governing equations. Same pattern will be followed for two-phase flow and its closure models are last presented.

Every assumption taken is based on the consideration that water is the main fluid simulated.

Finally, it is worth to define a couple of operators which similarity could lead to confusion:

1. The **gradient** of a scalar field s can be defined as the vector field indicating that the value of s changes with position in both magnitude and direction (Moukalled, Mangani, Darwish, et al., 2016). This operator is represented by the symbol ∇ together with the scalar field to be calculated.

$$\nabla s = \frac{\partial s}{\partial x} i + \frac{\partial s}{\partial y} j + \frac{\partial s}{\partial z} k \tag{3.1}$$

2. The **divergence** of a vector field v can be defined as the scalar resulted of the dot product of its components u; v; and w in the x; y; and z direction. Therefore this operator is represented with the ∇ symbol plus a dot to act for the dot product.

$$\nabla \cdot v = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \tag{3.2}$$

3.2 Single-phase conservation equations

The mass conservation equation for compressible single-phase flows found in buoyantPimpleFoam reads as follows

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho U) = 0 \tag{3.3}$$

where ρ is the density of the fluid and *U* the velocity. Equation 3.3 represents the change of mass per unit volume. It states that there is a mass balance in the system, that is, a null mass flow difference throughout system between inlet- and outlet-section.

In a single-phase simulation with fluids that are assumed incompressible, such as water in liquid state, density could be neglected, because the changes in this property are going to be insignificant. However, since this solver is also meant to work with compressible fluids, like gas state fluids, where density is always relevant, equation 3.3 is kept compressible. Therefore, the new mass equation only presents a simplification, which consists in the reduction to one dimension. Considering x axis as the preferential direction, the final single-phase mass conservation equation is given by

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho U}{\partial x} = 0 \tag{3.4}$$

The original momentum equation found in the same solver makes statements about the changes in fluid properties and it is given by the following

$$\underbrace{\frac{\partial(\rho U)}{\partial t} + \nabla \cdot (\rho U U)}_{\text{I}} = \underbrace{-\nabla p}_{\text{II}} + \underbrace{\nabla \cdot \tau}_{\text{III}} + \underbrace{\rho g}_{\text{IV}} . \tag{3.5}$$

The left hand side of the equation represents the rate of change in the fluid per unit volume and time due to the different forces that are presented in the right hand side. Deepen describing every term, equation 3.5 represents the following phenomena:

- Term (I) represents the convective term. This term involves changes in the fluid in the three axis and in time.
- Term (II) represents the pressure forces that impact the fluid. The direction of this force is normal to the inlet and outlet surfaces of the volume and the common sign is negative because this force tends to be compressive.
- Term (III) stands for the surface forces that act to the fluid due to viscosity effects. This term is commonly referred as *shear stress term* or momentum diffusion term. Effects due to turbulence are included in this expression.
- Term (IV) is the gravity force, which is a body force. This expression will only influence the system in the vertical component of the system.

The three dimensional momentum equation calculation can be very challenging due to the complexity of the phenomena, i.e. if the turbulence is considered. Therefore, different simplifications are taken according to the conditions of the problem to be solved. In particular, if the system is reduced to one dimension, the shear stress term can be reduced and some of its terms neglected.

The momentum diffusion term τ, that accounts for turbulence, is defined as

$$\tau = -\rho\nu[\nabla U + \nabla^T U] + \frac{2}{3}\rho\nu(\nabla \cdot U)I \tag{3.6}$$

where ν stands for the kinematic viscosity and I corresponds to the identity matrix. This expression is developed according to the turbulence model selected for the simulation.

However, since the purpose of this work is to develop a one-dimensional solver, only the axial direction of this term is considered. Keeping in mind this assumption, only the component related to the axial diffusion of $\nabla \cdot (\tau)$ affects the calculation. Furthermore, the diffusive effect is small compared to the axial convec-

tive term of the momentum ($\nabla \cdot (UU)$). Therefore, the turbulence and Reynolds stresses are going to be replaced by a **wall friction correlation**, in order to keep the diffusion effect in the equation.

Keeping this consideration in mind, momentum equation can be rewritten as

$$\frac{\partial(\rho U)}{\partial t} + \frac{\partial(\rho UU)}{\partial x} = -\frac{\partial p}{\partial x} + F_w + \rho g, \tag{3.7}$$

where F_w denotes the wall friction model.

Finally, the original energy equation in buoyantPimpleFoam is based on the first law of the thermodynamics, which states that the sum of the work and heat added to the system will result in the increase of energy of the system. Written in terms of enthalpy, this equation is stated as follows

$$\underbrace{\frac{\partial(\rho h)}{\partial t} + \nabla \cdot (\rho U h)}_{\text{I}} + \underbrace{\frac{\partial(\rho K)}{\partial t} + \nabla \cdot (\rho U K)}_{\text{II}} = \underbrace{\frac{\partial p}{\partial t}}_{\text{III}} + \underbrace{\nabla \cdot (\alpha_{eff} \nabla h)}_{\text{IV}} + \underbrace{\rho U g}_{\text{V}} + \underbrace{S_k}_{\text{VI}} \tag{3.8}$$

Equation 3.8 is structured as momentum equation, that is, left hand side of the equation represents the rate of change inside the fluid element while the right hand side represents the net flux in the element and the rate of work applied to the fluid due to the surface and body forces. In particular, every term stands for the following phenomena:

- Term (I) denotes the time rate-of-change of energy of the fluid element. It involves changes with time and per unit volume.
- Term (II) involves the change in the kinetic energy in the fluid. The kinetic energy K is defined as $K = 0.5U^2$.
- Term (III) accounts for the rate of reversible work done by surface forces and by kinetic energy. The temporal term only participates when the equation is written in terms of enthalpy, since $h = e + \frac{p}{\rho}$.
- Term (IV) represents the rate of heat added to the fluid due to heat conduction. α_{eff} is the effective diffusivity, which is defined as the diffusivity

involved in the thermal convection plus the turbulent thermal diffusivity α^t, which is defined according to the turbulent model used.

- Term (V) is the rate of work done by body forces.
- Term (VI) represents the volumetric heat absorbed due to heat sources, i.e absorption or emission of radiation.

It is often useful to separate the mechanical and thermal effects in the total energy equation when solving problems. Hence, from the standard method of dotting the momentum equation by the velocity, the mechanical energy equation can be obtained (Todreas and Kazimi, 2011):

$$\frac{\partial(\rho K)}{\partial t} + \nabla \cdot (\rho U K) = -U \cdot \nabla p + U \cdot \nabla \cdot R^{eff} + \rho g U \tag{3.9}$$

Then, substracting equation 3.9 from equation 3.8, one can obtain the final enthalpy formulation. Furthermore, replacing turbulence by a one-dimensional heat transfer model and reducing the equation to a preferential axis, final energy equation states as

$$\frac{\partial(\rho h)}{\partial t} + \frac{\partial(\rho U h)}{\partial x} = \frac{\partial p}{\partial t} - p\frac{\partial U}{\partial x} - \kappa\frac{\partial T}{\partial x} + \rho U g + q^{'''}. \tag{3.10}$$

It should be mentioned that the heat flux due to thermal conduction has changed the sign due to the direct application of Fourier's law, which state the gradient of temperatures inverse to the direction of heat (Moukalled, Mangani, Darwish, et al., 2016).

3.3 Single-phase closure equations

In order to solve the system composed of equations 3.4, 3.7 and 3.10, a wall friction model that stands for the wall drag force and a wall heat transfer model that accounts for the dissipation of heat, are necessary.

System is not exclusively for liquid fluids, but it can also work with fluid in gas state, i.e single-phase vapor simulations, so closure models for this situation are also presented.

3.3.1 Wall friction model

The wall drag force, which is represented in momentum conservation equation (eq. 3.7) as F_w, allows for the shear between fluid and a solid structure. This superficial shear influences de pressure drop, which will influence fundamental variables, such as massflow. This force is given by the following

$$F_w = -K_w \left|U\right| U, \tag{3.11}$$

where K_w is the wall drag coefficient, which is expressed as

$$K_w = f_w \frac{2\rho}{D_h}. \tag{3.12}$$

In equation 3.12, D_h is the hydraulic diameter and f_w is the friction factor coefficient, which is given by an empirical model. There are different friction factor models, whose validity depends on fluid conditions and regime and it is given in terms of the Reynolds number.

In particular for single-phase flows, an extended review of the different wall friction coefficient models that exist in the literature is given in Yildirim, 2009 and in Fang, Xu, and Zhou, 2011.

In this work, two different models were implemented. One was selected in order to verify the capability of the code to calculate the pressure. This one is based on Churchill, 1977 and it is used for single-phase flow simulations or when only liquid is in contact to the wall. The second one, based on Wallis work, it is used in two-phase flow simulations. The use of this model is very extended, and it will be presented in section 3.10.

Churchill model

First model implemented in the code was proposed by Churchill (Churchill, 1977). This model was selected mainly because it is valid in all regimes (laminar, transition and turbulent). This correlation is dependent on the Reynolds number of the fluid and roughness of the pipe, and it can be written as

$$f_w = 2\left[\left(\frac{8}{Re}\right)^{12} + \frac{1}{(a+b)^{3/2}}\right]^{1/12}. \tag{3.13}$$

In equation 3.13, Re is the Reynolds number a and b are empirical constants that correspond to the following expressions,

$$a = \left\{ 2.457 ln \left[\frac{1}{\left(\frac{7}{Re}\right)^{0.9} + 0.27\left(\frac{\epsilon}{D_h}\right)} \right] \right\}^{16} \tag{3.14}$$

and

$$b = \left(\frac{3.753 \times 10^4}{Re} \right)^{16}. \tag{3.15}$$

In equation 3.14, ϵ represents the roughness of the solid structure.

It should be mentioned here that f_w is the Fanning friction factor, when usual models returns the Darcy friction factor. This is pointed out because Fanning friction factor is four times smaller than Darcy factor.

The predictions of this model for for laminar regime are in agreement with $f_w = 16/Re$, and for turbulent flow they show a maximum deviation of about 3.2% for 4000<Re< 10^8 and 10^{-8}< ϵ/D_h < 0.05 (NRC, 2013). This variation is accepted for the purpose of this work.

3.3.2 Wall heat transfer coefficient

Regarding to the wall heat transfer model, it is required for closure energy equation. This term represents the volumetric heat transfer rate from the wall to the fluid and it is defined as

$$q^{'''} = \gamma_w (T_w - T) A_w'''. \tag{3.16}$$

In equation 3.16, γ_w is the wall heat transfer coefficient, T_w accounts for the wall temperature, T is the fluid temperature, and A_w''' is the wall heat transfer area per unit of volume.

As happened in section 3.3.1, there are a wide range of wall heat transfer coefficient models, which depend on the regime of the fluid.

The wall heat transfer coefficient is generally expressed in terms of the Reynolds number and the fluid properties. A general description is given as

$$\gamma_w = Nu\frac{\kappa}{D_h} \tag{3.17}$$

Where Nu represents the Nusselt number, κ is the conductivity of the fluid and D_h stands for the hydraulic diameter. Since the hydraulic diameter is a constant value, and the conductivity is inherent to the fluid conditions, the equation can be reduced to the definition of the Nusselt number. In order to cover all regimes, different correlations for Nusselt number have been implemented for laminar, turbulent and natural convection. The final value for Nusselt will be the maximum of these three values.

$$Nu = max(Nu_{lam}, Nu_{turb}, Nu_{NC}). \tag{3.18}$$

Nusselt number for laminar convection is assumed to be the value used for fully developed flow, so it remains constant, $Nu_{lam} = 4.36$ (Holman, 2010).

For turbulent forced convection, the model proposed by Gnielinski (Gnielinski, 1976) has been implemented

$$Nu_{turb} = \frac{(f/2)(Re - 1000)Pr}{1 + 12.7(f/2)^{(1/2)}(Pr^{(2/3)} - 1)} \tag{3.19}$$

In equation 3.19, Pr accounts for the *Prandtl* number, which represents the ratio of influence of the momentum diffusivity against the thermal diffusivity. The term *f* is called the friction factor and it is calculated using the correlation of *Filonenko*

$$f = [1.58ln(Re) - 3.28]^{-2} \tag{3.20}$$

Gnielinski model was chosen because of its wide validity (for 2300< Re < 5x10^6) and because it gives a better approximation in the transition region that other models, such as the correlation proposed by Dittus and Boelter. However, it is limited to a Reynolds number greater than 1000, otherwise it yields to negative values of Nusselt, which is physically unrealistic (NRC, 2013).

Finally, the Nusselt number for natural convection is given by the maximum value between the laminar and turbulent natural convection (Holman, 2010),

$$Nu_{NC} = Max(Nu_{NC,lam}, Nu_{NC,turb}), \tag{3.21}$$

where

$$Nu_{NC,lam} = 0.59(GrPr)^{1/4} \tag{3.22}$$

and

$$Nu_{NC,turb} = 0.1(GrPr)^{1/3} \tag{3.23}$$

Using the maximum of both values, continuity is ensured. The component *Gr* is the Grashof number and it is based on the properties of the fluid and the characteristic length, which, since this model is assumed for cylindrical pipes, is the hydraulic diameter

$$Gr = \frac{g\beta\Delta T D_h^3}{(\mu/\rho)^2} \tag{3.24}$$

The Grashof number is a relation that allows quantifying the balance between the buoyancy and viscous forces, and in this case it shows how strong the natural convection is over the fluid. In equation 3.24 β is the thermal expansion of the fluid, which is the tendency of the fluid to change in response to a change in temperature; ΔT is the difference between wall and liquid temperature and μ and ρ represent the viscosity and density of the phase, respectively.

This solver is also meant to work with single-phase vapor flows. The flow will be assumed to be vapor when the void fraction is greater than 0.9999. In order to determine the heat transfer coefficient of a single-phase vapor flow, same model is used, but small modifications are applied.

In particular, for turbulent regimes, a corrector factor is applied in the Gnielinski formula. This is due to the fact that the fluid properties can vary significantly due to large temperature gradients close to the wall. This corrector ratio was proposed by Sleicher and Rouse and it is given by

$$\frac{Nu_{vp}}{Nu_{cp}} = \left(\frac{T_w}{T_v}\right)^n, \tag{3.25}$$

where the exponent *n* is

$$n = -\left[\log_{10}\left(\frac{T_w}{T_v}\right)\right]^{\left(\frac{1}{4}\right)} + 0.3. \tag{3.26}$$

In equation 3.25, Nu_{vp} corresponds to a Nusselt number of variables properties, whereas Nu_{cp} represents the value of Nusselt for constant properties. T_v is the vapor temperature.

The range of validity for this correction is $1 < \frac{T_w}{T_v} < 5$, so it is assumed that the fluid is being heated. Otherwise, n=-0.36, as suggested by Petukhov and also used by TRACE.

3.4 Two-phase flow conservation equations

This section presents the conservation equations for a two-phase flow. Following the path showed in figure 3.1, the methodology used in the two-phase system is the Two Fluid Model. This model is based on Eulerian approach, where both phases are treated as continua where each phase has its own set of conservation equations. This approach is also known as Eulerian-Eulerian model.

The Two Fluid Model as currently known was first presented by Drew and Segel, 1971. This approach was widely applied for vertical pipes in different fields, as Krepper, Konar, and Egorov, 2007 or Hosokawa and Tomiyama, 2009. Simulations using such a simple geometry allows the evaluation of different models before applying them to large scale facilities.

In this approach, the phases are treated as one continuous and one dispersed, which, in this work, continuous phase represents water in liquid state and dispersed phase stands for vapor. During the average process, the use of the void fraction is introduced and new terms are required in order to closure the system. The interfacial momentum transfer term is presented and its proper implementation becomes critical to get a correct prediction of the flow behavior.

Next subsections are presented following same structure as section 3.2. First, original governing equations are presented in their extended form and the dif-

ferent 1D equations to replace the former are explained and the substitution is justified. Closure models are introduced after governing equations. There are many different closure models and they are presented according to the corresponding effect where they are involved, e.g interfacial force models, wall drag or subcooled boiling model.

3.4.1 Mass conservation equation

The mass conservation equation for a phase in a two-phase flow system in the built-in twoPhaseEulerFoam solver is given as follows:

$$\frac{\partial(\alpha_k \rho_k)}{\partial t} + \nabla \cdot (\alpha_k \rho_k U_k) = 0. \tag{3.27}$$

Where *k* represents the phase, which can be dispersed (k = v) or continuous (k = l). Term α_k represents the void fraction, U_k the velocity of the phase *k* and ρ is the density of the phase. One of the requirements of this work was to consider the contribution of the mass transfer between phases, in case it exists.

Let be Γ_k the mass gained by phase k. It is important to note that the increase of mass in one phase is equivalent to the loss of mass in the other phase ($\Gamma_v = -\Gamma_l$), so that the conservation is satisfied in the system.

As in section 3.2, assuming X axis as the preferential direction, the final one-dimensional two-phase equation implemented for a general phase *k* stays as follows

$$\frac{\partial(\alpha_k \rho_k)}{\partial t} + \frac{\partial(\alpha_k \rho_k U_k)}{\partial x} = \Gamma_k. \tag{3.28}$$

According to Rusche, 2002 and Ghione, 2011, the compressible equation 3.28 needs to be rearranged before being implemented in the code, since a direct discretization of this equation is unstable due to large density ratios between phases. Furthermore, solving the equation form in 3.28, boundedness is satisfied for $\alpha_k > 0$, but it may not be fulfilled for $\alpha_k < 1$. In this case, it is necessary to perform a rearrangement in equation 3.28 that allows to ensure the conservation.

One formulation that allows to apply a numerical method which is bounded in both sides is presented in Weller, 2005, which departs from defining the phase

velocity as a function of the average and relative velocity. Using the dispersed phase, the velocity is

$$U_k = U + \alpha_i U_r. \tag{3.29}$$

The relative velocity is given as

$$U_r = U_k - U_i \tag{3.30}$$

and the average velocity is

$$U = \alpha_k U_k + \alpha_i U_i, \tag{3.31}$$

On the other hand, mass continuity equation term are separated between those related to density and those related to transport. Applying the product rule for derivatives to 3.28, it can be expressed as

$$\rho_k \frac{\partial \alpha_k}{\partial t} + \alpha_k \frac{\partial \rho_k}{\partial t} + \rho_k \nabla \cdot (\alpha_k U_k) + \alpha_k U_k \nabla \cdot (\rho_k) = \Gamma_k, \tag{3.32}$$

which, according to Ghione, 2011, using the definition of total derivative and dividing by ρ_k, one can obtain

$$\frac{\partial \alpha_k}{\partial t} + \nabla \cdot (\alpha_k U_k) + \frac{\alpha_k}{\rho_k} \frac{D(\rho_k)}{\partial t} = \frac{\Gamma_k}{\rho_k}. \tag{3.33}$$

Introducing now the definition of phase velocity, equation 3.33 applied to dispersed phase can be written as

$$\frac{\partial \alpha_k}{\partial t} + \nabla \cdot (\alpha_k U_k) + \nabla \cdot (U_r \alpha_k (1 - \alpha_k)) + \frac{\alpha_k}{\rho_k} \frac{D(\rho_k)}{\partial t} = \frac{\Gamma_k}{\rho_k}. \tag{3.34}$$

Substituting 3.34, for both dispersed and continuous phase and rearranging, each phase equation states as

$$\frac{\alpha_k}{\rho_k} \frac{D(\rho_k)}{Dt} + \frac{\partial (\alpha_k)}{\partial t} + \nabla \cdot (\alpha_k U) + \nabla \cdot (\alpha_k \alpha_i U_r) = \frac{\Gamma_k}{\rho_k} \tag{3.35}$$

$$\frac{\alpha_i}{\rho_i}\frac{D(\rho_i)}{Dt} + \frac{\partial(\alpha_i)}{\partial t} + \nabla\cdot(\alpha_i U) - \nabla\cdot(\alpha_k\alpha_i U_r) = \frac{\Gamma_i}{\rho_i} \tag{3.36}$$

and summing both phases so that boundedness of the void fraction between 0 and 1 is guaranteed (Ghione, 2011), one leads to

$$\nabla\cdot(U) = \frac{\Gamma_k}{\rho_k} + \frac{\Gamma_i}{\rho_i} - \frac{\alpha_k}{\rho_k}\frac{D(\rho_k)}{Dt} - \frac{\alpha_i}{\rho_i}\frac{D(\rho_i)}{Dt}, \tag{3.37}$$

where $\alpha_k + \alpha_i = 1$. Equation 3.37 considers compressibility and phase change on the right hand side of the equation. Thus, this equation resembles to the incompressible version of continuity equation ($\Delta U = 0$), but it takes into account compressibility of both phases and phase change at the right-hand side. This allows that, if either continuous or dispersed phase tends to zero, the problems of divergence are removed due to the new compressible terms.

Substituting 3.37 in 3.35 and 3.36, and taking into account the product rule $\nabla\cdot(\alpha_k U) = \alpha_k\nabla\cdot(U) + U\cdot\nabla(\alpha_k)$, the following expression for phase fraction is given

$$\begin{aligned}\frac{\partial(\alpha_k)}{\partial t} + U\cdot\nabla(\alpha_k) + \nabla\cdot(\alpha_k\alpha_i U_r) = \\ \frac{\alpha_k^2}{\rho_k}\frac{D(\rho_k)}{Dt} - \frac{\alpha_k\alpha_i}{\rho_i}\frac{D(\rho_i)}{dt} + \frac{\alpha_i}{\rho_i}\frac{D(\rho_i)}{dt} - \alpha_k\left(\frac{\Gamma_k}{\rho_i} + \frac{\Gamma_k}{\rho_k}\right) + \frac{\Gamma_k}{\rho_k}\end{aligned} \tag{3.38}$$

and assuming that $\alpha_k^2 - \alpha_k = \alpha_k(1-\alpha_k)$, the formulation for both phases can be written as

$$\begin{aligned}\frac{\partial(\alpha_k)}{\partial t} + U\cdot\nabla(\alpha_k) + \nabla\cdot(\alpha_k\alpha_i U_r) = \\ \alpha_k\alpha_i\left(\frac{1}{\rho_i}\frac{D(\rho_i)}{Dt} - \frac{1}{\rho_k}\frac{D(\rho_k)}{dt}\right) + \alpha_k\left(\frac{\Gamma_k}{\rho_i} - \frac{\Gamma_k}{\rho_k}\right) + \frac{\Gamma_k}{\rho_k}\end{aligned} \tag{3.39}$$

$$\begin{aligned}\frac{\partial(\alpha_i)}{\partial t} + U\cdot\nabla(\alpha_i) - \nabla\cdot(\alpha_k\alpha_i U_r) = \\ \alpha_k\alpha_i\left(\frac{1}{\rho_k}\frac{D(\rho_k)}{Dt} - \frac{1}{\rho_i}\frac{D(\rho_i)}{dt}\right) + \alpha_i\left(\frac{\Gamma_i}{\rho_i} - \frac{\Gamma_i}{\rho_k}\right) - \frac{\Gamma_i}{\rho_i}.\end{aligned} \tag{3.40}$$

Only phase k (dispersed) is solved in the code. The continuous phase fraction α_i is calculated as $\alpha_i = 1 - \alpha_k$.

Since the objective of this work is to solve a one-dimensional system, one can take only one preferential direction and simplify equation 3.39 as

$$\frac{\partial(\alpha_k)}{\partial t} + \frac{\partial(\alpha_k U)}{\partial x} - \alpha_k \frac{\partial U}{\partial x} + \frac{\partial(\alpha_k \alpha_i U_r)}{\partial x} = \alpha_k \alpha_i \left(\frac{1}{\rho_i} \frac{D(\rho_i)}{dt} - \frac{1}{\rho_k} \frac{D(\rho_k)}{dt} \right) + \alpha_k \left(\frac{\Gamma_k}{\rho_i} - \frac{\Gamma_k}{\rho_k} \right) + \frac{\Gamma_k}{\rho_k}. \quad (3.41)$$

Finally, in order to reach the implemented equation, it is necessary to recall the condition which states that the mass gained by one phase is equivalent to the mass lost by the other phase. Furthermore, two different terms has been implemented to represent the mass transfer term:

- Γ_{Evap}, also called Γ_{ik}, which represents the evaporated mass, that is mass transferred from phase l (continuous, phase i) to phase v (dispersed, phase k).
- Γ_{Cond}, also called Γ_{ki}, which represents the condensed mass, which is the mass transferred from phase v (dispersed, phase k) to phase l (continuous, phase i).

Considering these terms, and defining Γ_{Cond} as negative when condensation takes place, one can draw the conclusion that $\Gamma_v = \Gamma_{Evap} + \Gamma_{Cond}$. Further description of these terms will be given in section 3.9.

The method presented above and proposed by Weller, 2005 is not the only method to overcome the boundedness and density gradients problems. An alternative method to solve unboundedness was proposed by Spalding, 1985, which consisted in weighting the final void fraction after solving equation 3.28 for each phase. The proposed weighting after solving α_v and α_l was the following

$$\alpha_v^* = \frac{\alpha_v}{\alpha_v + \alpha_l}. \quad (3.42)$$

This new weighted void fraction would overwrite the original α_v and this causes that only when convergence is reached, phase continuity is satisfied. Furthermore, according to Rusche, 2002, the new void fraction α_v^* is only bounded when both α_v and α_l are larger than zero.

Considering that the new solver should work for both single-phase and two-phase flows systems, it may happen that one of the phases tends to zero. Therefore, the method proposed by Spalding, 1985 was discarded for this work.

3.4.2 Momentum conservation equation

Regarding to momentum equation, the formulation used in twoPhaseEulerFoam is

$$\frac{\partial(\alpha_k \rho_k U_k)}{\partial t} + \nabla(\alpha_k \rho_k U_k U_k) = -\alpha_k \nabla p + \nabla \cdot [\alpha_k(\tau_k + \tau_k^t)] + \alpha_k \rho_k g + M_k, \quad (3.43)$$

where the subscript *k* represents the phase (e.g. continuous or dispersed phase). Departing from the beginning of the left-hand side of the eq. 3.43, the different terms represent the change of momentum in the fluid with time and the rate of change of momentum through the faces of the volume. On the right-hand side, following the same order, one can find the change in pressure due to the different forces, terms $\tau_k + \tau_k^t$ are the combined Reynolds viscous and turbulent stress, the gravity term and the averaged interfacial momentum transfer term.

In order to implement the momentum equation considering the effects due to mass transfer between phases, the momentum transfer during the phase change, $\Gamma_k U_k$, is added to the equation. Similar to section 3.2, turbulence (Reynolds stresses) in the new system is simplified and replaced by the axial viscous term, which is considered in the wall friction model.

Another assumption which is taken in this work is that the **pressure *p* is equal in both phases**, as well as the interfacial pressure. So there is only one pressure in the volume.

In order to introduce the equation which has been implemented in the code, one have to recall equation 3.43 and, applying the chain derivative rule to the left-hand side of the equation and considering the assumptions already made, the equation can be written as

$$\alpha_k \rho_k \frac{\partial(U_k)}{\partial t} + U_k \frac{\partial(\alpha_k \rho_k)}{\partial t} + \alpha_k \rho_k U_k \nabla \cdot (U_k) + U_k \nabla \cdot (\rho_k \alpha_k U_k) = - \alpha_k \nabla p + \alpha_k \rho_k g + \Gamma_{ki} U_i - \Gamma_{ki} U_k + M_{ki}. \quad (3.44)$$

Focusing on the left-hand side of the equation, this can be rearranged as (Holzinger, 2016)

$$U_k \underbrace{\left(\frac{\partial(\alpha_k\rho_k)}{\partial t} + \nabla\cdot(\rho_k\alpha_k U_k)\right)}_{I} + \alpha_k\rho_k \underbrace{\left(\frac{\partial(U_k)}{\partial t} + U_k\nabla\cdot(U_k)\right)}_{II} = R.H.S \tag{3.45}$$

Where the first term (I) corresponds to the phase continuity equation, $\nabla\cdot U$, which is equal to zero. The second term on the left-hand side (II), in particular, the terms included within the parenthesis, represent the total derivative of the velocity. Therefore equation 3.45 can be written as

$$\alpha_k\rho_k\frac{D(U_k)}{Dt} = R.H.S. \tag{3.46}$$

Left-hand side in equation 3.45 can also be written as

$$U_k\left(\frac{\partial(\alpha_k\rho_k)}{\partial t} + \nabla\cdot(\rho_k\alpha_k U_k)\right) + \alpha_k\rho_k\left(\frac{\partial(U_k)}{\partial t} + U_k\nabla\cdot(U_k)\right) = \frac{\partial(\alpha_k\rho_k U_k)}{\partial t} + \nabla\cdot(\alpha_k\rho_k U_k U_k), \tag{3.47}$$

which can be reorganized as follows

$$\frac{\partial(\alpha_k\rho_k U_k)}{\partial t} + \nabla\cdot(\alpha_k\rho_k U_k U_k) - U_k\left(\frac{\partial(\alpha_k\rho_k)}{\partial t} + \nabla\cdot(\rho_k\alpha_k U_k)\right) = \alpha_k\rho_k\left(\frac{\partial(U_k)}{\partial t} + U_k\nabla\cdot(U_k)\right) \tag{3.48}$$

Mathematically speaking, one could say that

$$\frac{\partial(\alpha_k\rho_k U_k)}{\partial t} + \nabla\cdot(\alpha_k\rho_k U_k U_k) - U_k(\frac{\partial(\alpha_k\rho_k)}{\partial t} + \nabla\cdot(\rho_k\alpha_k U_k)) = R.H.S \tag{3.49}$$

and

$$\alpha_k \rho_k (\frac{\partial (U_k)}{\partial t} + U_k \nabla \cdot (U_k)) = R.H.S \tag{3.50}$$

are equivalent (because term I in equation 3.45 was assumed to be zero). Nonetheless,when one now discretizes both equations in order to solve them numerically, the left hand sides of both equations 3.49 and 3.50 might actually be different, as the discretised phase continuity equation might not be equal to zero.

Adding the right-hand side to the equation 3.49, the final three-dimensional momentum transport formulation is

$$\frac{\partial (\alpha_k \rho_k U_k)}{\partial t} + \nabla \cdot (\alpha_k \rho_k U_k U_k) - U_k (\frac{\partial (\alpha_k \rho_k)}{\partial t} + \nabla \cdot (\rho_k \alpha_k U_k)) = \\ - \alpha_k \nabla p + \alpha_k \rho_k g + \Gamma_{ki} U_i - \Gamma_{ki} U_k + M_{ki} \tag{3.51}$$

And taking only axial direction, e.g X axis, the one-dimensional momentum equation can be written as

$$\frac{\partial (\alpha_k \rho_k U_k)}{\partial t} + \frac{\partial (\alpha_k \rho_k U_k U_k)}{\partial x} - U_k (\frac{\partial (\alpha_k \rho_k)}{\partial t} + \frac{\partial (\rho_k \alpha_k U_k))}{\partial x} = \\ - \alpha_k \frac{\partial p}{\partial x} + \alpha_k \rho_k g + \Gamma_{ki} U_i - \Gamma_{ki} U_k + M_{ki} \tag{3.52}$$

According to Rusche, 2002, this linear system cannot be directly solved, but it needs to be calculated by steps to avoid instabilities. First, an estimated velocity which does not satisfy continuity equation will be obtained. Then, the pressure will be solved, obeying this time continuity. Finally, the velocity will be corrected using the updated pressure. This method is called *segregated method* and it is explained in detail in section 4.4.

3.4.3 Energy conservation equation

The energy conservation equation for a two-phase flow system, in terms of specific enthalpy, and formulated for a generalized phase is given by the following:

$$\frac{\partial(\alpha_k\rho_k h_k)}{\partial t} + \nabla\cdot(\alpha_k\rho_k h_k U_k) + \frac{\partial\left(\alpha_k\rho_k K\right)}{\partial t} + \nabla\cdot(\alpha_k\rho_k U_k K) = \\ -\nabla\cdot[\alpha_k(q_k + q_k^t)] + \frac{\partial(\alpha_k p)}{\partial t} + \nabla\cdot(\alpha_k T_k U_k) + \alpha_k\rho_k U_k + S_k. \quad (3.53)$$

Where K states for the kinetic energy, $q_k + q_k^t$ account for both the thermal and the turbulent convection heat fluxes, T_k represents the work due to shear forces and S_k is the interfacial supply energy. In solving problems, it is often useful to separate the mechanical and thermal effects in the total energy equation. In these cases, the mechanical equation is substracted from the thermal energy equation (Todreas and Kazimi, 2011). This method is applied here, in order to focus the explanation on enthalpy equation, as it was applied in section 3.2.

The final system proposed in this work must also consider the possible interfacial energy due to the phase change, and this is done by adding to the equation the term $\Gamma_k h_{k,sat}$, which accounts for the heat transferred from phase k to the interface. Assuming x axis as the preferential direction, replacing turbulence work by simplified models that consider the axial viscosity in terms of wall friction and heat transfer and rearranging, the final one-dimensional energy equation for two-phase flow for a general phase *k* used in this work is given as

$$\frac{\partial(\alpha_k\rho_k h_k)}{\partial t} + \nabla\cdot(\alpha_k\rho_k h_k U_k) + \nabla\cdot[\alpha_k(q_k + q_k^t)] = \\ \frac{\partial(\alpha_k p)}{\partial t} - \alpha_k\rho_k U_k g + \Gamma_k h_{k,sat} + q_{ik}^{''} a_i + q_k^{'''} \quad (3.54)$$

where *k* and *i* represent the phases (e.g. dispersed and continuous).
Following the same order as in previous equations, the terms in the right-hand side of the equation represent the change of enthalpy with time, the rate of change of enthalpy through the faces of the volume and the heat due to thermal conduction and to the turbulence convection and shear effects within the field. On the left-hand side, the terms represent the rate of work done, the potential energy of the fluid, the heat due to the phase of change, the interfacial heat transfer source and the volumetric source.

The source term that represents the phase change, $\Gamma_k h_{k,sat}$ accounts for the mass energy added to the phase. In this case, the sub index *ik* means that the energy flux is transferred from phase *i* to phase *k*.

Since turbulence is not taken into account in this work, the term that represents the heat due to turbulence and shear effects can be neglected. Furthermore, only contributions from wall heat flux and interfacial heat flux are considered. Therefore, moving the heat due to conduction to the right-hand side of the equation and re-writing so that only these two phenomena are considered, equation 3.54 can be expressed as

$$\frac{\partial(\alpha_k \rho_k h_k)}{\partial t} + \nabla \cdot (\alpha_k \rho_k h_k U_k) = -\alpha_k q_k'' + \frac{\partial(\alpha_k p)}{\partial t} - \alpha_k \rho_k U_k g + \Gamma_k h_{k,sat} + q_{ik}'' a_i + q_k''' \quad (3.55)$$

Moreover, it is assumed that no heat generation source exists within the fluid, so the term q_k''' can be neglected. Following the same procedure as in previous sections, the one-dimensional energy equation is given by the following

$$\frac{\partial(\alpha_k \rho_k h_k)}{\partial t} + \frac{\partial(\alpha_k \rho_k h_k U_k)}{\partial x} = \frac{\partial(\alpha_k p)}{\partial t} - \alpha_k q_{w,k}'' - \alpha_k \rho_k U_k g + \Gamma_k h_{k,sat} + q_{ik}'' a_i \quad (3.56)$$

3.5 Solid conduction equation

The structures that surrounds the fluid, whether pipes, fine plates or vessels, may influence in the local regions close to the wall, as seen in section 3.12. The heat may reach the water through these structures and the balance fluid-solid becomes important in order to keep a proper heat transfer so that the solid structures do not reach a temperature that might compromise their internal properties, consequently compromising the whole installation.

The general heat conduction equation is given by the following

$$\underbrace{\rho C_p \frac{DT}{\partial t}}_{\text{I}} + \underbrace{\nabla \cdot \overline{q}}_{\text{II}} = \underbrace{q'''}_{\text{III}} \quad (3.57)$$

In equation 3.57, the first term in right hand side (I) represents the change of temperature with time and the thermal expansion in the three dimensions. This thermal expansion considers possible convective heat transfer due to rotational

o translational motion of the solid. The following term (II) stands for the heat transfer through the solid. The volumetric heat on the left hand side (III) accounts for the volumetric heat generation rate coming from a possible heat source in the system.

The heat flux in term (II) in equation 3.57, which represents the conduction heat transfer is defined by Fourier's law and can be expressed as

$$q = -\lambda \nabla T \tag{3.58}$$

where λ is the thermal conductivity and C_p in equation 3.57 is the specific heat. Equation 3.58 can be substituted in equation 3.57, and rewriting the equation, one gets the following

$$\rho C_p \frac{DT}{\partial t} = \nabla \cdot (\lambda \nabla T) + q''' \tag{3.59}$$

This work is meant to develop a one-dimensional solver, as mentioned before. In previous equations, generally the axial direction is taken, since this is the direction of the fluid motion, but in this case the radial direction is preferably against the axial direction. Taking this option, the direction of the heat transfer is followed. The axial heat transfer can also be considered by solving equation 3.59 in 2D, but in general, it is going to be assumed that the axial conduction is significantly small in comparison to the radial heat transfer and it can be neglected. In this work, in addition, the thermal expansion is also neglected. Therefore, the heat transfer conduction in the radial direction is given by the following

$$\rho C_p \frac{\partial T}{\partial t} = \frac{\partial}{\partial y}\left(\lambda \frac{\partial T}{\partial y}\right) + q'''. \tag{3.60}$$

Equation 3.60 is only solved in the solid structure. The key point in the system will be determining the boundary conditions so that the balance between fluid heat flux and solid heat flux is accomplished. In order to verify the balance, radial conduction term $\lambda \frac{\partial T}{\partial y}$ will be used. From now on, this term will be called as q''_{cond}. The boundary condition determined for the interface between fluid and solid is defined in chapter 4. No radiation heat transfer is considered in the solver, as it is not considered in TRACE system code.

3.6 Two-phase closure models

In chapter 3, the system to solve the main variables that characterize the fluid has been set up. However, there is no a proper relation between phases unless the terms that couple the phases are defined. Dispersed and continuous phases affect each other by means of the interfacial mass, momentum and heat transfer, and wall momentum and heat transfer. These effects need to be defined with an extra set of equations. One can find a wide variety of correlations that agree with different experimental data sets. Since this work is focused on a one-dimensional system, the field can be reduced significantly. Furthermore, these closure models are dependent on the behavior of the phases, and this behavior is characterized in different regimes. In this work, only bubbly and slug regimes are considered. Therefore, the selected closure equations have been taken from the subgroup that accomplish these two assumptions. It is worth mentioning that all selected closure models are applied in this work generally for both bubbly and slug flows, without distinctions among these regimes.

3.6.1 Flow regimes in vertical pipes

As the flow becomes two-phase, the complexity of the phase interactions increase and so does the prediction difficulty. The behavior depends on the size of bubbles, the relative flow rates and the channel orientation, among others. The regime is not only important because of the behavior, but also because the heat transfer varies according to the pattern. In order to get an optimum heat transfer between phases and between flow and external systems, it is fundamental to know the pattern and the characteristics of the flow.

Wallis, 1969 stated a classification used still nowadays and shown in figure 3.2 for vertical pipes.

The prediction of the behavior of the two-phase flow starts by predicting the proper pattern, which is a critical feature in the study and development of computational codes for two-phase flows, due to the interest for many other industrial applications, such as the chemical, aerospace and power industry.

The bubbly flow is the first regime that appears in two-phase flows. It is characterized by small and dispersed bubbles with a spherical size. This is the most studied pattern, so it is an adequate pattern to perform validations. As the bubble size increases, either because of boiling effects or pressure drop and expansion phenomena, the bubbles get a spherical cap and a ragged, relatively flat bottom and the flow enters to slug regime. According to Wallis, bubbly regime is

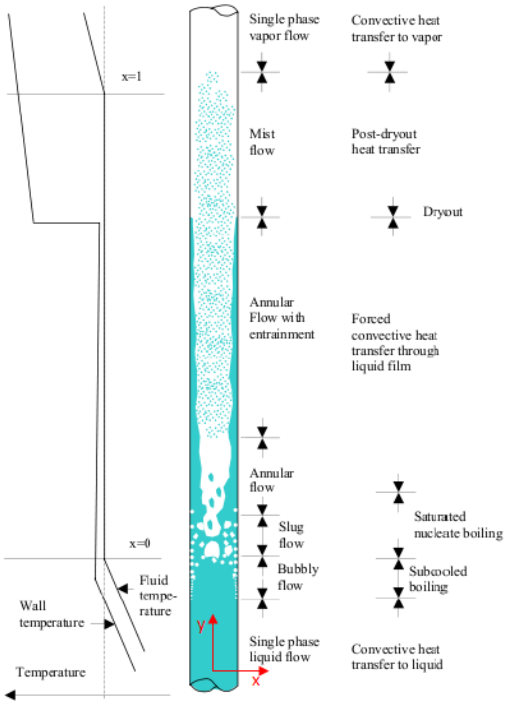


Figure 3.2: Flow patterns and heat transfer regimes in vertical pipes (H.Anglart, 2010).

always observed when the void fraction is below 0.2 (based on air-water experiments). However, in this work, Ishii and Mishima recommendation is followed and the transition to slug regime is assumed for a void fraction of 0.3. The upper limit of slug bubble regime is defined as 0.5, as Taitel, Bornea, and Dukler recommended. They suggest that the maximum void fraction at which the bubbles remain rather spherical is 0.52, but this value decreases as the space between bubbles falls down. Therefore, the maximum limit of 0.5 is close enough and it considers that the bubbles are not ideal.

As the flow evolves and bubbles and consequently, void fraction grows up above 0.5, the bubbles are big enough to be in constant coalescence, becoming a gas flow that runs along the bulk whereas the liquid phase is sent to the wall. This new regime is known as Annular Flow. When the gas flow (for this work purposes, vapor flow) is larger than the liquid flow at the walls, it starts dragging water drops dispersed in the gas flow. This regime is called Annular Flow with

entrainment. The flow can reach a state where the rest of fluid is dispersed as droplets within the gas flow, which is known as Mist Flow. Apart from the regime evolution presented, it is worth remembering that if boiling is presented, vapor is generated at the walls, generating more vapor apart from the vertical evolution of the flow. Finally, if the gas phase is hot enough to evaporate all the droplets, the gas single-phase is reached and the flow becomes again single-phase.

3.6.2 Time and Volume averaging

In multiphase flows, the presence of interfacial interactions introduces serious difficulties that increases the complexity of the problem. The bubbles interacts with the continuous phase and vice-versa, the former phase is deformable, becoming the interface deformable, and the same interface can fluctuate due to the motion of the fluid. This influence in the other phase creates the necessity of determining the macroscopic properties by using an averaging procedure (Ishii and Hibiki, 2011).

In this work, the two averaging operations used are the time-averaging and the volume-averaging procedures. The time-averaging process consists in the following.

Being a function $F = F(t, x)$, the time-averaging is given by

$$\overline{F} = \frac{1}{\Delta t} \int_{\Delta t} F(t, x) dt, \tag{3.61}$$

where $\overline{F}$ is the averaged value. Undertaking same procedure for volume-averaging, this process is given by

$$\overline{F} = \frac{1}{\Delta V} \int_{\Delta V} F(t, x) dV, \tag{3.62}$$

By proper averaging, the mean value of the properties can be obtained, eliminating the influence of the fluctuation. This is due to the fact that, since these averages are integral operations, the local fluctuation values will be smoothed within the integral domain.

3.7 Interfacial momentum transfer

According to Newton third law, the total force acting from continuous to dispersed phase is equal to the total force acting in the opposite direction, so $M_k + M_i = 0$. M_k and M_i are the total momentum transfer terms affecting to the phase *k* and phase *i* respectively.

The interfacial momentum transfer expression is the most significant closure equation in a two-phase flow (Ishii and Hibiki, 2011). In a general system and for a generalized phase, M_k is the sum of different forces

$$M_k = M_k^D + M_k^L + M_k^{WL} + M_k^{VM} + M_k^{TD}, \tag{3.63}$$

which represent the drag force, the lift force, the wall lubrication force, the virtual mass and the turbulent dispersion force, respectively.

Before introducing the different interfacial forces, the concept of *Sauter mean diameter*, also called, Sauter diameter, needs to be explained because it is involved in various of the following forces. This concept represents an average of a size. Particularly, it is defined as the diameter of a spherical bubble that has the same ratio of volume to surface area as the target particle. Therefore, as mentioned before, it is used as a characteristic mean. Let *b* be a general bubble, then its surface diameter can be given as

$$d_s = \sqrt{\frac{A_b}{\pi}} \tag{3.64}$$

and its volume diameter as

$$d_v = \left(\frac{6V_b}{\pi}\right)^{1/3} \tag{3.65}$$

being A_b and V_b the area and volume of the bubble. The final Sauter diameter of the bubble is

$$S_d = d_{32} = \frac{d_v^3}{d_s^2}. \tag{3.66}$$

If the general bubble *b* is assumed to be an sphere, the Sauter diameter can be rewritten as

$$S_d = d_{32} = \frac{4/3\pi(d_v/2)^3}{4\pi(d_s/2)^2} = \frac{(d_v/2)^3}{3(d_s/2)^2} = \frac{d_{32}}{6} \tag{3.67}$$

The complete development to obtain the Sauter diameter can be found in Kowalczuk and Drzymala and in Azzopardi works.

The drag force represents the opposition to the motion of bubbles in the continuous phase, so it affects directly to the axial velocity, influencing its terminal value. Its general representation is

$$M_k^D = C_D(U_k - U_i) \tag{3.68}$$

Where C_D represents the drag coefficient and $U_k - U_i$ is the relative velocity between the two phases. Many empirical models have been developed to represent this effect, such as Schiller-Naumann (Schiller, 1933), Ishii&Zuber (Ishii and Zuber, 1979) or Tomiyama (Tomiyama et al., 1998).

The lift force is a lateral force that push bubbles towards the wall or towards the centerline of the bulk fluid. It plays an important role in the radial bubbles distribution (Ghione, 2011). The direction to which the bubbles will be sent depends on the mean Sauter diameter of the bubble (Rollins, 2018), but for a spherical bubble, this force should act towards the wall. A common expression for this force is

$$M_k^L = C_{l,k}\alpha_k\rho_i U_k \cdot \nabla \cdot U_i \tag{3.69}$$

Where *k* and *i* are the dispersed and continuous phases, respectively. $C_{l,k}$ represents the lift coefficient, which can be estimated from an empirical correlation, such as Tomiyama (Tomiyama, 1998).

This force is not considered in one-dimensional simulations, due to the fact that the term $U_k \cdot \nabla \cdot U_i$ is voided in such situations.

The wall lubrication force also acts in radial direction. This force pushes small bubbles away from the wall to avoid higher bubble concentrations close to the wall. The model of this force is given by the following

$$M_k^{WL} = C_{wl}\alpha_k\rho_i|U_r - (U_r \cdot \vec{n_w})\vec{n_w}|^2(-\vec{n_w}) \tag{3.70}$$

where n_w represents the vector normal to the wall. Again, since this force acts in a perpendicular direction respect the axial velocity, it is neglected in one-dimensional simulations.

The virtual mass force represents the inertia produced by a change in the velocity of one of the phases. Its general correlation is given by

$$M_k^{VM} = C_{vm}\alpha_k\rho_i\left(\frac{DU_k}{dt} - \frac{DU_i}{dt}\right). \tag{3.71}$$

This force has very little influence when running steady state simulations, but it allows to increase the accuracy on transient flow developments.

The turbulent dispersion force takes into account the effects that the fluid turbulence may have on the bubbles. According to Rollins, 2018, the influence of this force is proportional to the gradient of void fraction. There is not a general model for this force, so here it is presented the model derived by Lopez de Bertodano (Bertodano, 1993).

$$M_k^{TD} = C_{td}(-\rho_i)k_i\nabla\alpha_k. \tag{3.72}$$

In one-dimensional simulations this force is not considered, since it represents microscopic effects in different directions while a one-dimensional analysis takes only macroscopics effects in a preferential direction (Ishii and Hibiki, 2011).

Therefore, only correlations for drag force have been implemented in this solver.

3.7.1 Interfacial Drag Force

It is not difficult to see that in one-dimensional analysis, the drag force represents the main contribution to the coupling degree between phases. The drag model is regime dependent and Reynolds number dependent (Ishii and Mishima, 1984). Within the drag force models for bubbly and slug regimes, up to four subregimes can be found according to different authors. These are the viscous, the distorted-particle, the churn-turbulent-flow and the slug-flow regime, which are presented in order from the more dispersed (so the bubbles are the smallest) to the more slug, respectively. Therefore, as the Reynolds number increases and

the bubbles grow up, the flow evolves from the viscous regime to the different regimes until the slug-flow, where the bubbles move in an erratic way. If the Reynolds number continues to increase, the bubbles might turn into cap bubbles, leading to a constant drag coefficient.

The drag correlation is different according to the regime. In general, this work is focused in viscous regimes, but one of the four implemented models, that is, the drift flux model, considers also the evolution of the system until the slug-flow regime.

In this section, two different approaches for this model are presented, the drift flux model and the drag correlation approach. Both correlations have been implemented in the code, giving good agreements with the empirical data. Both are empirical correlations, with different perspective. The drift flux model is based on the mixture motion to solve both phases, while the drag correlation approach is an intermediate step in the mechanistic model aim. It is based on the buoyancy force, but it needs an empirical coefficient to adjust it. Within the latter approach, three different correlations are included in the solver, so that a significant extension of the literature can be covered, since many different correlations for drag force have been stated over the years. The difference among these three models is basically the calculation of the drag coefficient C_d. Of these three approach, in two of them the only difference is the value of the constants, which according to different authors, vary. The third one considers the spatial distribution of the bubbles, and it also considers the expansion of these along the pipe.

Drift Flux Correlation

The first model is a drift flux based model, which principle is that both continuous and dispersed phases are solved as an indivisible system (Peña-Monferrer et al., 2018c). In order to calculate the drag coefficient, a balance in the direction of flow is performed considering pressure, buoyancy and the interfacial drag between both phases. Assuming same pressure for both phases, as mentioned in section 3.4 and total interfacial momentum transfer to be zero, the drag momentum term can be written as

$$C_D = \frac{\alpha_k(1-\alpha_k)g\Delta\rho}{U_r^2} \tag{3.73}$$

where α, g and $\Delta\rho$ represent the void fraction, gravity and the difference between phase densities. Now, according to NRC, 2013, it remains the substitution of U_r^2 by the drift flux model.

In this model, the velocity is void-weighted and area-averaged, which is represented as

$$\bar{U_k} = \frac{\langle \alpha_k U_k \rangle}{\langle \alpha_k \rangle}, \tag{3.74}$$

where $\langle\rangle$ denotes the average over the cross sectional area and *k* represents the phase.

Thus, the relative velocity is given by the following

$$\bar{U_r} = \bar{U_g} - \bar{U_l}. \tag{3.75}$$

According to (Zuber and Findlay, 1965), the local drift velocity term can be expressed as

$$v_{gj} = U_g - j, \tag{3.76}$$

where v_{gj} takes into account the relative velocity between both phases, and *j* represents the total volumetric flux of the mixture. If one expresses equation 3.76 in void-averaged area-averaged terms and rearranges the terms, the following can be obtained

$$\bar{v}_{gj} = \frac{\langle \alpha v_{gj} \rangle}{\langle \alpha_k \rangle} + \frac{\langle j \rangle}{\langle \alpha_k \rangle}. \tag{3.77}$$

Introducing the distribution parameter coefficient, that considers the effects due to the spatial distribution of the phase and is defined as

$$C0 = \frac{\langle \alpha_k j \rangle}{\langle \alpha_k \rangle \langle j \rangle}, \tag{3.78}$$

the average velocity of the dispersed phase can be expressed in terms of the drift flux velocity and the distribution parameter in the following way

$$\bar{v}_g = \frac{\langle \alpha v_{gj} \rangle}{\langle \alpha_k \rangle} + C0\langle j \rangle. \tag{3.79}$$

It should be mentioned that usually in this model $\langle U_g - U_l \rangle$ is not equal to $\bar{U_g} - \bar{U_l}$, unless the distribution parameter will be 1 (Caleb S. Brook, 2012). This is due to the local relative motion of the phases and to the integral effect of the phase and velocity distributions. This means that, due to the area averaging that it is applied, if the dispersed phase is locally concentrated in the high velocity region, this velocity must be higher than the continuous velocity, which will be concentrated in the low region velocity, generally close to the wall.

(Ishii and Mishima, 1984) showed that the averaged relative velocity can be expressed by the following

$$\bar{U_r} \approx \frac{\bar{v_{gj}}}{1 - \langle \alpha_k \rangle} = \frac{1 - C0\langle \alpha_k \rangle}{1 - \langle \alpha_k \rangle} \bar{U_g} - C0\bar{U_l} \tag{3.80}$$

And writing equation 3.73 in terms of averaged relative velocity, one can read

$$C_D = \frac{\alpha_k(1-\alpha_k)g\Delta\rho}{\langle U_g - U_l \rangle^2} \frac{\langle U_g - U_l \rangle^2}{U_r^2} = \frac{\alpha_k(1-\alpha_k)^3 g\Delta\rho}{\bar{v_{gj}}^2} \frac{\left(\frac{1-C0\langle \alpha_k \rangle}{1-\langle \alpha \rangle}\bar{U_g} - C0\bar{U_l}\right)^2}{U_r^2} \tag{3.81}$$

Once the drag coefficient model is presented, it remains to define a correlation to calculate the drift flux velocity and the distribution parameter.

In order to calculate the drift flux velocity, only one type of bubble regime is considered, which is the churn-turbulent regime. Considering this regime, the model covers from a dispersed regime with small uniform bubbles to a slug flow, where larger bubbles flow through the pipe.

Ishii and Hibiki, 2011 introduced the following formula for this regime

$$\bar{v_{gj}} = \sqrt{2}\left(\frac{\sigma g \Delta\rho}{\rho_l^2}\right)^{1/4} \tag{3.82}$$

and the distribution parameter, which is used for all regimes, is given by

$$C0 = 1.2 - 0.2\sqrt{\frac{\rho_g}{\rho_l}} \tag{3.83}$$

However, as NRC, 2013 mentioned, as the void fraction increases, the bubbles grow up towards a Taylor cap regime and the flow might progressively reach slug situation. In these cases, where larger bubbles are presented and they might be agglomerated in some zones of the pipe, the previous model, which is devoted to uniform dispersed flows, is no longer accurate.

A new model has been implemented for slug regimes, based on Kataoka and Ishii, 1987. Now the drift flux velocity depends on a non-constant term

$$v_{gj}^{-} = v_{gj}^{-+}(\frac{\sigma g \Delta\rho}{\rho_l^2})^{1/4} \tag{3.84}$$

where v_{gj}^{-+} is known as the weighted drift velocity. This term is function of the pipe diameter, which is written in non-dimensional form, and is given by the following

$$v_{gj}^{-+} = 0.0019 \cdot Min[30, D_h^*]^{0.809}(\frac{\rho_g}{\rho_l})^{-0.157}(N_{\mu l})^{-0.562} \tag{3.85}$$

where dimensionless diameter is defined as

$$D_h^* = \frac{D_h}{\sqrt{\sigma/g\Delta\rho}} \tag{3.86}$$

and the continuous phase viscosity number is given as

$$N_{\mu l} = \frac{\mu_l}{(\rho_l \sigma \sqrt{\sigma/g\Delta\rho})^{1/2}}. \tag{3.87}$$

The value of *30* in equation 3.85 for the dimensionless diameter establishes a maximum limit between what is considered small pipes and large pipe diameters. As the pipe diameter increases, the bubbles could lead to cap bubble regime due to potential surface imbalances that might disintegrate them.

The distribution parameter has the same formulation for churn-turbulent phase (eq. 3.83).

The final drift flux velocity will be the maximum value of both models (eq. 3.82 and 3.84).

Drag Coefficient Approach (DCA)

The general expression to introduce the interfacial drag term is given by the following

$$C_D = \frac{1}{8} C_d \rho_k (U_k - U_i) a_i \tag{3.88}$$

Where, as in section 3.4, *k* and *i* represent dispersed and continuous phase respectively, a_i represents the interfacial area, which accounts for the area of the dispersed particle and C_d is the drag coefficient.

The drag coefficient model implemented in this work is based on the empirical correlation given by Ishii and Chawla, 1979 for viscous regimes, which is

$$C_d = \frac{24}{Re_k}(1 + 0.1 Re_k^{0.75}) \tag{3.89}$$

Re_k represents the Reynolds number, which is defined using a viscosity that is a mixture of both phases

$$Re_k = \frac{2 r_k \rho_i |U_k - U_i|}{\mu_m} \tag{3.90}$$

where μ_m is the mixture viscosity given by

$$\mu_m = \mu_i (1 - \alpha)^{-2.5 \frac{(\mu_k + 0.4\mu_i)}{(\mu_k + \mu_i)}} \tag{3.91}$$

The term r_k represents the radius of the dispersed particle (in this case, bubbles), which is defined as

$$r_k = \frac{0.5 We_k \sigma}{\rho_i (v_k - v_i)^2} \tag{3.92}$$

Using the radius, one can obtain the interfacial area used to calculate the drag coefficient term, which is also function of the void fraction and the volume of fluid. Since the bubbles are assumed to be sphericals, the interfacial area is calculated as

$$a_i = 3\alpha \frac{V}{r_k} \tag{3.93}$$

Drag Coefficient Approach considering Sauter diameter (DCA)*

This approach departs again from equations 3.88 and 3.89. The main difference of this model is the bubble diameter calculation and consequently, the interfacial surface area.

The drag coefficient depends on the flow parameters, and the bubble size. The maximum bubble diameter, $d_{\text{b,max}}$, is calculated from the critical Weber number:

$$We_{crit} = \frac{U_r^2 d_{b,max} \rho_l}{\sigma}. \tag{3.94}$$

When working with bubbles, a value of 10 is specified for the We_{crit} (Wallis, 1969). In this equation, v_r^2 refers to the velocity difference that gives the maximum bubble size (RELAP5/MOD3 code manual, 1995) instead of calculating the difference between the phase velocities. The Reynolds number is also calculated using this velocity. The following equation is applied:

$$U_r^2 = \max\left[(U_{\text{v}} - U_{\text{l}}), \frac{We_{crit}\sigma}{\rho_{\text{c}} \min(D' \alpha_d^{(1/3)}, D_h)}\right] \tag{3.95}$$

where D' is set to 0.005 m for bubbly/slug flows and D_h is the hydraulic diameter.

The bubble diameter is calculated from the maximum bubble diameter with the following assumption:

$$d_b = 0.5 d_{b,\text{max}} \tag{3.96}$$

The interfacial area concentration is then given in terms of the mean bubble diameter (Caleb S. Brook, 2012):

$$a_i = \frac{6\alpha\mu_m}{d_{32}} = \frac{3.6\alpha}{d_b} \tag{3.97}$$

where d_{32} is the Sauter mean diameter of the distribution related to the bubble diameter d_b assuming a Nukiyama-Tanasawa distribution (RELAP5/MOD3 code manual, 1995), which is a distribution for droplet diameter for a spray.

*Drag coefficient approach with specific drag closure and bubble size distribution (DCA**)*

Considering the bubble size distribution (BSD) at each node implies that, on the one hand the influence of the bubble size in the terminal velocity can be incorporated through the drag force, on the other hand the interfacial area can be computed directly from the definition of the Sauter mean diameter without any assumption.

The drag coefficient approach described in subsection 3.7.1 uses a set of assumptions that may influence the prediction of the two-phase flow behavior. In Peña-Monferrer et al., 2018c, a different approach that allows the validation of bubbly flow events is presented. This model proposes a drag coefficient correlation specific for the scenario, where the calculation of the dynamics is bubble size and area dependent. It is also based on the consideration of the BSD and its axial evolution. Considering the BSD at each node implies that, on the one hand the influence of the bubble size in the terminal velocity can be incorporated through the drag force, on the other hand the interfacial area can be computed directly from the definition of the Sauter mean diameter without any assumption.

The drag correlation used and implemented was proposed by Tomiyama et al., 1998 for contaminated systems

$$C_{\mathrm{D}} = \max\Big[\frac{24}{Re}(1 + 0.15Re^{0.687}), \frac{8}{3}\frac{\mathrm{Eo}}{\mathrm{Eo}+4}\Big]. \tag{3.98}$$

In expression 3.98, there is a region dominated by the Eötvös number (Eo). This approach requires an accurate representation of the bubble size in the system in order to have an appropriate calculation of the drag coefficient. Figure 3.3 shows the comparison of the terminal velocity of the bubbles as function of the equivalent diameter using the drag force coefficients of equations 3.89 and 3.98. The terminal velocity of a particle is that velocity resulting from the action of accel-

erating and drag forces. Most commonly it is the freefalling speed of a particle in still air under the action of gravity (Snowsill, 2010).

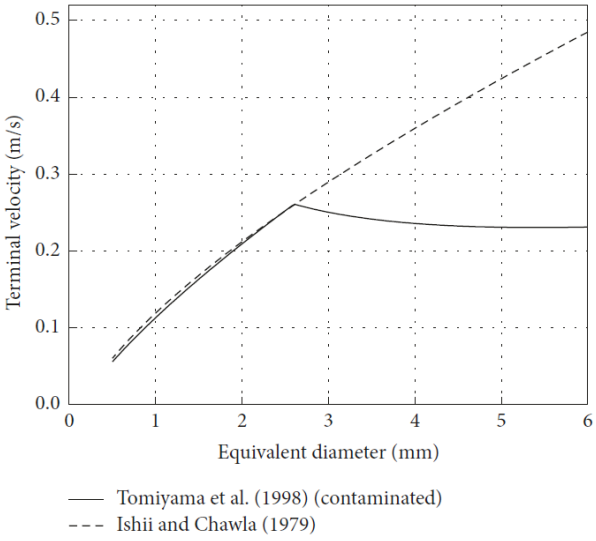


Figure 3.3: Terminal velocity for Tomiyama and Ishii (Peña-Monferrer et al., 2018c).

The difference in sizes between two different heights in a pipe, excluding the breakup and coalescence mechanisms, are due to the pressure changes. Note that, breakup and coalescence are neglected because of the bubbly flow conditions, which is a regime where these mechanisms do not appear.

The depressurization in the system and its influence in the bubbles size is described by an expansion factor f_i that is related to the inlet values (Peña-Monferrer et al., 2018c). At each node i the relation states

$$f_i = \left(\frac{\alpha_i}{\alpha_{\text{inlet}}}\right)^{\frac{1}{3}} \tag{3.99}$$

where α_i and α_{inlet} are the void fractions at the specified cell and at the inlet respectively.

If the bubbles change their size proportionally to the factor f_i, this means a relative increase of the bubble size to this factor and it is equivalent to the multiplication of an arbitrary variable by a constant amount. Consequently, the av-

erage or expected value of the BSD, which is dependent on the diameter value, is multiplied as well by this uniform value and the same is used for the standard deviation:

$$E[f_i d] = f_i E[d] \tag{3.100}$$

$$\mathrm{Var}[f_i d] = f_i^2 \mathrm{Var}[d] \tag{3.101}$$

Therefore, the BSD can be computed as a scaled concentration of the BSD at the different levels or nodes. A normal distribution at a particular cell or point would calculate the following statistical parameters related to the inlet:

$$\mu_i = f_i \mu_{\text{inlet}} \tag{3.102}$$

$$\sigma_i = f_i \sigma_{\text{inlet}} \tag{3.103}$$

From the definition of the numeric mean diameter given in 3.101, and assuming that the function $E(d)$ follows a normal distribution, a mean bubble diameter of the distribution can be defined as:

$$d_b = d_{10} = \frac{\int_0^\infty d^1 f(d) \mathrm{d}d}{\int_0^\infty d^0 f(d) \mathrm{d}d} = \mu \tag{3.104}$$

Knowing that the bubble size follows a normal distribution, the Sauter mean diameter in this case can be calculated from its popular formulation (Peña-Monferrer et al., 2018c).

In order to complete the calculation, the interfacial area concentration is also function of the Sauter mean diameter in the following way:

$$a_i = \frac{6\alpha_k}{d_{32}}. \tag{3.105}$$

3.8 Interfacial heat transfer

The heat is transferred from one field to the other one and this transference can occur in both directions (e.g from liquid to vapor and vice versa).The general formulation for the interfacial heat transfer is given by

$$q_{ki}^{''} = \gamma_{k,i}(T_k - T_{sat}), \tag{3.106}$$

where $\gamma_{k,i}$ is the interfacial heat transfer coefficient.

There are four different cases where interfacial heat transfer may happen:

- When liquid is subcooled
- When liquid is superheated
- When vapor is subcooled
- When vapor is superheated

Two of these situations correspond to evaporation heat and the other two accounts for the heat due to condensation. Every phenomena leads to a different heat transfer coefficient correlation. The models to account for these effects are regime dependent, so in this work only models for bubbly and slug flows are considered.

The bubbly flow is limited by the void fraction value, as mentioned in section 3.7.1. The maximum void fraction value for bubbly flow regime is generally 0.3, since at this point the bubbles move fast to Taylor or slug due to coalescence. However, this is also dependent on mass flux, since the higher the mass flux is, the more possibilities of break-up bubbles, keeping an homogeneous dispersed bubbly regime with larger void fractions. Considering this behavior, the criterion proposed by Choe, Weinberg, and Weisman was used in the solver, which first gives a maximum void fraction for bubbly regime, according to the mass flux of the system. Then, it establishes a ramp to move from dispersed bubbly regime to slug

$$\alpha_{disp} = \begin{cases} 0.3 & G < 2000.0 \\ 0.3 + 0.2\left(\frac{G-2000.0}{2700.0-2000.0}\right) & 2000 < G < 2700.0 \\ 0.5 & G > 2700.0 \end{cases} \tag{3.107}$$

where G is the mass flux, measured in $kg/(m^2 s)$.

Therefore, the closure models for the interfacial heat transfer coefficient presented in this work are used in systems that satisfies criterion 3.107.

If liquid is either subcooled or superheated, the heat transfer coefficient between the liquid and the interface is calculated by means of the Ranz-Marshall correlation. The general formulation is

$$\gamma_{l,i} = Nu \frac{\kappa_l}{D_{b,disp}} a_i \tag{3.108}$$

Where *Nu* is the Nusselt number, κ_l the liquid conductivity, $D_{b,disp}$ the diameter of bubbles and a_i is the interfacial area for this regime, which is given by equation 3.93. The Nusselt number in this model is given by

$$Nu = 2 + 0.6 Re_b Pr^{1/3} \tag{3.109}$$

Where the Reynolds number Re_b is calculated by equation 3.90 and Pr is the number of Prandtl.

The bubble diameter used in both Reynolds bubbles number and equation 3.108 for this regime is taken as an approximation to the general diameter for distorted bubbles and is given by a proportionality to Laplace coefficient (La)

$$D_b = 2 \cdot La = 2\sqrt{\sigma/(g\Delta\rho)}. \tag{3.110}$$

Therefore, equation 3.110 replaces equation 3.92 to calculate diameter. Then, with this value the solver calculates the interfacial area following equation 3.93.

There is one last consideration to take into account in this model, which is the calculation of the relative velocity used in the Reynolds bubble number. Since this solver is one dimensional, there is only one vapor velocity to represent both big and small bubbles, so the real dispersed velocity in bubbly regime should be lower than the difference between phase velocities, that is

$$U_{r,disp} << |U_g - U_l| \tag{3.111}$$

Therefore, the terminal velocity is used as an upper limit so that the relative velocity is more accurate

$$U_{r,disp} = min\left[|U_g - U_l|, U_{disp,term}\right] \tag{3.112}$$

The terminal velocity of a distorted bubble is given by Wallis, 1969 as follows

$$U_{term} = \sqrt{2}\left(\frac{\sigma g \Delta\rho}{\rho_l^2}\right)^{1/4} \tag{3.113}$$

which is generalized to the regime calculation using the approximation of Richardson and Zaki, 1997.

$$U_{disp,term} = U_{term}(1-\alpha_k)^{1.39} \tag{3.114}$$

This model is used for both liquid evaporation and condensation. Furthermore, when liquid is subcooled, it may happen that a portion of the fluid closer to the wall is being heated, generating vapor while the bulk is subcooled. The amount of heat used in this effect is considered in the subcooled boiling closure model, presented in section 3.12.

On the other hand, the interfacial heat transfer coefficient for vapor side when this is either subcooled or superheated is set as constant, $\gamma_{v,i} = 1000.0 W/m^2K$. This term has a very little influence in bubbly and slug regimes, so it could be neglected. However, in order to avoid large thermal disequilibrium, this value was selected, since it avoids thermal large discontinuities but still its value is very small. This value was also chosen due to its implication also in the reference system code model. Cases for superheated vapor are not considered in this work.

With these correlations one can calculate the heat due to evaporation and condensation. The interfacial heat allows also to quantify the transferred mass between phases, which will be explained in section 3.9

3.9 Interfacial mass transfer

The mass transfer term takes into account the mass transfer due to evaporation (from continuous phase to dispersed) and due to condensation (from dispersed phase to continuous). This was already mentioned in chapter early in section 3.4. This closure is specially relevant because it entails in mass and energy equations. The total mass transfer term is

$$\Gamma_k = \Gamma_{k,i} + \Gamma_{sub}, \tag{3.115}$$

where Γ_k accounts for the mass transfer per unit volume due to interfacial heat transfer and Γ_{sub} is the mass transfer occurred when subcooled boiling takes place. The former term covers both evaporation and condensation mass transfer terms, so $\Gamma_{k,i} = \Gamma_{Evap} + \Gamma_{Cond}$, while the latter only participates in those cases where subcooled boiling is happening.

The mass transfer rate caused by the heat transfer between phases is given by the following:

$$\Gamma_{k,i} = \frac{q'''_{l,i} + q'''_{v,i}}{h^*_v - h^*_l}. \tag{3.116}$$

The terms $q'''_{l,i}$ and $q'''_{v,i}$ are the interfacial heat transfer rate per unit volume, where the former represents the liquid-to-interface heat transfer rate (evaporation) while the latter accounts for the vapor-to-interphase heat transfer rate (condensation). These terms were introduced in section 3.8 (equation 3.106).

Following the description given so far, one can draw that the terms Γ_{Evap} and Γ_{Cond} can be defined separately, where

$$\Gamma_{Evap} = \frac{q'''_{l,i}}{h^*_v - h^*_l}, \tag{3.117}$$

and

$$\Gamma_{Cond} = \frac{q'''_{v,i}}{h^*_v - h^*_l}. \tag{3.118}$$

On the other hand, the denominator of equation 3.116 does not account precisely for the latent heat, that is, the enthalpy difference at saturation conditions, but it discriminates two situations:

$$h_v^* - h_l^* = \begin{cases} h_{v,sat} - h_l & \Gamma_k > 0 \\ h_v - h_{l,sat} & \Gamma_k < 0 \end{cases} \tag{3.119}$$

Using this discrimination, the sensible enthalpy required to start a phase at saturation is considered, as well as the sensible enthalpy required to remove a phase at saturation conditions.

The mass transfer rate due to subcooled boiling heat transfer and all the terms required to obtain it are introduced in section 3.12.

3.10 Two-phase wall friction model

In this work it is assumed that the dispersed phase is not in contact with the wall, since it is limited to bubbly/slug flow. Therefore, $F_{wv} = 0$ and this model will be only applied to continuous phase.

For the liquid phase, Churchill model (Churchill, 1977), same model as in single-phase flow, is applied (see section 3.3.1).

However, since this work accounts for situations where subcooled nucleate boiling takes place, a correction term for this model is necessary. The general procedure consists in adding a two-phase multiplier that enhance the calculation. Nonetheless, according to Ferrell and McGee, 1966, in boiling situations the multiplier tends to over-estimate the pressure drop. Moreover, the mass flux is also significantly affected. The deviation of the mass flux due to the multiplier can be found in NRC, 2013.

Therefore, instead of a multiplier term, a correction factor for subcooled nucleate boiling and nucleate boiling flows was implemented.

$$f_{wl,\Phi} = f_{wl}(1 + C_{NB}) \tag{3.120}$$

Equation 3.120 presents the correction made for the liquid friction coefficient. The subscript Φ is used to differentiate this friction factor for the single-phase friction coefficient, f_{wl}. The correction term, C_{NB}, is a function of the hydraulic

diameter, as suggested by Collier and Thome, 1994 and of the void fraction, and it is given by

$$C_{NB} = Min\left\{2,155\left(\frac{d_B}{D_h}\right)\left[\alpha_k(1-\alpha_k)\right]^{0.62}\right\}. \tag{3.121}$$

The equation 3.121 was obtained as an empirical function based on the data given in Ferrell and McGee, 1966. The ratio $\frac{d_B}{D_h}$ is a balance between the surface tension and the drag force. The following model was developed by Levy, 1967

$$\frac{d_B}{D_h} = 0.015\left[\frac{\sigma}{\tau_w D_h}\right]^{1/2}. \tag{3.122}$$

Finally, to complete the model, it should be remarked that the wall shear stress τ_w is calculated without considering the enhancement due to the subcooled nucleation

$$\tau_w = \frac{f_{wl}}{2}\rho_l U_l^2. \tag{3.123}$$

Consequently, with the correction factor for subcooled wall nucleation, the wall fraction is able to represent the pressure drop for a wide range of mass fluxes. It is worthy to mention that, since this correction factor is function of the bubble diameter, when this decreases, it may occur that the correction term would reach large values. In order to avoid unreasonably values, the lower limit of 2 was imposed.

Wallis model

The wall friction model for two-phase flow calculations is based the work of Wallis, 1969. Again, this correlation is function of the Reynolds coefficient and is defined by the following

$$f_w = max\begin{cases}\frac{64}{Re} & \text{if Re} \leq 2000\\ 0.0055 + 0.55\cdot Re^{-1/3} & \text{if Re} > 2000\end{cases} \tag{3.124}$$

The Reynolds number Re is calculated for each phase according to the properties of the fluid. The calculation states as

$$Re = \frac{D_h \rho U}{\mu}. \tag{3.125}$$

3.11 Two-phase wall heat transfer model

In bubbly and slug flows, it is assumed that only liquid phase is in contact to the solid surface. Thus, the heat transferred through the wall to the liquid is calculated by the model presented in subsection 3.3.2, while the the dispersed phase is not in contact with the wall, so $q''_{w,v} = 0$.

However, this is not the only effect that may happen when heat transfer occurs from a solid to the fluid, but nucleation near the wall can also appear. This phenomena is known as subcooled nucleate boiling and it is presented in section 3.12.

Therefore, different phenomena may take place while the fluid is heated, and the wall temperature and void fraction will determine which one is happening. Figure 3.4 shows the logic used by the new solver to select the appropriate wall heat transfer model.

3.12 Subcooled boiling model

The subcooled boiling model accounts for the vapor generation close to the wall. This phenomena occurs when the wall temperature is higher than the saturation temperature of the fluid, so the fraction of fluid in contact to the wall is heated while the bulk fluid remains subcooled. In this cases, when wall temperature is higher that a particular temperature (called *temperature at the onset nucleate boiling*, T_{ONB}), early bubbles may appear close to the wall, at determined "preferred" sites (cavities), and the system needs to be treated as a two-phase flow, even when the bulk is still below saturation. Figure 3.5 shows an illustration of this effect. According to this figure, the void fraction generated is due to subcooled boiling until a location called net vapor generation (NVG), where the vapor generation increases sharply due to the fluid temperature increase.

The vapor generation rate due to this effect can be expressed as

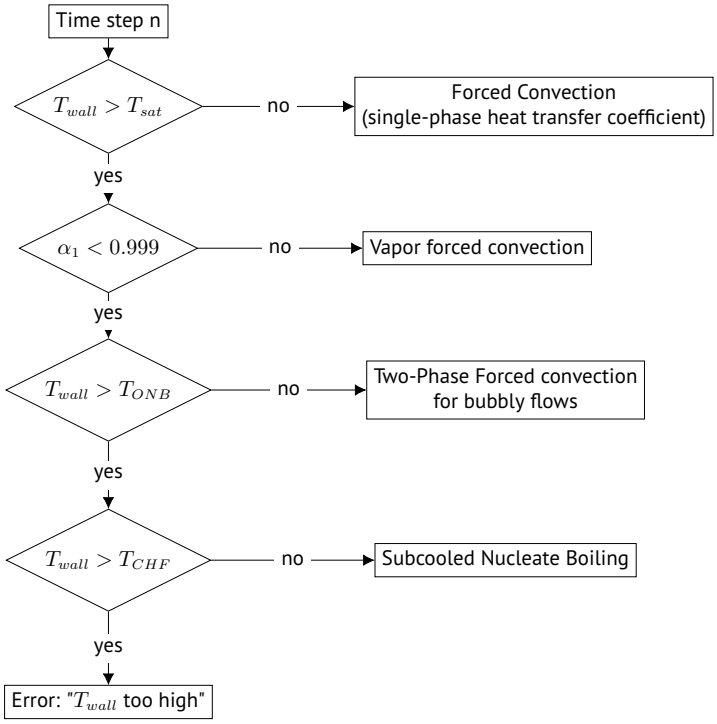


Figure 3.4: Different wall heat transfer models according to wall temperature and void fraction.

$$\Gamma_{sub} = \frac{f_{sub}(q''_{NB} - q''_{conv})A_w}{h_{v,sat} - h_{l,sat}} \tag{3.126}$$

where f_{sub} is the fraction of the heat transfer from the wall to the liquid that generates evaporation near the wall when the bulk liquid temperature is subcooled. A_w is the wall heat transfer surface area per unit volume and is given by

$$A_w = \frac{4}{D_h} \tag{3.127}$$

This fraction of vapor generation rate will be added to the vapor phase while $T_w > T_{sat}$ and $T_l < T_{sat}$. Both conditions must be satisfied.

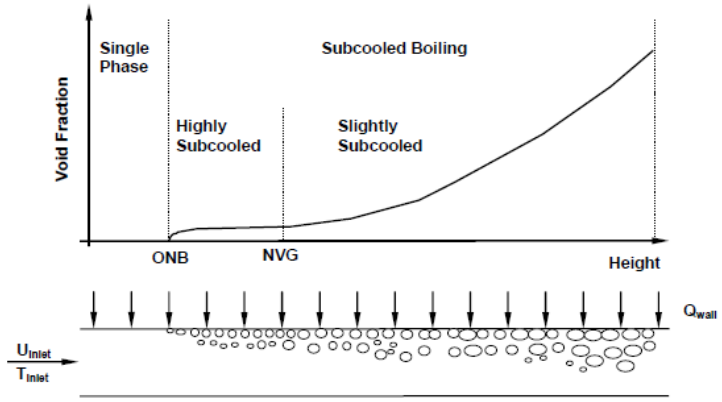


Figure 3.5: Schematic view of the subcooled boiling effect in a heated pipe (Yeoh and Tu, 2009).

The term q''_{conv} is the convective heat flux due to the forced convection regime. It is calculated by means of the wall heat transfer coefficient and the difference of temperatures between wall and liquid. On the other hand, q_{NB} represents the nucleate boiling heat flux. This heat flux accounts for the total heat in the fluid, the one due to convection, the one due to subcooled boiling and any other interaction between phases. Thus, the nucleate boiling heat flux is given by the following:

$$q''_{NB} = q''_{conv} + (q''_{PB} - q''_{BI}) \tag{3.128}$$

So, according to equation 3.128, the total heat flux is given by the convective heat flux, the heat flux given by the pool boiling (q_{PB}), which states for the subcooled boiling, and the heat flux at the point when the subcooled nucleation starts (q_{BI}), which is substracted.

Then, it only remains to present both the correlation to calculate the heat flux that generates wall nucleation (pool boiling model) and the fraction f_{sub}.

Subcooled boiling is considered by using a pool boiling model. In this case, the model known as *Gorenflo model* was chosen, whose heat transfer coefficient is the following

$$\gamma_{PB} = h_O F_P (\frac{q''_{PB}}{q''_0})^n \tag{3.129}$$

A priori, equation 3.129 is implicit, and an iterative method should be required to solve it. However, this equation can become explicit by undertaking some reajustments. In equation 3.129, h_O and q''_0 are constant parameter and, for water, $h_O = 5600.0\,\mathrm{W\,m^{-2}\,K^{-1}}$ and $q''_0 = 20\,000.0\,\mathrm{W\,m^{-2}}$. F_P and *n* are empirical correlations given by

$$F_P = 1.73 p_r^{0.27} + (6.1 + \frac{0.68}{1 - p_r}) p_r^2 \tag{3.130}$$

$$n = 0.9 - 0.3 p_r^{0.15} \tag{3.131}$$

The parameter p_r represents the reduced pressure $p_r = p/p_{crit}$. Rearranging the equation and replacing the pool boiling heat flux for its definition, $q_{PB} = \gamma_{PB}(T_w - T_{sat})$, equation 3.129 can be rewritten as

$$\gamma_{PB} = (h_O F_P / q''^n_0)^{\frac{1}{1-n}} (T_w - T_{sat})^{\frac{1}{1-n}} \tag{3.132}$$

Finally, it is necessary to define the fraction of boiling heat flux that results in subcooled vapor generation, which takes the following form (Lahey Jr, 1978)

$$f_{sub} = Max[0, \frac{(h_l - h_{ld})}{(h_{l,sat} - h_{ld})}]. \tag{3.133}$$

The term h_{ld} represents the enthalpy at which bubbly detachment occurs. This effect is based mainly in local thermal conditions. However, the real implementation was undertaken rather simpler and the fraction f_{sub} in the code is a function of the temperatures instead of enthalpies,

$$f_{sub} = Max[0, \frac{(T_l - T_{ld})}{(T_{sat} - T_{ld})}], \tag{3.134}$$

and T_{ld} is obtained in the solver from the following expression

$$(T_{sat} - T_{ld}) = \frac{q''_{NB} D_h}{PeSt} \tag{3.135}$$

Pe and *St* the Peclet and Stanton numbers, which are two constants based on the fluid thermodynamic properties. In this model, St takes the constant value of 0.0065 and Pe has the lower limit of 7×10^4

So far, the heat transfer rate devoted to boiling generation has been determined and so the subcooled boiling mass transfer. But in order to get a realistic approach, it is necessary to define the point at which the subcooled boiling starts. This point is determined by a temperature, in particular, by the *temperature for the onset nucleate boiling*, T_{ONB}. In this work, the model proposed by Basu, Warrier, and Dhir, 2002 was implemented. This model takes into account the effects of the surface properties of the solid material from which the pipe is made. The surface properties are related by expressing the size of the cavities available at the surface in function of the contact angle between the liquid and the surface.

T_{ONB} is defined as

$$T_{ONB} = tl_{Limit} + \frac{q''_{ONB}}{\gamma_l}. \tag{3.136}$$

Where γ_l is the wall heat transfer coefficient for the corresponding regime (see 3.11).

The term tl_{Limit} is defined as the minimum between the liquid temperature and the saturation temperature, $tl_{Limit} = min(T_{liq}, T_{sat})$.

Q''_{ONB} is the *Onset Nucleate Boiling* heat, which is the heat at which the subcooled boiling starts happening. This term is given as

$$q''_{ONB} = \frac{\gamma_l}{4}(\sqrt{\Delta T_{ONB,sat}} + \sqrt{\Delta T_{ONB,sat} + 4 \cdot \Delta T_{sub}})^2. \tag{3.137}$$

ΔT_{sub} is the difference between liquid and saturation temperatures (subcooling temperature), $\Delta T_{sub} = T_{sat} - T_{liq}$.

$\Delta T_{ONB,sat}$ is the wall superheat necessary for the onset of nucleate boiling when the liquid is at saturation and it is stated as the following

$$\Delta T_{ONB,sat} = \frac{2\gamma_l \sigma T_{sat}}{angF^2 \rho_v h_{fg} \kappa_l}. \tag{3.138}$$

The only unknown in eq. 3.138 is $angF$, which is a correction factor that depends on contact angle. This factor is determined to be

$$angF = 1 - exp(-\phi^3 - 0.5\phi), \tag{3.139}$$

where ϕ is an angle measured in radians that depends on the solid material and, for stainless steel, this value is constant and equal to 0.663225 rad (Basu, Warrier, and Dhir, 2002).

At this point, T_{wall} can be compared to T_{ONB}. In NRC, 2013, this model is compared to two empirical correlations at two different wall heat fluxes. The results show that this model predicts an adjusted temperature for both high and low wall heat fluxes.

CHAPTER 4

METHODOLOGY (I): OPENFOAM, UTILITIES AND EXTERNAL APPLICATIONS

This chapter presents a general description of OpenFOAM CFD code and the main capabilities of this program that were used during this work. This CFD includes a set of standard solvers that are used for the calculation of specific fluid dynamics problems. The discretization is based on the Finite Volume Method and the SIMPLE, PISO and PIMPLE algorithms can be used in order to perform the calculations. It will be seen that PIMPLE algorithm is an evolution and a mixture of SIMPLE an PISO methods. This chapter also describes the different boundary conditions used in the simulations, including the mixed boundary condition, adapted for the convection-conduction case, and the standard thermophysical models that the user can select for the simulations and the performed modifications to use an external application to calculate the fluid thermodynamic properties through the IAPWS-IF97 steam table.

4.1 Introduction to OpenFOAM

This solver has been implemented within the framework of OpenFOAM (Open source Field Operation and Manipulation). This code is a free and open-source CFD software package, also known as a set of libraries that allows to calculate fluid dynamics systems. It was initially developed at the Imperial College during the 80's, being officially released in 2004. It is open-sourced and object-oriented, which means that anyone can add new classes (or models) without making many

changes in the main code. One of its most popular features is that the syntax in which the equations are written is very similar to the mathematical one. For instance, the mass conservation equation for an incompressible fluid

$$\frac{\partial U}{\partial t} + \nabla \cdot (\phi U) - \nabla \cdot (\nu \nabla U) = 0$$

is written in the code in the form shown in figure 4.1. Every term in the code formulation can be easily associated to its corresponding term in the physical equation.

```
solve
(
    fvm::ddt(U)
    + fvm::div(phi, U)
    - fvm::laplacian(nu, U)
);
```

Figure 4.1: Equation representation in OpenFOAM.

The code structure of OpenFOAM is divided into three main parts, as shown in figure 4.2.

The structure presented in figure 4.2 is applied in most CFD codes. First, the simulation starts with the pre-processing tasks. In this initial part, the geometry is defined, as well as the boundary conditions and fluid properties. Then, the model is meshed and the numerical options are selected. Next, the simulation is solved by one of the compiled solvers in OpenFOAM. In this stage, the equations are discretized and solved in each of the cells in which the model has been meshed. Finally, the results are post-processed and visualized by an external tool, which, for a case calculated by OpenFOAM, this tool is usually *Paraview* (Ahrens et al., 2005), although there are others that allows representing OpenFOAM results.

OpenFOAM does not include a graphical user interface (commonly called GUI) to control the different options, but all the information is written in a set of files. The data for every simulation is always stored in an structure that consists of three main folders. The general case structure is shown in figure 4.3.

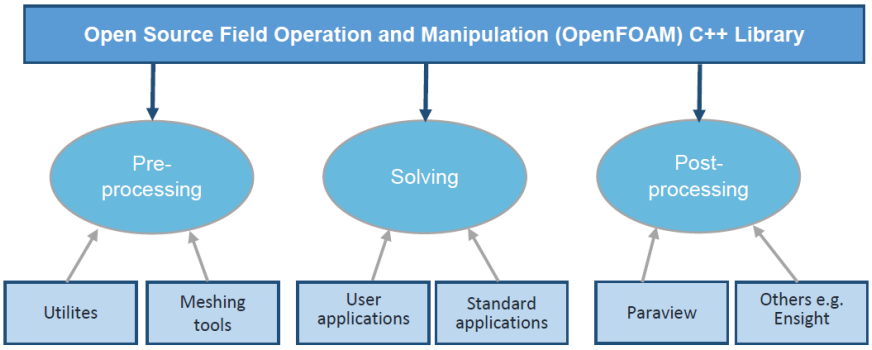


Figure 4.2: OpenFOAM code structure (OpenCFD-Ltd, 2018).

Following the structure shown in figure 4.3, the folder system contains all the information regarding to solver set-up, such as time step, spatial discretization scheme or matrix system solution method. It also includes options to run in parallel or to define samples so that the variables can be analyzed on the fly. The folder constant stores general information about the geometry and fluid properties. The thermophysical model or the turbulence model, if there is any, are defined in files that should be located in this folder. Finally, boundary conditions and initial values are stored in folder 0, as the initial time directory. When the case is running, folders for several time steps are created, where the values for that time step of different variables are written. The interval of time between the written folders is defined by the user in the file ControlDict, located in system. Deep information about the case set-up can be found in OpenCFD-Ltd, 2018 and in Marić, Hopken, and Mooney, 2014.

The structure presented in figure 4.3 does not change from one version to the following, but new files for novel features or capabilities may appear with future versions. The particular version used during this work is OpenFOAM+ v1712, which was released in January of 2018.

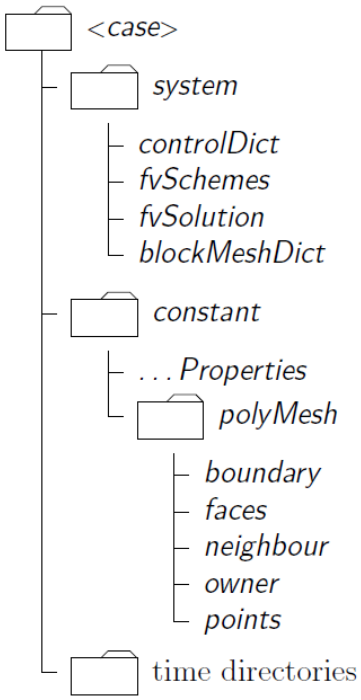


Figure 4.3: OpenFOAM case general structure (OpenCFD-Ltd, 2018)

4.1.1 Solvers

Trying to benefit from the libraries included in OpenFOAM, the solver presented in this work departs from an standard solver from OpenFOAM. First part of the work, when the solver was only single-phase, the base solver was buoyantPimpleFoam, which is defined as a *transient solver for buoyant, turbulent flow of compressible fluids for ventilation and heat-transfer* (OpenCFD-Ltd, 2018). Initially, it was considered as incompressible, but it evolved considering density, since this is fundamental in order to take into account buoyancy effects.

The second part, focused on two-phase flows was based on a different solver, called twoPhaseEulerFoam. This solver is devoted to calculate two-phase compressible systems, where one phase is dispersed in the other one. It uses an Euler-Euler model, which means that each phase is treated as a continuum and represented by averaged conservation equations. It also considers heat transfer between phases. However, this built-in solver does not consider some of the re-

quirements defined in this work, so these were implemented in order to adapt the solver to the specifications. One of the main issues is that this standard solver does not simulate the interfacial mass transfer. The inclusion of the mass transfer phenomena is explained in section 5.4.3. It does not considers the regime flow map either and the thermophysical models are basic, not being able to access the steam tables. Furthermore, it will be modified so that this solver can work with an external solid domain.

This final one-dimensional two-phase flow solver is meant to be the final solver. Therefore, and since the objective is having only one solver to simulate any kind of flow, it was last tested to check its capability to simulate single-phase flows. All the results presented in this work were simulated with this last solver.

4.2 Finite Volume Method

The Finite Volume Method (FVM) is the most extended method for equation discretization in CFD. It is very popular because of its high flexibility for discretization. One of its main strengths is its conservation of every property, and the characteristics of every discretized term. Therefore, there is no need of simplifications of the physics or principles that are going to be modeled. Furthermore, adjustments between the system of equations and system of coordinates is not necessary, since the equation discretization is directly applied within the volume to be solved (Moukalled, Mangani, Darwish, et al., 2016). In this section, the basis of the FVM and its numerics are presented.

The particular approach that characterizes this method is the **integration of the control volume**. To apply the integration, first the problem domain is divided into small control volumes (CV), defining the grid. Figure 4.4 shows an example of a 2D geometry divided into different CV. The grid will be closed by a set of boundary conditions, defined in section 4.3. The values in each control volume are calculated at the centroids of each CV, not at the vertex.

Once the grid is defined, the integration of each CV is applied. Lets consider the conservation equation of a general property ϕ in its steady-state form,

$$\underbrace{\nabla \cdot (\rho v \phi)}_{Convective\ term} = \underbrace{\nabla \cdot (\Gamma \nabla \phi)}_{Diffusive\ term} + \underbrace{S_\phi}_{Source\ term} \tag{4.1}$$

Integrating the equation for a specific CV, one gets

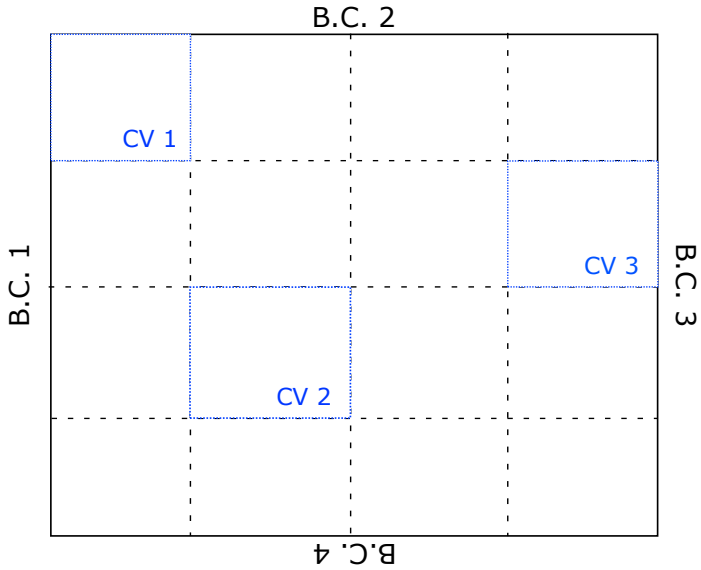


Figure 4.4: 2D Mesh.

$$\int_{CV} \nabla \cdot (\rho v \phi) dV = \int_{CV} \nabla \cdot (\Gamma \nabla \phi) dV + \int_{CV} S_\phi dV \tag{4.2}$$

And applying Gauss divergence theorem over equation 4.2, the volumetric integrals are transformed into surface integrals, as shown by the following

$$\int_{S} \nabla \cdot (\rho v \phi) dS = \int_{S} \nabla \cdot (\Gamma \nabla \phi) dS + \int_{S} S_\phi dS \tag{4.3}$$

Therefore, at this point the FVM consist in the integration over all the surfaces of the CV. The integration over the volume surfaces can be replaced by the summation of the integral of each surface of the CV, rewriting equation 4.3 as

$$\sum_{S\,CV} \left(\int_{S} \nabla \cdot (\rho v \phi) dS \right) = \sum_{S\,CV} \left(\int_{S} \nabla \cdot (\Gamma \nabla \phi) dS \right) + \sum_{S\,CV} \left(\int_{S} S_\phi dS \right) \tag{4.4}$$

Therefore, at this point, the terms are evaluated at the faces of each CV. When summing all faces of the volume, internal faces are canceled out and one can see that global conservation is satisfied. It is important to point out that control

volumes must not overlap among them to keep conservation (Ferziger and Perić, 2002).

In order to solve the system, it remains to calculate the surface integrals. Getting the exact value of the integral would require to know the value of the flux at every point of the surface, but since this is not possible, the alternative is to know the value only at one point or a few points of the surface. The method to solve the integrals at these points are explained in Moukalled, Mangani, Darwish, et al., 2016, Ferziger and Perić, 2002 or Versteeg and Malalasekera, 2007.

The discretized system needs to satisfy a group of properties to ensure that the result approaches the exact solution. The proper behavior should show that the result of the system tends to the exact solution as the size of the control volume tends to zero. The different properties that should be accomplished are listed in the following enumeration.

- Conservation. This property has been already mentioned and it is demonstrated that FVM satisfies conservation.
- Accuracy. The accuracy measures the proximity of the calculated solution to the exact one. The closest the calculated and the exact results are, the highest the accuracy is. The error for FVM is defined as $O|x - x_f|^2$, which means that the error is reduced by a power of 2, so if the size is reduced to the half, the error is reduced to a fourth. This error is called a second order of accuracy.
- Convergence. This property evaluates the difference in the results for a two consecutive time steps. The value of convergence is generally a tolerance defined by the user, and when the difference between the results of the two time steps is below the defined tolerance, the problem is considered converged.
- Consistency. The problem is consistent when the results approximate to the exact solution as the time step or the CV size tends to zero (so the discretization error tends to zero). This is especially important in turbulent cases or multiphase problems. But it also may lead to excessive computational resources.
- Stability. The stability behavior of a system of equations represents the capability of the system to be solved with different boundary conditions or even different numerical schemes. The more adaptability against different boundary conditions, the more stability presents. This property is less related to the FVM and more to the system of equations itself.

- Economy. This property is directly related to the computational time that requires the system. The smaller the cell is, the higher the total number of cells, so the matrix system will increase and the computational cost will rise. It is necessary to find the balance between the accuracy and consistency of the system on the one side, and the required time to solve the system on the other.
- Transpotiveness. Transportiveness means how influences the property on the closest region in the domain. A diffusive property will keep its influence within the radius of the cell, affecting all directions in a similar way. A property dominated by the convective influence may adapt an hyperbolic behavior, influencing the region upwards in the direction of the flow, and having a very weak influence in the region backwards the centroid of the cell (in the direction of the fluid). Not considering this effect in the evolution of the system might lead to unstable results.
- Boundedness. The system must be bounded between those values that are physically realistic. Conservation does not ensure this by itself. This can be achieved by selecting the appropriate discretization scheme and correctly linearizing the equations. Therefore, the user should be able to evaluate this property by controlling the different numerical schemes.

Thus far, only cell-centered FVM has been considered. This collocation is the most extended way of using the FVM, it is the general discretization method in OpenFOAM and the selected one for this work. However, FVM can also be used with a vertex-centered mesh, so, to complete the introduction about the FVM, a brief description of each type of mesh and its similarities and differences is presented.

4.2.1 Vertex-centered arrangement

In a vertex-centered grid, the values of the properties are stored at the vertices of the cells. In this structure, a different *virtual grid* is needed to connect the different vertices. The new grid may not be formed by regular cells, and in order to define these new grid, one should use shape functions. Therefore, the mesh should be based on a set of elements types for which shape functions are predefined.

The vertex-centered arrangement is a low order accurate method of element-based integration, since the centroids of the virtual grid may not be at the midpoint between vertex, leading to some discrepancies when this elements-based values are calculated.

One of its advantages is the accurate calculation of fluxes, but on the other hand, it may generate a larger matrix size, requiring higher computational memory. Furthermore, it has problems in meshes with sharp edges.

4.2.2 Cell-centered arrangement

In a cell-centered arrangement, the values are stored at the centroids of the cells. Applying this discretization, the grid is always the same as the discretized mesh, without the need of shape functions and obtaining a second order of accuracy. Therefore, all values are calculated at cell and face centroids, giving a discretization error equal to $O|\Delta x^2|$.

There are two main disadvantages in this arrangement: First, the resolution of diffusion in non-orthogonal grids is not accurate. This is because the average value at the midpoint between cell centers will never match the value at the centroid of the face. The second is related to the smoothness of the mesh, which means that, in sharp edges can appear discontinuities that lead to an increase in the error.

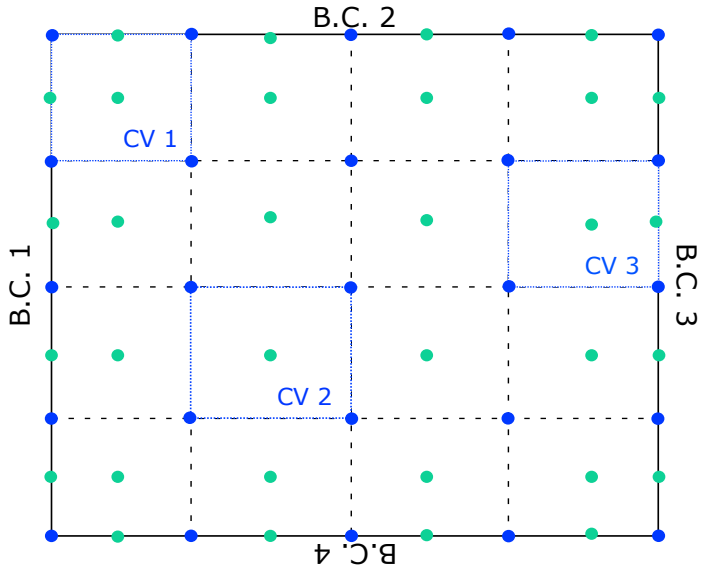


Figure 4.5: 2D Mesh with vertex-centered (blue dots) and cell-centered (green dots) arrangements.

Figure 4.5 shows the location of the values for each type of discretization. In this figure, B.C. accounts for the boundary conditions.

4.3 Boundary conditions

Boundary conditions are a part of the information of the problem so that this can be solved. These are used to define the patches, which are the limits of the geometry in each side. The boundary cell, which is a cell with one or more boundary faces, will store the discrete value defined by the boundary condition at the centroid of the boundary face and of the boundary cell.

OpenFOAM provides a wide variety of boundary conditions for every type of patch. Hereafter, only those boundaries used in this work are going to be introduced. The boundary conditions (from now on, B.C.) are generally classified in function of the type of patch. Since this work is aimed at one-dimensional simulations, only three types of patches are used: inlet, outlet and wall.

4.3.1 Inlet boundary conditions

The OpenFOAM boundary conditions used in this work for inlet patch are the following:

- fixedValue. This is the most extended inlet boundary condition. It is a Dirichlet condition, which defines a constant value at the patch for any defined variable ϕ of the system. This condition is often used for temperature or void fraction.
- zeroGradient. This boundary is a particular case of Von Neumann boundary condition, which defines the flux at the inlet (in this case) surface. In zeroGradient, the flux of ϕ at the boundary is zero. This B.C. is mainly used for pressure.
- flowRateInletVelocity. With this condition, the massflow is fixed at the inlet, instead of the velocity. It requires that the inlet density is also specified.

FixedValue and ZeroGradient are the most used boundary conditions and they can be used for any type of patch, as it will be seen up next.

4.3.2 Outlet boundary conditions

In the case of outlet patch, among the OpenFOAM most common B.C. are also found FixedValue and ZeroGradient. The former is used to define pressure while the latter allows setting void fraction or fluid temperature B.C. In addition to these, other B.C. commonly used in this work are:

- InletOutlet. This is a type derived from ZeroGradient. In this case, it depends on the sense of the flux. When the flux is positive (according to the coordinate system), the B.C. works as Von Neumann, imposing the flux defined by the user. When the flux is negative, the B.C. operates as ZeroGradient.

4.3.3 Wall boundary conditions

In a wall boundary patch is very common to use FixedValue and ZeroGradient. They are used to define velocity (zero fixed value), pressure or void fraction. But there are also typical alternatives, such as

- ExternalWallHeatFluxTemperature. This boundary condition is used to apply an external heat flux to the wall of the domain. It can be defined as a total power, as a heat flux or as a combination of the external wall heat transfer coefficient and the ambient temperature. Depending on the mode given as input, it performs different calculations (if necessary) to get the heat flux, which is the final parameter used by the B.C.

- Mixed. A mixed B.C. consists in a condition that applies two different types of conditions. It is used in situations where there is convection on one side while the other side applies a fixed value. Therefore, it takes into account the two sides of the boundary. In order to use the standard version of this B.C., the user must define in the input a reference value and the gradient applied. Having this, the code performs the following calculation

$$x_p = wx'_p + (1 - w)\left(x_c + \frac{\nabla_\perp x}{\Delta}\right), \tag{4.5}$$

  where x_p is the patch value, which will depend on the reference value (x'_p), the gradient and the value of the adjacent cell (x_c). The term w is a weighting value that needs to be defined too.

The mixed B.C. is the chosen one to calculate the balance between the fluid convective and the solid conduction heat. However, the standard boundary condi-

tion is not able to take the proper values from the different domains to apply the heat balance, because they are defined to work with only one domain. Therefore, it was modified in order to be able to take the values of each domain every time step. Thus, the development performed for a fluid-solid interface and the final equivalence of the terms with equation 4.5 is presented here.

Figure 4.6 shows two meshes connected by an interface. On the left-hand side, a fluid (water) is running, whereas the other side of the interface is a solid structure which is being heated.

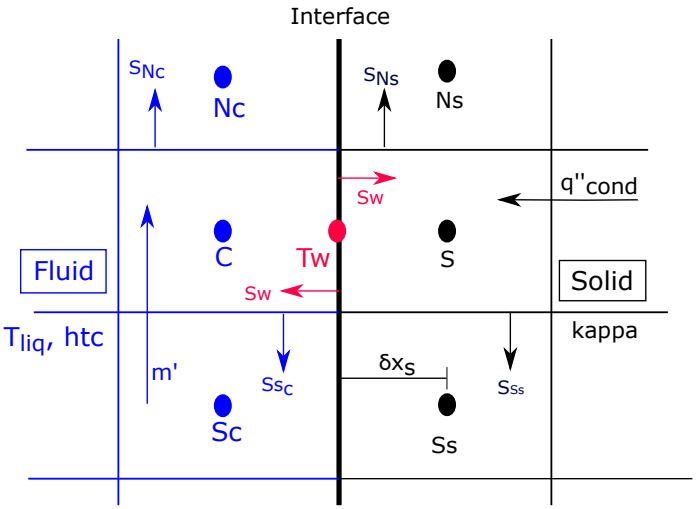


Figure 4.6: Fluid-Solid mesh connection.

The value at the interface is given by the following general balance

$$-\Gamma_b^{\phi} S_b \left(\frac{\phi_b - \phi_C}{\delta x_b} \right) = -h_{\infty}(\phi_{\infty} - \phi_b) S_b \tag{4.6}$$

where, rewriting the equation to calculate the value at the interface, equation 4.6 is now

$$\phi_b = \frac{h_{\infty}\phi_{\infty} + (\Gamma_b^{\phi}/\delta x_b)\phi_C}{h_{\infty} + (\Gamma_b^{\phi}/\delta x_b)}. \tag{4.7}$$

Adapting this condition to the study case, the left-hand side of Eq. 4.6 represents the solid structure and the right-hand side stands for the fluid convection. In this case, a heat balance is applied. Then, in the left-hand side, Γ can be replaced by the solid conductivity (κ_S), δx_b is the distance from the interface to the solid cell center and ϕ_C is the value of temperature of the solid cell (T_S). On the fluid side, h_∞ accounts for the wall heat transfer coefficient (γ_w) and ϕ_∞ will be replaced by the temperature at the bulk, T_l. S_b is the interfacial transverse surface.

Therefore, equation 4.7 for temperature calculation is given by

$$T_w = \frac{\gamma_w T_l + (\kappa_S/\delta x_s) T_S}{\gamma_w + (\kappa_S/\delta x_s)}. \tag{4.8}$$

In equation 4.8, T_l can be directly applied for both single and two-phase flows, since in bubbly regime, it is assumed that only liquid is in contact with the wall.

Once the development was applied and the final calculation defined, it is possible to carry out an equivalence between terms in equation 4.8 and in equation 4.5. Table 4.1 shows the comparison of terms.

Table 4.1: Equivalence of terms between equations 4.5 and 4.8.

Equation 4.5	Equation 4.8
x_p	T_w
x'_p	T_l
$\nabla_\perp x$	0
w	$1/(1+\frac{\kappa_S}{\gamma_w \delta x_s})$

It is worth noting that in simulations where void bubbles are concentrating in the wall due to subcooled boiling, the heat transfer coefficient term in equation 4.6, γ should account for the convective heat and also for the fraction of subcooled heat that is used to evaporation. Being γ_{sub} the heat transfer coefficient that accounts for the evaporation heat, then the term w will change to $w = 1/(1+\frac{\kappa_S}{(\gamma_w+\gamma_{sub})\delta x_s})$. The heat transfer coefficient due to subcooled heat is calculated as

$$\gamma_{sub} = \frac{q''_{SB}}{T_w - T_{l,sat}}. \tag{4.9}$$

In OpenFOAM, the parameters involved in the mixed B.C. calculation are input defined, so they remain constant over time. In the present work, this boundary

condition has been modified so that the different variables are updated every time step.

4.4 Solution algorithms: SIMPLE, PISO and PIMPLE

The usual method to solve Navier-Stokes equations consists in decoupling velocity and pressure in the system. In order to decouple the calculation of these two variables, a predictor-corrector procedure is used. This method is known as *Segregated method*.

There are alternatives to solve the fully coupled system of equations at once, which is known as *block-coupled solution*. This method gets better convergence rates, but the matrix system is larger than in segregated methods, requiring high memory investment.

In this work, only the first method is presented, which is the one used for all the simulations.

4.4.1 SIMPLE Algorithm

The algorithm SIMPLE (*Semi-Implicit Method for Pressure-Linked Equations*) is used for solving the conservation equations in steady-state situations. This method is based on the calculation of a *predicted* velocity in order to solved the pressure and then correct the previous velocity so that continuity is satisfied. The general procedure that follows the SIMPLE algorithm is:

1. Set the boundary conditions.
2. Set an initial guess pressure p'.
3. Solve the discretized momentum equation to obtain predicted velocities u', v' and w'.
4. Calculate the mass fluxes at the cell faces.
5. Calculate the new pressure p*.
6. Correct the pressure p = p'+ p*.
7. Correct the velocities u, v and w using the new pressure p*.

When the difference of the values of u, v, w and p of two consecutive time steps are below the defined convergence value, the algorithm is finished. Figure 4.7 shows the run process of SIMPLE algorithm in OpenFOAM.

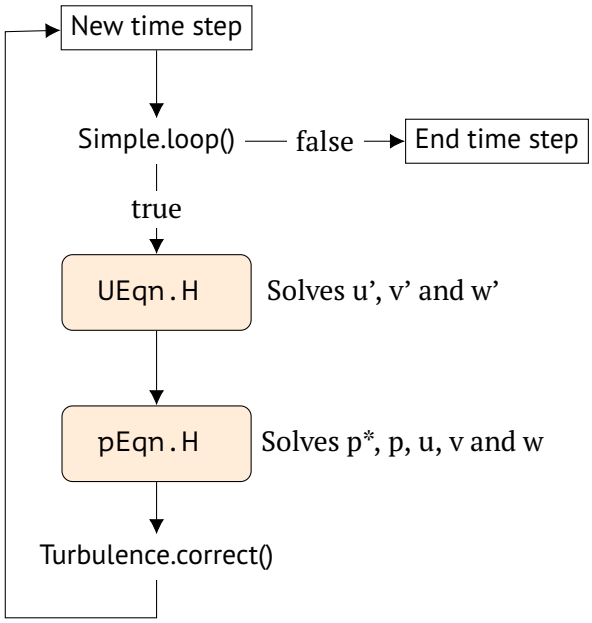


Figure 4.7: SIMPLE loop structure (Holzinger, 2016).

4.4.2 PISO Algorithm

In most solvers, the transient simulations are solved with the algorithm called PISO (*Pressure-Implicit with Splitting of Operators*). In this algorithm, the velocity and pressure corrections are calculated during determined number of iterations defined at the input. Its iterative process can be described by the following (*openfoamwiki.net* 2013)

1. Set the boundary conditions.
2. Solve the discretized momentum equation to calculate predicted velocity field.
3. Compute the mass fluxes at the cells faces.
4. Solve the pressure equation.

5. Correct the mass fluxes at the cell faces.
6. Determine the correct velocities using the new pressure field.
7. Update the boundary conditions.
8. Repeat from 3 for the prescribed number of times (the pressure can be calculated as many times as decided by the user).
9. Increase the time step and start from 1.

The complete sequence used by OpenFOAM in order to apply this algorithm is shown in figure 4.8.

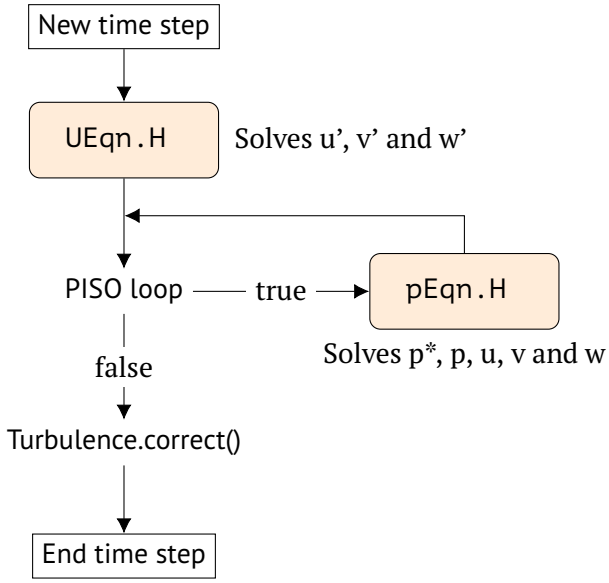


Figure 4.8: PISO loop structure (Holzinger, 2016).

4.4.3 PIMPLE Algorithm

The solution algorithm used in this solver is PIMPLE algorithm. This solution method is defined in OpenFOAM© documentation as *Large time-step transient solver*. This algorithm is basically divided into two loops, one outer loop that involves the second loop, which are called pimple.loop() and pimple.correct() respectively. Figure 4.9 shows the structure of this algorithm adapted for the new solver.

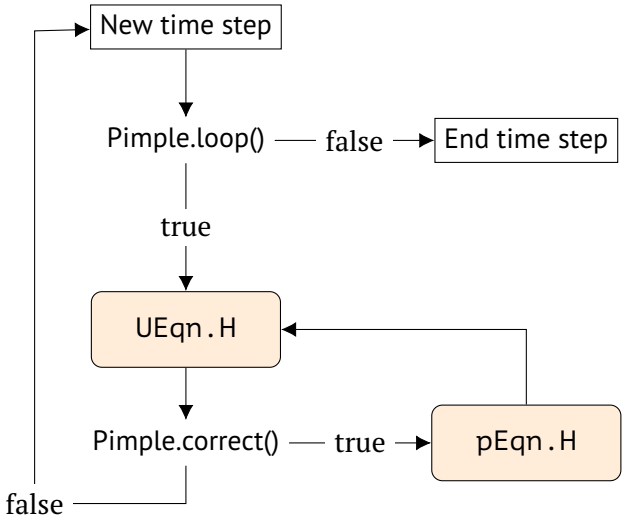


Figure 4.9: PIMPLE loop structure (Holzinger, 2016).

The loop pimple.loop() is used when a convergence is required every time step. If this option is not used, then this algorithm works as a usual transient solver. The loop pimple.correct() allows the user to define a number of iterations in which the correction of pressure and velocity is computed in a time step. The PIMPLE structure provides the advantage of working with large time steps without being limited by the Courant number and otherwise strengthen the convergence. This algorithm is specially useful in complex geometries with different models that may cause stiff challenges, such a large models with small cell sizes and rapid velocity changes.

The work presented in this paper uses pimple.loop() only once every time step, since the geometry is very simple. Therefore the solver works as typical transient, which is equivalent to perform the calculations using the PISO methodology (Issa, 1986).

4.5 Short introduction about OpenFOAM programming

Once the general introduction for CFD codes has been presented, the typical algorithms and boundary conditions introduced, and the discretization method explained, the reader may know the scope in which this work was developed. However, in order to follow properly the different developments performed in the code, it is necessary to bring in next concepts about OpenFOAM.

4.5.1 General file distribution in OpenFOAM

The code is distributed into the following folders:

- Src: Location of all the core libraries. They are alphabetically sorted.
- Applications: Divided in *solvers*, which contains the different built-in solvers and *utilities*, which are extra functions that are commonly used for the different solvers, generally for pre- and post- processing tasks.

The solvers encompass different functions that are focused on a particular fluid dynamic problem. For instance, simpleFoam solver contains the necessary routines to solve stationary incompressible fluids. They are all executable, as well as the utilities. Files in src cannot be executable by their own, they need to be called by a solver or utility.

4.5.2 Using Classes and Objects

OpenFOAM is developed in C++ language, so one of its most powerful features is its *object oriented* capability. The definition of *class* and *object* is presented in this subsection, in addition to the properties of each term.

The concept of *class* can be defined as an abstract idea. This idea should have a group of properties and should be able to modify these properties. In programming language, the *class* is considered as a data structure, which is compose of attributes (*member data*) and functions (*member functions*). Both variables and functions are *objects* of the class.

Let us apply the previous concept to an example: An abstract idea could be *Time*. Using the concept of *time* in a solver, it is desirable that this concept would involve the control of the time along the simulation, including when to start and end and when to write results, as well as the discretization of the time step, as minimum. It should also be able to differentiate between the computational and

the real time, and the time step value and the value of the size of the time step. Table 4.2 shows some of the attributes here mentioned which already exists in the class *Time* of OpenFOAM.

Table 4.2: Some basic objects of class *Time*

Class Time			
member data		member functions	
Object	Description	Object	Description
startTime	Initial time	startTime()	Return initial time
endTime	Final time	endTime()	Return final time
deltaT	Time step value	read()	Read time controls from input
		deltaT()	Return current time step

An object is not necessarily a function, but it is more likely a variable which has multiple properties, also called attributes. These properties can be modified. This object will share attributes (member data) with the rest of objects of the same class. The attributes can be private (only seen by the functions of **this** class), protected (shared with particular classes) or public (shared with all classes). The complete structure of a class must include the construction the objects (how are they constructed: which type of structure defines the object, which inputs requires,...), their destruction (once the object has been used, this memory is released by destructing the object), and the function to manipulate this object.

There is one last idea that needs to be clarify, which is the concept of *inheritance*. A class can inherit the objects and attributes of a different class, by including it as a *parent*. In order to understand this idea, the already known class *Time* is used again. Figure 4.10 presents the real assembling of the class *Time*. Here one can see that *Time* inherits the attributes and objects of five different classes (*parents*), where each one of these is focused in a different aspect of *Time*.

For instance, class *clock* aims at formatting the system clock according to the required use. On the other hand, class *TimeState* focuses on the time value from the time-stepping point of view: its adaptability, the frequency for writing the output (if it is dependent on the number of time steps), etc. Therefore, *Time* involves their own objects plus the attributes and objects of its parents, as long

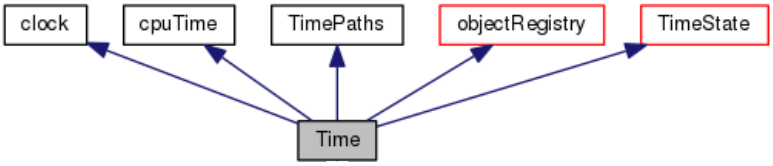


Figure 4.10: Inheritance of class Time (OpenCFD-Ltd, 2018).

as these attributes are not private. The use of multiple parents to create a new class allows the combination of different features.

In order to create a clear image of the described concepts, the schematic shown in figure 4.11 was created, taking a general class and its structure.

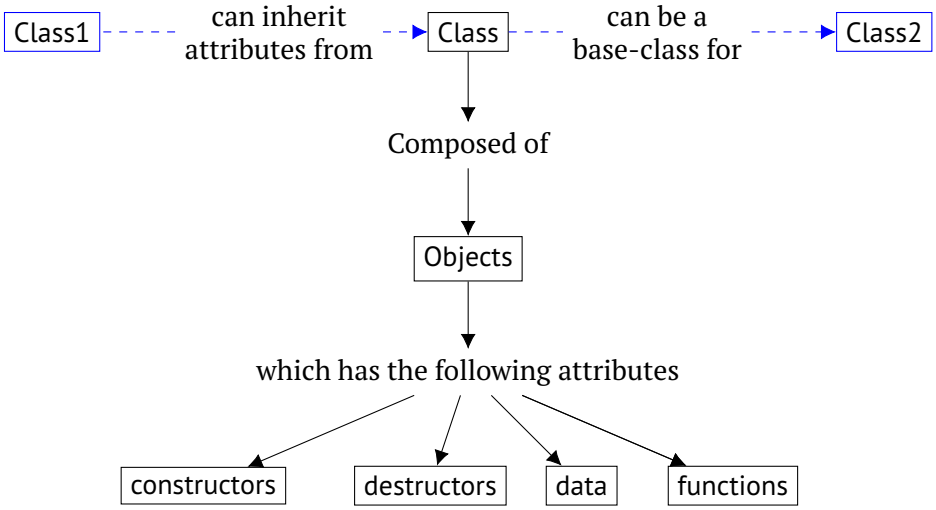


Figure 4.11: Class structure.

4.5.3 Numerical Operators

Solvers in OpenFOAM applies different numerical operations in order to discretize, interpolate or calculate the equation terms. The properties of each numerical operator can be different according to its use, and making the correct choice requires of some expertise in CFD and a deep knowledge of the equations. These properties are assembled as functions templates which depends on

the type of tensor which are going to set. This capability allows to write models in a way so similar to the mathematical formulation. Among these operators, one can highlight the *divergence* ($\nabla\cdot$), *gradient* (∇) or *laplacian* ($\nabla \cdot \nabla$).

These operators are divided into two main groups, these are:

- fvm, which are implicit numerical operators. These operators are used to build coefficient matrices from the equation terms discretization.
- fvc, which are explicit numerical operators. From these operators one gets a new result.

Some of these operators can be both implicit and explicit, but the way of working is different for each case. Therefore, in order to avoid confusions, the operators are written using the group to which it belongs as prefix, e.g. fvm::div would represent the divergence implicit operator.

Only those which are going to be mentioned at some point in this work are presented in table 4.3.

Table 4.3: Numerical operators mentioned in this work.

Operator	Description
div	Divergence
grad	Gradient
laplacian	Laplacian (only implicit)
ddt	Temporal derivative
Sp	Implicit source term
Su	Explicit source term
SuSp	Implicit/Explicit source term

The operator SuSp is used when the value of the source field can change its sign. If the field is implicit, a positive value of the field would be subtracted from the diagonal, leading to an ill-conditioned matrix. Therefore, this operator treats the field as implicit or explicit depending on its sign.

4.6 Conjugate Heat Transfer

Besides the simulations of single- and two-phase flows, the new one-dimensional solver was required to be able to simulate the radial temperature distribution within a solid in contact with the fluid in those cases where heat transfer plays an important role. There are components that needs to know the solid behavior to get the whole perspective of the heat transfer phenomena. This is the case of heat exchangers or fuel rods in nuclear field, but also in chemical field, heat sinks or in combustion engines simulations one can find other examples.

Even though the wall temperature can be calculated by using a wall heat flux model as boundary condition, such as the models proposed by Steiner, Kobor, and Gebhard, 2005 or Kurul and Podowski, 1990, without the necessity of simulating a solid, in these cases the feedback between domains is lost. Furthermore, the temperature and heat flux distributions on the boundary depend not only on the thermal properties and the flow characteristics of the fluid, but also on the properties of the wall (Barozzi and Pagliarini, 1985), and in most physically realistic situations, however, the boundary conditions at the interface are not known a priori, but depend on the coupled conduction-convection mechanism.

Therefore, an extra domain was included in the solver to consider the conduction phenomena in a wall. Thus, if a solid exists, its geometry and mesh are added to the fluid model as an independent structure which exchanges information with the fluid. The solid domain is also solved for one dimension, but in this case the heat conduction is calculated in the direction perpendicular to the surface.

Hence, the solid domain is considered by using an extra mesh that allows calculating the conjugate heat transfer in the solver (CHT). This method consists in coupling the solid conduction equation to the fluid energy equation. This feature gives the solver powerful capabilities to simulate the heat transfer, providing a complete perspective of the fluid-solid interaction, so a better understanding of the process can be acknowledged. Different authors applied the conjugate heat transfer to the two-phase flow, like Welch and Rachidi, 2002, which applied the conjugate heat transfer for film boilings in horizontal pipes, obtaining good agreements. Also the authors in Yapici and Albayrak, 2004 calculated the temperature and stress ratio distributions inside the pipe wall for uniform and non-uniform heat flux boundary condition. This method has also been applied in PWR pressurizer surgeline Jo and Kang, 2010, to realistically simulate the thermally stratified flow in the pipe, and in Ates, Darici, and Bilir, 2010 a parametric study was done to demonstrate the strong influence effects of four defining parameters namely, wall thickness ratio, wall-to-fluid thermal conductivity ratio,

wall-to-fluid thermal diffusivity ratio and the Peclet number in the final wall temperature.

The conjugate heat transfer (CHT) allows the simulation of the heat transfer between fluids and solids. Therefore, there is an interface between both regions that allows the thermal energy exchange. This heat transfer happens when there are no other source terms or radiation heat and only conduction and convection are left. The heat exchange takes place through an interface that links both fluid and solid domains. This section is directly related to the mixed boundary condition described in section 4.3.

The CHT approach is not a new strategy, but it has been studied for several years, starting with (Perelman, 1961). However, it became popular recently due to the improvements in computer technology that started to allow performing calculations in a reasonably short time.

OpenFOAM includes a built-in solver for CHT problems, called chtMultiRegionFoam. This solver is a mixture of a compressible solver (also built-in) and a solver to simulate the diffusion equation. It can work with as many different domains as the problem needs.

ChtMultiRegionFoam departs from an only one mesh or system, which is divided into as many different regions as domains exist in the problem. First, the whole geometry is defined, and during this process, the limits of each different region are determined. Once the mesh of the whole system is created, this is split into regions. For example, if the system has two domains, a fluid and a solid, the mesh will be one limited by regions, where each region has its own boundary conditions. Those faces that are shared by two different regions, and they do not limit the system, they work as any other internal face.

This can be seen in figure 4.12, where the interface, which would be a boundary condition for each domain, is turned into an internal face.

Regarding to the distribution of the case, and the information of each region, figure 4.13 shows a case distribution for a system with two domains, the liquid and the solid. In this solver, the information of the case is divided into the different regions of the problem, following the main structure for setting a case.

In figure 4.13 two new files are shown. First, the file CellToRegion defines all the patches of each region. This file is created automatically when the different regions of the mesh are established. On the other hand, the file RegionProperties contains which region corresponds to each domain (fluid or solid). In the case

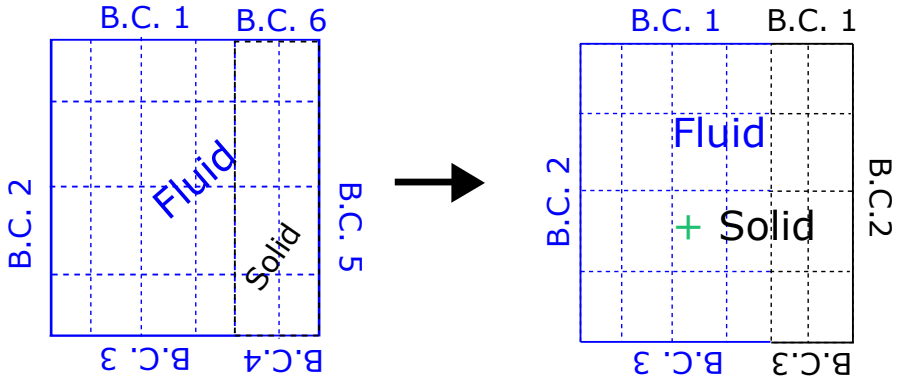


Figure 4.12: In chtMultiRegionFoam, the mesh is composed of different domains. Left-hand side mesh: Initial definition. Right-hand side: Final mesh (already split into regions).

shown in 4.12, there are two regions and each one corresponds to one domain. This file is user-defined and it is required to start the simulation.

Once the problem has been set up, it is time to solve it. The sequence followed by this solver consists in first solving fluid equations and then solving the solid conduction equation sequentially. In order to couple both domains and find out the converged solution, first, the fluid energy equation is solved with a value at the boundary which is equal to the solid wall temperature calculated at the previous time step. Next, the solid conduction is solved by fixing the heat flux at the boundary as the convective heat flux obtained when the fluid energy equation is solved. Figure 4.14 shows a flow chart with the solving sequence. The solver allows to define more than one iteration in each time step to ensure energy balance. This is done by energyCoupling() option.

Nonetheless, this solver can only be applied, in domains with, at least, two dimensions, the axial and radial. Therefore, this methodology was modified for the work presented here.

The solver developed in this work follows the CHT philosophy in solving structure matters. However, the solid is treated as a second mesh which is linked to the fluid by a contact surface. Comparing to the former structure, in this case these both regions are not merged, but they only share a surface. Therefore, in the new 1D solver it was created a routine that builds the solid mesh, but this remains as an independent mesh from fluid and they remain in this way all the simulation. Taking this option, the solid is left as an optional (not all simulations requires solid) and this second domain can also be simulated in one dimension.

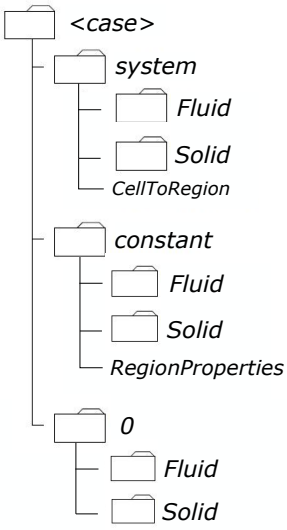


Figure 4.13: Directories and new files required in a chtMultiRegionFoam case.

Figure 4.15 shows the new situation, in terms of figure 4.12. The development of the routine that builds the solid mesh is explained in section 5.3.2.

The shared surface between meshes will be treated as a mixed boundary condition (see section 4.7). Leaving the solid as a separate geometry, it allows the user to give different properties to the each mesh, such as the number of dimensions to be solved or the discretization method. The new solver has to detect the solid mesh and be able to transfer information between both fluid and solid meshes. This information flow is shown in chapter 5.

4.7 Thermophysical models

The standard thermophysical models in OpenFOAM® are used for defining how the energy, heat and physical properties of the fluid are calculated. These variables are determined using models that are relations of a Pressure-Temperature equation system. The three main groups of thermophysical models are:

- *Compressibility-based thermophysical models*, which are based on the compressibility ψ to calculate other thermodynamic parameters.

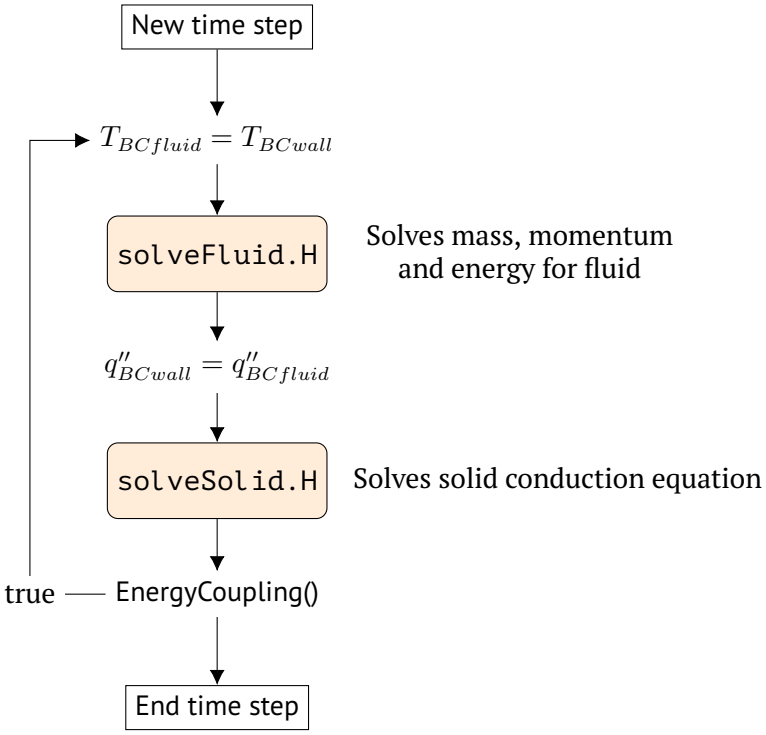


Figure 4.14: chtMultiRegionFoam loop structure.

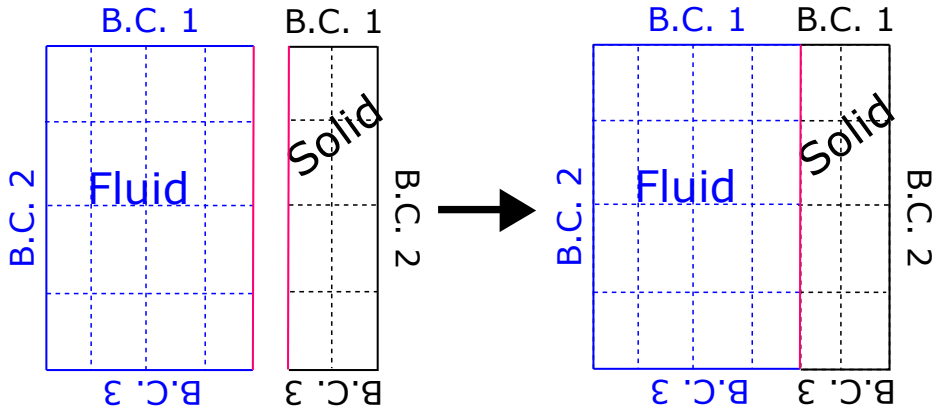


Figure 4.15: Mesh structure in 1D solver. Red vertical line is the shared surface.

- *Density-based thermophysical models*, those which are based on density to compute the rest of thermodynamic properties. Only this group is relevant in heat transfer simulations.
- *Reaction thermophysical models*, used when chemical reactions are involved in the simulations. This is the widest group, since the change in pressure and temperature during the reactions influences very significantly the calculations.

Once one of these three main groups has been chosen, a thermodynamic model can be also selected to calculate the rest of fluid properties. According to the type of variables, there are different models: There are models for mixtures (used for chemical reactions, in case there are more than one reactive), for transport properties, for the calculation of enthalpy considering thermodynamic properties, like C_p; for calculating density, for species (where the substance is defined by the number of moles and molecular weight) and for defining the type of enthalpy which is going to be used. This groups are always defined in a file called thermodynamicalProperties, and the definition can be seen in the portion of the file shown in figure 4.16.

```
thermoType
{
    type            heRhoThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConstant;
    equationOfState  rhoConst;
    specie          specie;
    energy          sensibleEnthalpy;
}
```

Figure 4.16: Definition of the thermophysical model for each fluid.

For each case, there are diverse models, some of them use constant properties during the entire simulation and some use a polynomial equation to calculate the property, such as the following formula.

$$C_p = \sum_{i=0}^{N-1} a_i T^i$$

But these polynomials are only dependent on one field, either pressure or temperature. The standard models in OpenFOAM are enough for incompressible fluids, chemical reactions or situations where the change in the properties according to pressure and temperature evolution is not significant. However, in the opposite situation, when properties are influenced by pressure and temperature in a relevant way, accurate models are required.

System codes get fluid properties from steam tables, either from polynomial functions or tabulated values, in both cases using two thermodynamic variables as inputs paramaters, pressure and temperature for instance. Using this methodology, properties at saturation conditions can also be obtained.

The steam tables are a data set that contains the thermodynamic properties of water and steam. This data set is tabulated in terms of pressure and temperature, using a wide range of values. Using the steam tables, the standard international values for the different water properties is applied. The standard used in this work was the one adopted by the International Association for the Properties of Water and Steam (IAPWS) in 1997 (Wagner, 2008). This organism is responsible for the international standards for thermophysical properties. The routine implemented that permits the access to these tables was based on a previous work Thiele and Lucas, 2016.

In particular in this work, the new solver was linked to an external module that computes the thermophysical models using the IAPWS-IF97 steam tables (IAPWS, 2019). This module consists in a external library for OpenFOAM that includes a new complete thermophysical model that allows calculating the fluid properties through the steam tables. It has been developed by Thiele and Lucas (Thiele and Lucas, 2016). This model is classified as a density-based model and the properties are calculated in an external application written in C called **freesteam** (Pye, 2013). This program is open sourced and it consists in the implementation of the standard IAPWS-IF97 steam tables that allows the calculation of the water properties for a wide range of pressures and temperatures. Consequently, both the transport properties, such as viscosity or conductivity, as well as termodynamic properties, like density, heat capacity or entropy can be obtained from this application. The range of working pressure and temperature are bounded by the limits of the steam tables. These are $0\,^{\circ}\mathrm{C} < \mathrm{T} < 800\,^{\circ}\mathrm{C}$ for $\mathrm{p} < 1 \times 10^{8}\,\mathrm{Pa}$ and up to $800\,^{\circ}\mathrm{C} < \mathrm{T} < 2000\,^{\circ}\mathrm{C}$ if $\mathrm{p} < 1 \times 10^{7}\,\mathrm{Pa}$.

Thiele and Lucas work included the necessary commands to call the steam tables from the solver and to make the definition available for OpenFOAM® solvers of the compressible family. This conversion is necessary in order to get a translation between C++ OpenFOAM® routines and C language in the *freeSteam* library

(Pye, 2013). Therefore, the main purpose of this extension is to translate the information so that the new 1D solver can use it.

Once the program and the external library are compiled, the procedure consists in defining the use of this library in the input and, every time the thermophysical model is applied, this library will be called, which, in turn, will access the program freesteam, and the property or properties are calculated according to a given pressure and temperature. Then these properties are used in the solver.

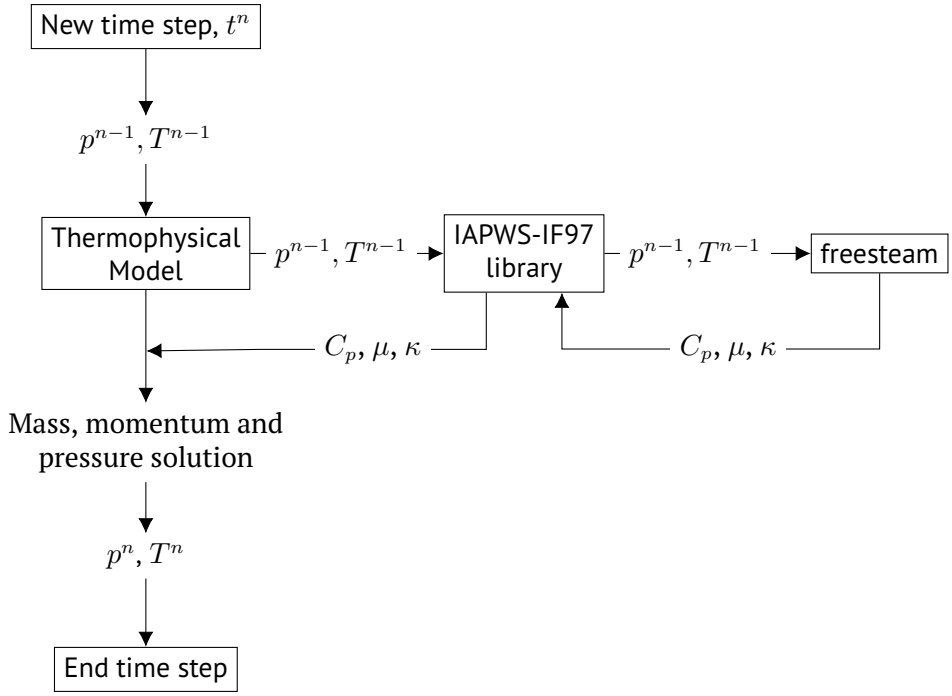


Figure 4.17: Use of the external thermophysical model.

The definition of the new thermophysical model in the thermophysicalModel input file is given in figure 4.18.

The external IAPWS-IF97 library was implemented to work with compressible single-phase fluids, since that was the requirements defined by the authors for their work. Therefore, the library takes the current pressure and temperature and sends them to the freesteam program to get the fluid properties, which are sent back to the main solver.

```
thermoType
{
    type            heRhoThermo;
    mixture         pureMixture;
    transport       IAPWSTransport;
    thermo          hIAPWS;
    equationOfState  eosIAPWS;
    specie          specie;
    energy          sensibleEnthalpy;
}
```

Figure 4.18: Definition of the new user application for thermophysical model.

During the first part of this work these tables were used without modifications. However, they were not suitable to work with two-phase flows, where most properties are used at saturation conditions. In this situation, it was observed that the module did not give back the proper value because it was accessing the tables with pressure and temperature and it could not properly check which side of the bell it was being requested. Figure 4.19 shows an example for a fluid at $200C$.

Therefore, the external IAPWS-IF97 library has been updated in order to be able to:

- Give back liquid and vapor properties at saturation conditions.
- Access the steam stables with pressure and **enthalpy** values, so the state of the fluid is accurately defined.

The process of updating the library implied the modification of this file, but also the modification of the thermophysical model class, because the different properties that are called should belong to the class.

Regarding to the first requirement listed above, it was necessary to start by defining the saturation temperature for a particular pressure. Once the saturation temperature was determined, it remained to figure out if the conditions correspond either to liquid or vapor state.

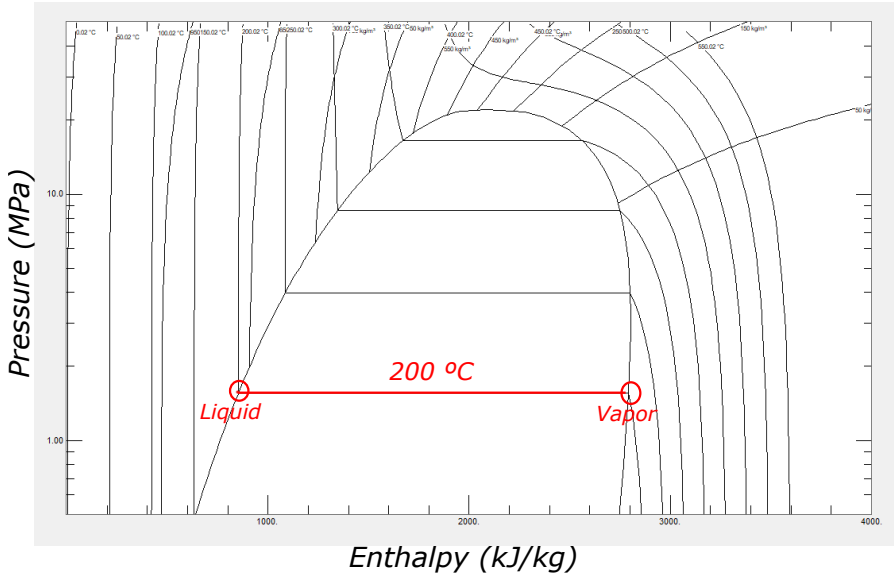


Figure 4.19: A fluid has different properties for same saturation temperature depending on its state. Graphic obtained from REFPROP (Lemmon et al., 2018).

On the other hand, accessing the steam tables with enthalpy instead of temperature avoids lacks of information in case the fluid reach the interior of the bell due to a overheating of the liquid or vapor subcooling. Table 4.4 presents a list of the different functions that were created to update the library.

The process of updating the external library involves various steps. First of all, the new functions were created in the external IAPWS-IF97 library, and the proper calling to freesteam program was defined for each function. Figure 4.20 shows the calling to freesteam to get the saturation temperature for a particular pressure.

Once the new functions were defined in the external library, it is necessary to define them within the object thermophysical Models. This is an intrinsic feature of OpenFOAM, where all the functions should belong to an object. When the function is related to thermophysical properties, the corresponding object is thermophysical Models, as mentioned before. Within this object, each property belongs to a particular class from the group of classes mentioned at the beginning of the section.

Table 4.4: List of new functions implemented in IAPWS-IF97 library and in the thermophysical model.

Functions implemented	Description
tsat_p(scalar p)	Saturation temperature for a given pressure
hsat_Tx(scalar T, scalar x)	Saturation enthalpy for a given temperature and quality
rhoVsat(scalar T)	Density for saturated vapor for a given temperature
rhoLsat(scalar T)	Density for saturated liquid for a given temperature
cp_v_Tx(scalar T)	Heat capacity for saturated vapor
cp_l_Tx(scalar T)	Heat capacity for saturated liquid
mu_ph(scalar p, scalar h)	viscosity for a given enthalpy and pressure
tc_ph(scalar p, scalar h)	thermal conductivity for a given enthalpy and pressure
beta_ph(scalar p, scalar h)	thermal expansion for a given enthalpy and pressure

```
//congoque: Tsat for a given pressure (K)
Foam::scalar Foam::tsat_p(scalar p)
{
    return freesteam_region4_Tsat_p(p);
}
```

Figure 4.20: Function in IAPWS-IF97 library for accessing steam tables to get saturation temperature for a particular pressure.

Therefore, in order to properly link the external library to the solver, most of the new functions were defined within the class thermo, focused on enthalpy calculation. Following the same example, the definition of the function to get the saturation temperature within the class thermo is shown in figure 4.21.

```
//- congoque: Tsat for a given pressure (K)
virtual tmp<volScalarField> Tsat() const;
//
// Tsat for a given pressure (K)
virtual tmp<scalarField> Tsat
(
    const scalarField& p,
    const label patchi
) const;
```

Figure 4.21: Definition of the function to calculate the saturation temperature for a given pressure within the class thermo.

Finally, the calculation in the solver of saturation conditions and fluid properties for a particular pressure and enthalpy of the fluid was implemented in a file called fluidProperties.H and part of this implementation can be seen in figure 4.22.

```
/*liquid Properties*/


volScalarField SatTemp = phase2.thermo().Tsat();
volScalarField hlsat = phase2.thermo().HLsat(SatTemp);
volScalarField rho2Sat = thermo2.rhoLsat(SatTemp);
volScalarField Cp2Sat = thermo2.CpSatl(SatTemp);
volScalarField mu2 = thermo2.mu_h(p, he2);
volScalarField kappa2 = thermo2.kappa_ph(p, he2);
```

Figure 4.22: Definition of the thermophysical model for a particular fluid in the new solver.

CHAPTER 5

METHODOLOGY (II): ONE-DIMENSIONAL SOLVER DEVELOPMENT

This chapter presents the final development of the one-dimensional solver. It collects the different aspects introduced in section 4 and features already seen in chapter 3 along with the modification of the standard solvers used as base. This section is divided into two parts, the two stages in which this thesis was developed: the one-dimensional single-phase solver and its evolution to a two-phase solver. The final version of the new solver is called my1DTPFoam.

5.1 Introduction

The development of a one-dimensional solver within the OpenFOAM CFD platform is the main target of this thesis, as mentioned in previous chapters. Coupling a 3D geometry with a 1D model allows the use of a unique time step, as well as only one matrix system, simplifying the complexity of the calculation and avoiding unnecessary external communications that may delay the time of computing.

In order to get this target, OpenFOAM was selected due to its open-sourced feature that gives the developer full control over each of its applications, as explained in chapter 4. Furthermore, the use of a one-dimensional open-sourced solver allows the verification and validation of different and new correlation due to the possibility of being implemented and simulated using a free license. In addition to this feature, OpenFOAM is able to calculate systems with various do-

mains and at different scales, considered therefore a multi-physics multi-scale platform.

This work was divided into two stages: First, a one-dimensional single-phase flow solver was developed, since its performing and verification was simpler and could help us to totally understand the library OpenFOAM. Once this work was already completed and the code was fully understood, the step to a two-phase flow solver development was given. In this second stage, all the previous work was used as a basis, since the final code should be able to simulate one-dimensional single and two-phase flows and boiling phenomena.

5.2 Single-phase flow solver

In order to develop the one-dimensional solver for single-phase flows, the standard solver buoyantPimpleFoam was taken as basis, as mentioned in section 4.1.1. This solver is meant for compressible fluids in transient cases where heat transfer occurs.

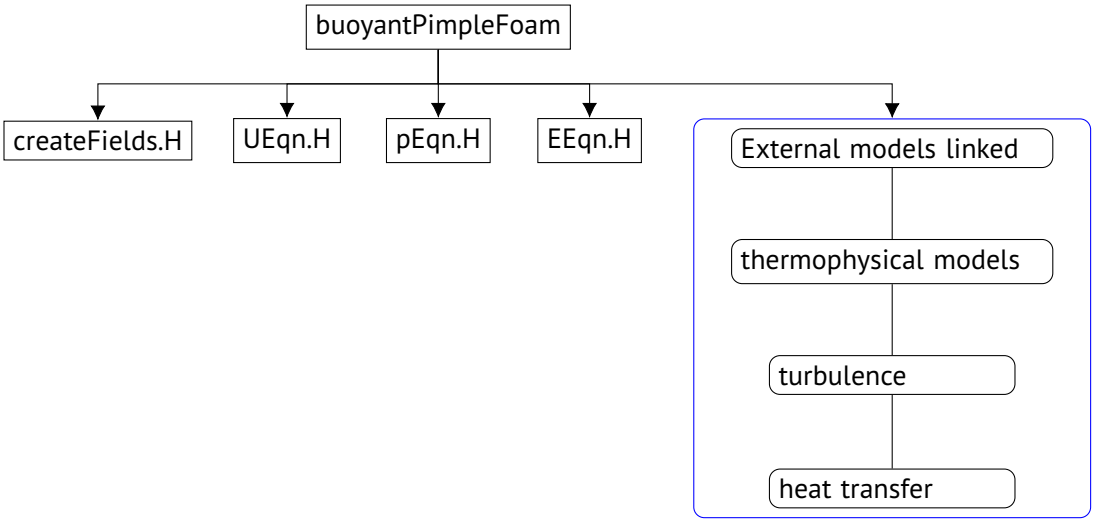


Figure 5.1: buoyantPimpleFoam structure.

Figure 5.1 shows the structure of the standard solver, which is made up of files that are own and external models (objects) that are common to different solvers. The latter ones are shown in the figure surrounded by a blue rectangle. Briefly describing each file from left-hand side, createFields.H initializes the main vari-

ables involved in the solver, UEqn.H sets the predictive velocity, pEqn.H solves pressure equation and corrected velocity, and EEqn.H calculates the enthalpy. Regarding to the external libraries, the names are self-explanatory: thermophysical models manages the different models to calculate the fluid properties; turbulence works similar with the turbulence models and finally heat transfer deals with the possible heat exchanges or sources.

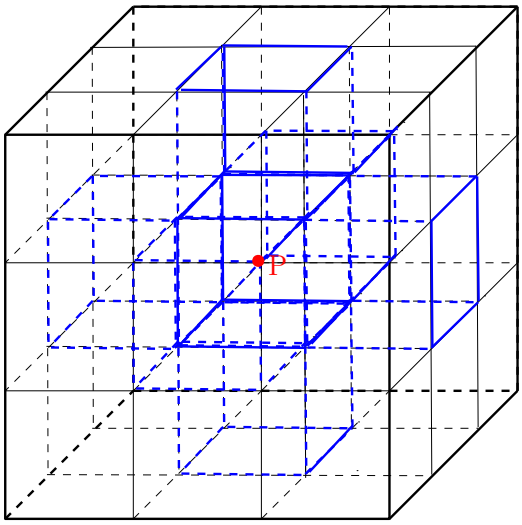
Departing from the standard solver, a series of tasks must be carried out to achieve the required solver. The main points are

- One-dimensional transformation of the solver and assessment of the different terms of the conservation equations
- Turbulence removal
- Wall friction and wall heat transfer models addition
- External solid mesh annexing

The tasks are listed in priority order to obtain the desired solver. However, during the development of the tasks, it is necessary to replace turbulence before transforming to one-dimensional calculations, since turbulence is a phenomena that only makes sense in three dimensions. In order to substitute this phenomena, first, the terms in the equations that accounts for turbulence effects are replaced by a wall friction model that considers the axial effect of the viscous forces. This simplification was shown in section 3.2. Once 3D turbulence terms are out of the equation, it is possible to remove from the solver some of the classes which are related to turbulence. This is due to the fact that OpenFOAM is written in C++ language, which one of its powerful tools is the object-oriented development. Erasing the classes that are not used and do not influence others reduces the number of tasks the code should perform so less time is required and thus, the solver gains velocity since it has to create less variables.

The following task is the simplification of the equations to calculate only in the preferential axis. The reduction of the number of direction in which the system must be solved is directly related to the matrix system itself. The matrix system is composed of the terms of the linear algebraic equations discretized for each cell. In FVM strategy, the size of the matrix system is equivalent to the number of cells of the mesh, since the equations are discretized in each CV, and each line corresponds to the discrete algebraic equation system for a particular cell. In a general system, only the corresponding matrix terms of this particular cell in which the equation are being applied and the surrounding cells terms are non-zero. The rest of terms in this row are null. The advantage is that those null

terms do not need to be stored (Versteeg and Malalasekera, 2007), so the memory requirements are not dependent on the matrix size. Therefore, reducing the number of directions to be solved, the number of non-zero entries are reduced, and hence, the stored system is also reduced.



(a) Cells involved in cell *P* discretization

$$\begin{pmatrix} a_1 & -a_{1,e} & -a_{1,n} & -a_{1,t} & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & -a_{P,w} & -a_{P,s} & -a_{P,b} & a_P & -a_{P,t} & -a_{P,n} & -a_{P,e} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & -a_{m,b} & -a_{m,s} & -a_{m,e} & a_m \end{pmatrix}$$

(b) Matrix structure of mesh in 5.2a

Figure 5.2: In a 3D geometry, each cell discretization involves the 6 surrounding cells.

In this context, figure 5.2a shows an example of a 3D mesh divided into 27 cells. If one focuses on the cell whose centroid is P, it can be seen that each face of this cell is in contact to another cell which will impact in the dynamics of P cell. Calling the adjacent cells *west* (w) and *east* (e) in X-direction, *north* (n) and *south* (s) in the Y-direction and *top* (t) and *bottom* (b) in the Z-direction, the discretization of cell P will be given by image 5.2b. In this figure, one can see the general

structure of the dependent matrix (matrix A in the system $Ax = b$). In this matrix, every row accounts for the terms that involve a particular cell, so the size of the matrix will be determined by the size of the mesh. Therefore, focusing on the corresponding row of cell P, one can see there is a positive term that represents P itself and then, there are a non-zero term for each cell that influences P, which are the surrounding cells. This is repeated for every cell and the positions for cells that do not interact between them are set to zero. Therefore, the dependent matrix can be considered as a sparse matrix, due to the amount of terms that are set to zero and that, according to Versteeg and Malalasekera, 2007, are not stored in the memory. However, for a mesh with thousands or millions of cells, the system can still be huge. The value of each matrix term will depend on the discretization method selected for each equation term. The user should select a temporal discretization method, as well as a gradient, divergence or laplacian discretization methods, among others.

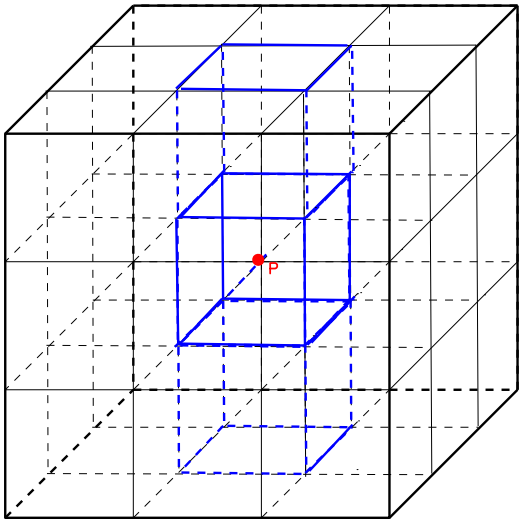
It is also worthy to mention that the opposite philosophy, which is building a mesh with big cells so that the matrix system remains small is not recommendable, since big cells could lead to numerical diffusion due to the fact that changes in the transported property may not be properly acquired. And this leads to accuracy losses.

The parameter that provides a good idea about the correct size of the cell is the Courant number, also known as CFL (Courant-Friedrchs-Lewy) number and it is defined as

$$CFL = \frac{U\Delta t}{\Delta x}. \tag{5.1}$$

This number is a ratio between velocity and time against size of the cell, and it should not be bigger than one. It allows to set the computational time similar to the time required by the property to cross the cell at a given velocity. Setting a proper Courant number allows avoiding numerical diffusion.

Applying the same philosophy for a one-dimensional calculation, the matrix system is even much simpler. Figure 5.3a shows the cells that influences cell P for the same mesh as in the previous case but applying a one-dimensional discretization. In this case, only one axis is considered, which, following same nomenclature as before, Z-direction has been considered, so only top (t) and bottom (b) cells can affect P cell evolution.



(a) Cells involved in cell *P* discretization in 1D simulation

$$\begin{pmatrix} a_1 & -a_{1,n} & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & -a_{P,b} & a_P & -a_{P,t} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & -a_{m,s} & a_m \end{pmatrix}$$

(b) Matrix structure of mesh in 5.3a

Figure 5.3: In a 1D geometry, each cell discretization involves only 2 surrounding cells.

The resulting matrix for this discretization is shown in figure 5.3b, where the size of the matrix is the same as before, which is consequent with the mesh and now there are more zero terms, reducing the memory requirements of the system.

In order to reduce the new solver calculations only to one dimension, it was enough to indicate this to the solver. The command mesh.solutionD() controls every axis, and it was forced to calculate only one axis, in particular X-axis. Figure 5.4 shows the lines implemented in the solver in order to set the calculations only in one axis. In this figure, the vector structure solD contains the solution for every axis in each one of its components, where solD[0] contains the solution for X-axis, solD[1] for Y-axis and solD[2] for Z-axis. Setting solD[1] and solD[2] to

-1, the calculations are disabled. Regarding to the discretization methods, this model requires of the same methods previously mentioned, being able to accept any of the methods implemented in OpenFOAM.

```
/*****congoque: Disabling solution in two axis*****/

Vector<label>& solD =
const_cast<Vector<label>&>(mesh.solutionD());

Info« "Disabling solution in the y & z direction" « endl;
solD[1] = -1;
solD[2] = -1;

//****************End**********************//
```

Figure 5.4: Setting one-dimensional calculations.

Once the 3D turbulence has been removed and the system simplified to 1D, it was necessary to closure the system by adding the wall heat transfer coefficient model (see section 3.3). In order to include the different models in the solver, two new classes have been created, once for each model. Implementing these two models as new classes, it gives them independence, being able to be used in different solvers or applications.

Therefore, a new class, called WallFriction have been developed to use the different wall friction models in the new solver. This class can then contain as many different wall friction models as wished (as long as one implements them). Same happens to the class of wall heat transfer models. This class was called htcModels and for the moment it only contains the *Gnielinski model* (see section 3.3.2). Regarding to the wall friction models, two different correlations were implemented, as seen in section 3.3.1.

5.3 Solid mesh implementation

5.3.1 The blockMesh utility

Once all the fluid models were developed in their respective classes and properly implemented in the code, the last step in this first stage consisted in the integration of a solid system. In order to include this new domain, three main steps were accomplished: The integration of the new mesh in the solver, the implementation of the solution system and the assessment of the feedback with the fluid domain. All three steps have been already mentioned in different sections of this document (in particular in sections 4.6 and 4.3.3, respectively).

The methodology to integrate the solid mesh in the solver consisted in using part of the routines already applied in OpenFOAM to create a fluid mesh.

In the standard OpenFOAM platform, a mesh is built by means of an external utility called blockMesh. This utility reads the information from the input and uses different libraries to represent the mesh. This mechanism only allows the generation of hexahedral meshes, so no tetrahedra nor unstructured meshes can be generated by this method. The basis of blockMesh is the domain decomposition into different hexahedral blocks, where each block is a particular region of the domain. Every block is defined by 8 vertices that establish the limits in this block. Furthermore, each block can be labeled and then identified by its label. The discretization may vary for the different blocks, as long as the number of faces in the boundary coincides with the number of faces of the neighbor block boundary. Figure 5.5 shows an example of a mesh composed of different blocks.

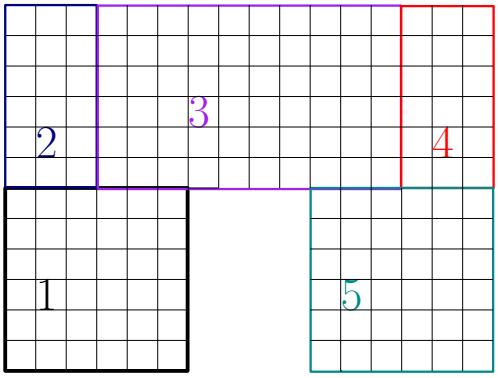


Figure 5.5: Multi-block mesh.

The information about the vertices, blocks and definition of boundary conditions is written in a file called blockMeshDict. This dictionary is read by the utility to create the mesh. BlockMesh is an external application and it is independent of any solver, so the usual process in a simulation is first generating the mesh by blockMesh and then running the solver, which reads the information created by the previous utility.

There is one exception to the general hexahedral mesh, which is the *axisymmetric* mesh. BlockMesh allows to build a geometry of tetrahedral cells to simulate cylindrical geometries using a very small angle, usually 5 degrees. When this mesh is used, the system is solved as a pseudo-2D model, where axial and radial direction are calculated. This mesh reduces the computational resources, but still has some drawbacks: First, the volume of the cells close to the axis may have very small values, giving numerical issues. Furthermore, the axial direction must be always the same, being unable to solve elbows or geometries where the preferential axis changes the direction. Figure 5.6 shows an example of an axisymmetric mesh with one radial node per axial level.

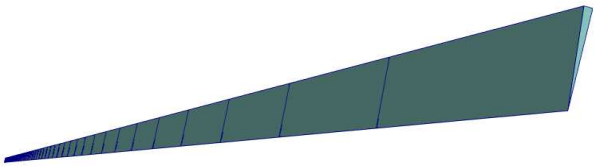


Figure 5.6: Axisymmetric model built with blockMesh.

5.3.2 Solid mesh development

Considering the blockMesh process, same methodology was followed for the solid mesh. A new file was created to write the solid domain information. This file, called blockMeshSolidDict, is stored along with the original blockMeshDict. The main difference is that in case of the solid mesh, the solver itself will call the application to create the solid mesh.

In order to implement this functionality in the solver, some files from the source were modified and adapted for this solver. First file taken from src was the library createMesh.H, which was renamed as createSolidMesh.H. This file was modified so that the information was always searched in the blockMeshSolidDict. Also, the names of the classes and different variables were created with the suffix *Solid*, in order to avoid misunderstandings with the fluid mesh. File createSolidMesh.H reads the information included in the input dictionary and creates the class poly-

MeshSolid, where the cells, patches and faces information is stored and sorted following a particular order: the numbering of the vertices, faces and cells starts always from the closest position to the origin of coordinates, prevailing X-axis, then Y-axis last Z- axis. Once the all the structure is set and the different zones (if exist) are defined, this information is stored in the fvMesh **meshSolid** class, which will be used then in different routines of the solver.

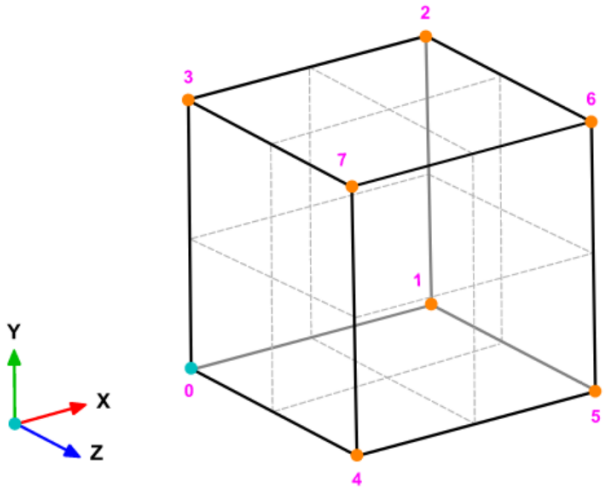


Figure 5.7: 8-cells mesh construction (Dynamics, 2019).

A second file was also copied from the src and adapted, which was createFields.H file. The original version of this library creates the different main variables of the problem (velocity, pressure, temperature...) and provides them with an structure that allows to relate each value with a particular cell, face or patch. The modified version of this file was called createSolidFields.H and in this document the different solid variables were created. In the solid domain, the main parameters are wall temperature, conductivity, density and heat capacity. These variables were created as an structure where the information is stored according to the solid mesh. Therefore, there are internal values corresponding to the cell values, which are set as vector of values, where size of this vector corresponds to the number of cells of the solid domain. Similarly, this structure also stores the value of each face of the different patches and it allows the calculation of the internal face values by a simple interpolation. Figure 5.8 shows the declaration of the wall temperature in the file createSolidFields.H.

```
volScalarField Tw          //Wall temperature variable
(
     IOobject
     (
        "Tw",
        runTime.timeName(),
        meshSolid,
        IOobject::READ_IF_PRESENT,
        IOobject::AUTO_WRITE
     ),
     meshSolid,
     dimensionedScalar
     (
        "Tw",
        dimensionSet(0, 0, 0, 1, 0, 0, 0),
        298.0
     )
);
```

Figure 5.8: Definition of Wall temperature in createSolidFields.H.

In figure 5.8 there are different terms that should be briefly described: All the information related to the variable (in this case, related to *Tw*), including boundary conditions, is stored in the structure volScalarField. Variables that are read from input files belong to IOobject class. This type of objects can be read always (IOobject::MUST_READ), never (IOobject::NO_READ) or only if the input file exists (IOobject::READ_IF_PRESENT). This information should be found in a file called *"Tw"* and its size will be that of meshSolid, which is the stores the mesh data. Up here, the information is enough for those variables that must be always read. However, if the object may not be among the input files, some extra information must be provided. In particular, it is necessary to provide the units of the variables and an initial value. In this case, the initial value given is 298.0, and Kelvin units are given following the general OpenFOAM structure to read variables, which is based on the international system and it follows the next order: [kg, m, s, K, mol, A, cd]. For units which are inversely proportional, the value will be negative.

Therefore, eventually, two different geometries are modeled, fluid domain and solid domain. Conservation equations are solved in the fluid region while the conduction equation is calculated in the solid domain. It remains to link both systems so that they exchange the required information to solve them considering the influence of the other system. This connection is performed according to section 4.6 development. Figure 5.9 shows the relation between meshes.

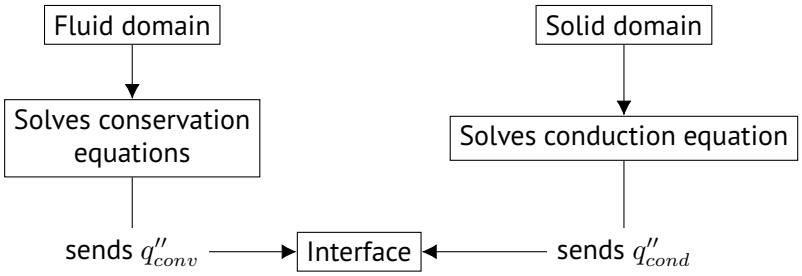


Figure 5.9: Mesh structure in the new solver.

5.3.3 User-defined solid properties

When a new region is defined, as happened in this case, the initial variables must be set-up. This is performed in 0 folder, where a new directory called with the same name as the new region has to be created (section 4.6). Furthermore, a secondary requirement for this new solver was to be able to work with solid variables that were temperature dependent. Thus, the solver is able to read solid properties that are tabulated according to temperature values. These values are user-defined and should be written in a file called materialProperties that should be located in constant/Solid, according to figure 4.3. The structure of the new file is the following: First, the properties to be read are defined. Next, the values of the temperature are given. Then, the value of the different variables for each temperature points are written. The solid parameters do not necessary have to be listed in the same order as they were defined. Figure 5.10 shows an example of this file where it is shown which properties are listed, the temperature points, and the heat capacity values. The file presented corresponds to a portion of the materialProperties file for a case with constant properties, for instance.

The methodology followed by the solver to update the solid properties each time step is as follows: Once the conduction equation was solved, the solid properties are updated according to the new wall temperatures. For each cell the value of every property is interpolated using this new temperature. The command used

```
SolidProp
( Temp
   Cp
   Cond
   Dens);
//Temperatures (K)
Temp
( 450.0
   550.0
   650.0);
// Heat Capacity (J/kg)
Cp
( 400.0
   400.0
   400.0);
```

Figure 5.10: Definition of solid properties in the file materialProperties.

in the solver is shown in figure 5.11, which shows the portion of the code where this interpolation is carried out.

```
/*———-Updating solid properties———————-*/
forAll(cellSol, cell)

Conduct[cell] = interpolateXY(Tw[cell], Temp_, Cond_);
SolidCp[cell] = interpolateXY(Tw[cell], Temp_, SolidCp_);
Diffus[cell] = Conduct[cell];

Diffus.correctBoundaryConditions();
/*———————End updating solid properties———-*/
```

Figure 5.11: Interpolation of the solid properties using the new wall temperature.

5.3.4 Final one-dimensional solver

Once the solver is complete, the structure of the final one-dimensional single-phase solver is shown in figure 5.12, where it is displayed in same terms as presented the original buoyantPimpleFoam (figure 5.1).

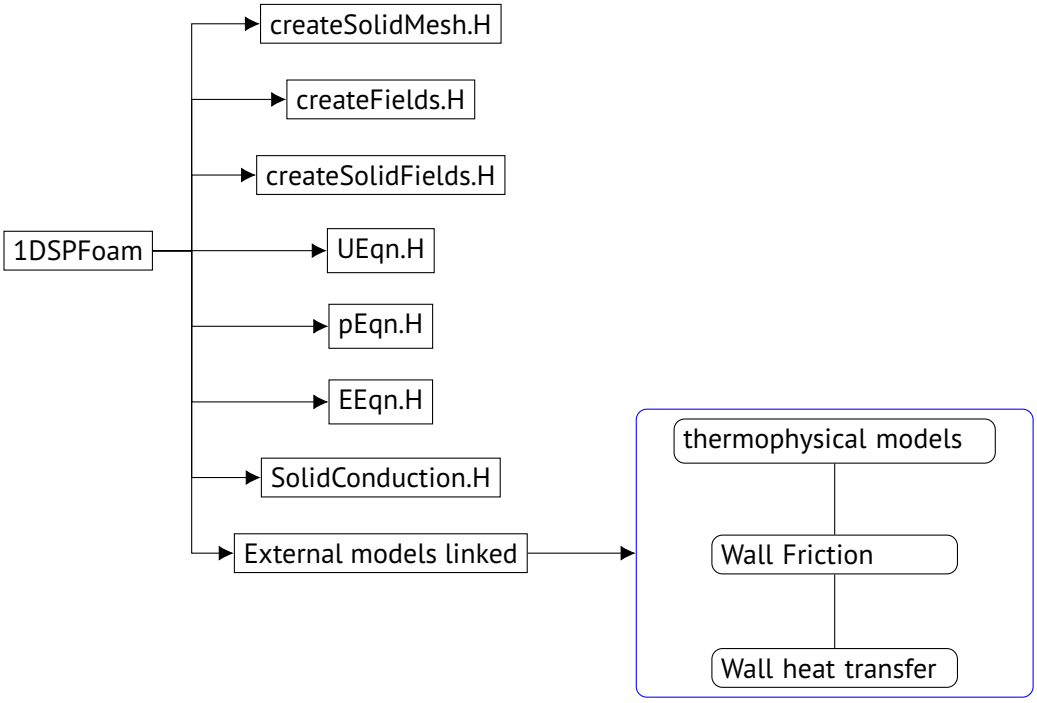


Figure 5.12: 1DSPFoam final structure.

In this figure, one can see the distribution of the new files and new classes. The latter are highlighted by a blue rectangle, where the class thermophysical models includes the modifications applied in order to use the steam tables to calculate the fluid properties; the class Wall Friction may involve as many different wall friction models as implemented and the Wall heat transfer class works similar but with wall heat transfer models. Regarding to the new files, and starting from the top, the file createSolidMesh.H covers the different tasks to create the solid mesh; the files createSolidFields.H involves the initializations of the solid fields and SolidConduction.H incorporates the resolution of the solid conduction and the routines needed for the heat balance and the fluid-solid feedback.

The presented structure results in the general loop of the solver, given by figure 5.13. In this figure, one can follow the sequence of the solver: First, the solid mesh is generated. Then, the fields of both liquid and solid systems are initialized and once all variables are initially set, the loop starts. First, the solid conduction is solved (SolidConduction.H), so the wall temperature is solved in an semi-implicit way, by using the liquid temperature of the previous time-step. Then, the terms related to the velocity are set so that the dependent matrix is built. Next, the energy equation is solved in (EEqn.H), but since the momentum equation is set but not solved, the energy system uses the velocity of the latter time step. In this file, enthalpies are calculated, and from these and from pressure, the temperature is solved. The last file of the loop is pEqn.H, where pressure and corrected velocity are calculated. These variables are located within the PISO loop, an it will be calculated as many times as indicated in variable nCorrectors (see section 4.1.1). Once the time step is finished, the output variables are written. The output is written according to the frequency set by the user in the controlDict.H input file.

The verification of this solver is shown in chapter 6, together with the results of the two-phase flow simulations.

5.4 Two-phase flow solver

So far, a new complete solver for one-dimensional simulations of single-phase flows has been presented. This solver is able to work with both liquid and gas states and it is optimized for water calculations, for which it was improved by adapting the steam tables for fluid properties acquisition.

This solver allows a wide number of different simulations, from fluid behavior through pipes in a PWR to chemical reactors or natural circulation. However, this stage was only a initial step to develop a more general one-dimensional solver, which were also able to perform two-phase flow simulations. These kind of flows play a fundamental role in nuclear field or petrochemical industry, among others.

In order to carry out the development of the two-phase flow, same procedure as in section 5.2 was followed. First, an OpenFOAM built-in two-phase flow solver was taken as a basis. Then, different tasks were performed to adapt the solver to the defined requirements. Initially, it was intended to apply the modifications to the solver 1DSPFoam, but it was necessary to modify the roots of the solver, due to the application of the eulerian model (explained in section 5.4.1). However, taking advantage of the flexibility of the library due to its classes and objects,

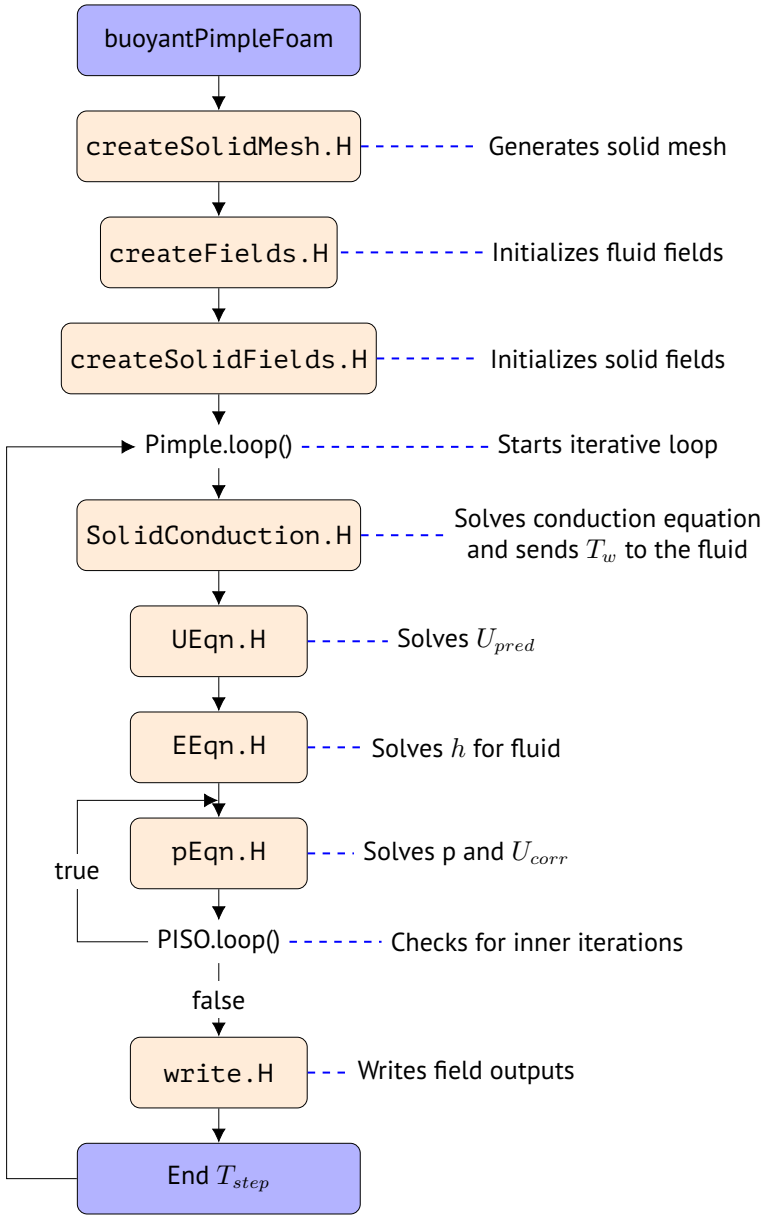


Figure 5.13: General loop of 1DSPFoam.

which are independent of the solvers and they can be applied to any of them, it was easier to select the most similar built-in solver and integrate the new classes.

The selected solver used as a base for the aim of this work was twoPhaseEulerFoam. The mentioned solver was meant to solve *a system of 2 compressible fluid phases with one phase dispersed, e.g. gas bubbles in a liquid including heat-transfer*, according to OpenCFD-Ltd. However, it does not consider boiling or condensation and therefore, it is either not able to take subcooled boiling phenomena into account. Furthermore, it only calculates the fluid parameters, without studying the wall temperature or solid heat conduction implications. The standard thermophysical models that calculates the fluid parameters are also limited to be constant or dependent of one variable. These disadvantages along with the computational time and resources required to solve the 3D mesh, where the time step is several times the order of magnitude of a single-phase simulation, brings the necessity of applying modifications and simplifications to this solver.

In this second part, all tasks already listed in section 5.2 must be undertaken here again, and, in addition, two main tasks are appended to the original list, which are

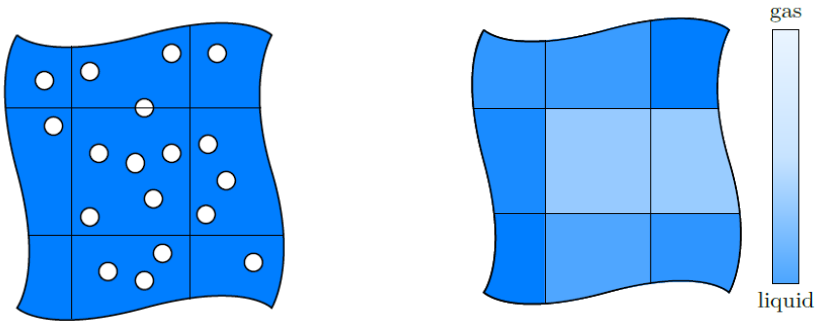
- Addition of the mass transfer term. The solver must be able to calculate boiling flows.
- Inclusion of a subcooled model. Subcooling may happen in bubbly and slug flows.

Along with these two new tasks, it is also remarkably the adaptation of the interfacial forces in order to satisfy one-dimensional simulations, satisfying the different requirements pointed out in section 3.7. The following sections will explain the main features that differentiate this solver from 1DSPFoam.

5.4.1 TFM numerical method

The solver twoPhaseEulerFoam belongs to the family of the solvers based on the Eulerian approach, according to the definition given in section 3.1. This approach is also known as Two-Fluid Model (TFM). Furthermore, since it is a solver used for two-phase flows, and each phase is treated as an interpenetrating continua represented by its own set of averaged equations, the basis of this solver is also known as Euler-Euler approach, due to the fact that both phases are based on the same model. The void fraction α, which is the relation term between phases, appears as a consequence of the discretization process. The development of this

approach was given in section 3.4. Figure 5.14 shows how the results of a TFM problem are seen due to the implicit simplifications.



(a) Discrete bubbles in a continuous liquid

(b) Continuum approach

Figure 5.14: Modeling approach on the example of a gas-liquid two-phase system (Holzinger, 2016).

In this image, one can see a real case on the left-hand side of the figure (image (a)), where the bubbles are dispersed within the continuous phase. On the other hand, an example of results given by an Eulerian-Eulerian calculation are presented in picture (b). In this image, the fluid is presented as a mixture where one can only obtain the influence of each phase in the cell, which is given as an average value. However, the bubbles are not detected. Therefore, this system does not provide a high resolution, but the computational time is lower than methods with higher resolution, such as Lagrangian methods.

The current twoPhaseEulerFoam is the evolution of a previous solver called bubbleFoam and whose development and justification can be found in Rusche PhD thesis (Rusche, 2002). The standard two-phase flow solver is based on averaged equations and it provides the results as mean values for the different fields. This is the usual method to solve two-phase flows when using an Eulerian-Eulerian system. Another main difference of this methodology respect the one seen for single-phase flows is that, due to instabilities found in regions where abrupt changes of density and void fraction occurs, the velocities are obtained from the phase fluxes. Therefore, the system is constructed for liquid and vapor fluxes, these are predicted and corrected and then the cell-centered velocity fields are reconstructed.

Figure 5.15 presents the current built-in twoPhaseEulerFoam solver. Cyan rectangles represent folders that contain different models or classes that can be selected to simulate that particular phenomena involved in the system (for in-

stance, in liftModels, the user can choose between Tomiyama, Moraga, LegendreMagnaudet or constant models to represent this force), while the black rectangles are files where the main routines are written. It can be seen that it is more complex than buoyantPimpleFoam according to the number of files and folders involved. This is expected since the physics of this system is also more complicated. Only the first level of the flow chart is going to be described here, since this solver is not the main purpose of this work, and the names of the different files are self-explanatory.

Starting from the top, in folder twoPhaseSystem, the main fields of each phase are defined, as well as the different functions that are related to the Eulerian method. In particular, mass equation is set and solved in this file. In phaseModel, both phases are initialized, relating all the fields that belong to a particular phase, such as the thermophysical model used in this phase to calculate properties or the different interfacial models assigned. The class diameterModel allows to select a model to calculate the bubble diameter among as many models as implemented. The selected model is defined by the user in the input. The files contErrors.H and corrcontErrors.H keeps the conservation of the equations satisfied. The former performs the calculation at the beginning of the time step, while the latter applies the correction once the pressure has been calculated. Regarding to the interfacial models, each force belong to its own class, where different approaches or correlations can be implemented. The following files have been seen before: EEqn.H includes the energy equation for both phases while createFields.H initialize the general variables, once both phases are set up. The field createFieldRefs.H stands for the initial pressure and phase fluxes, and write.H generates the outputs the selected time steps. The file phaseCompressibleTurbulentModels.H considers special models for solid-gas simulations. Then, there are two folders that contain the same information, these are pU and pUf. Both folders includes the files that performs the momentum equation calculations, but the difference between both files is that the former is applied when the mesh is cell-centered and the latter when the system is modeled using a staggered grid. This work is focused on cell-centered meshes, so only files located at pU folder are used. Along with the files and classes that belongs to this solver, the routines can call to general libraries that are shared with the rest of the models, such as the thermophysical models and turbulence library.

Figure 5.15: Built-in solver twoPhaseEulerFoam structure.

5.4.2 One-dimensional conversion

First steps in this two-phase flow solver were equivalent to those defined in section 5.2. First of all, the turbulence was replaced by the corresponding wall friction model and the calculation was reduced to an unique direction, typically X-axis, although the main direction is defined by the geometry. If the geometry was defined along Y-axis, the solution of the system will be calculated in this axis.

In addition to the one-dimensional conversion and the turbulence removing, the wall friction model was added. This solver was focused on bubbly and slug regimes, as mentioned in different moments during this document, and in this regime it is assumed that only water is in contact to the wall. Therefore, same model as in single-phase solver can be included in this version. Similar situation takes place with the convective heat transfer coefficient, which is calculated with same routine as in single-phase simulations (see section 3.3.2) for the liquid phase, but the wall heat transfer that represents the heat involved in subcooled boiling will be calculated by the subcooled boiling model.

Next, the addition of the solid mesh and the corresponding conduction equation is performed so that all the steps undertaken in the single-phase solver were also applied here.

Therefore, at this moment, the original twoPhaseEulerFoam, shown in figure 5.15, was converted to a one-dimensional solver that considers the conduction heat of an external solid. However, it still considers interfacial forces that does not apply in one-dimensional cases, such as lift, turbulent dispersion, wall lubrication and wall dependent forces, and it does not contemplates boiling situations or mass transfer between phases.

5.4.3 Mass transfer term

Originally, the standard solver twoPhaseEulerFoam was thought for an incompressible two-phase system (Rusche, 2002), but it evolved and currently it can be also used for compressible two-phase flows. However, it is yet not able to consider mass exchange, so it was necessary to implement this feature. Different authors have already done this task in previous works (Ghione, 2011, Rollins, 2018), although they worked with the incompressible version of the solver. Nevertheless, those works can be taken as a reference and only the main and novel contributions of this work are going to be presented here.

The standard mass equation can be found within the file twoPhaseSystem.C, which is the main file of the class twoPhaseSystem, which is the class where all the properties of each phase are defined and created the link to the phase. There, the equation is solved by a method called MULES (Multi-dimensional Universal Limiter with Explicit Solution), which maintains boundedness of the phase fraction by setting an upper and a lower limits ($0 < \alpha < 1$). MULES is based in Flux Corrected Transport method (Zalesak, 1979) and adapted by OpenFOAM developers specifically for this framework. The original calling of the method is shown in listing 5.16.

```
MULES::explicitSolve
(
    geometricOneField(),
    alpha1,
    phi_,
    alphaPhic1,
    Sp,
    Su,
    phase1_.alphaMax(),
    0
);
```

Figure 5.16: MULES calling in twoPhaseEulerFoam.

In listing 5.16, the terms S_p and S_u are the sources of the equation. S_p is the dependent source, which is a function of the compressibility and the void fraction, whereas S_u is the independent source term. Only the latter was modified so that it included also the possible mass transfer between phases.

Therefore, after the modification, the definition of the source term S_u can be seen in listing 5.17.

One of the problems of MULES is that it conserves volume, which if the fluid is incompressible, it is equivalent to conserve mass. However, if the fluid is compressible, it is not equivalent. Furthermore, the advection term needs to be stabilized due to its non-linearity (Márquez-Damián, 2013). Along with the presented disadvantages, it was shown that MULES algorithm converges with high difficulty for CFL > 0.1 (Pedersen et al., 2017) and there is a lack of information about this method.

```
volScalarField::Internal Su
(
    IOobject
    (
        "Su",
        runTime.timeName(),
        mesh_
    ),
// Divergence term is handled explicitly to be
// consistent with the explicit transport solution
    fvc::div(phi_)*min(alpha1, scalar(1))
);

//- congoque: Mass transfer source terms
Su[celli] += ( GammaEvap_[celli] - GammaCond_[celli] ) /
rho1[celli];
```

Figure 5.17: Definition of S_u.

Therefore, in order to avoid instabilities and the lack of references, this method was substituted in this work by a more known system to solve mass equation, which can be found in Rusche, 2002, Ghione, 2011 and Rollins, 2018. This new algorithm consisted in the resolution of the equation by a usual matrix system, where again the interfacial mass transfer term is included in the equation as a source term. The final equation implemented in the solver is equation 3.41 and it is presented in listing 5.18.

As mention in section 3.4, the original two-phase continuity equation was rewritten in order to keep boundedness. Consequently, same implementation was perform in the code and one can see the presence of the relative velocity (term phir in line 6 of listing 5.18) and the void fraction of both phases in the implemented equation. Besides, this system can be solved using the different discretization schemes and equation discretization methods provided by OpenFOAM.

```
/* ——————congoque: alpha1 Equation Solve ——————————— */
fvScalarMatrix alpha1Eqn
(
   fvm::ddt(alpha1)
   + fvm::div(phic, alpha1, alphaScheme)
   + fvm::div( -fvc::flux( -phir, alpha2, alpharScheme), alpha1,
alpharScheme)
   ==
   fvm::Sp(Sp, alpha1) + Su
   + fvm::SuSp
      (
      ( GammaEvap_ - GammaCond_ ) * ( 1.0 / rho2 - 1.0 / rho1 ) ,
alpha1
      )
);
alpha1Eqn.relax();

Foam::solverPerformance sp = alpha1Eqn.solve();
```

Figure 5.18: New mass equation definition.

5.4.4 Interfacial Models

In one-dimensional simulations only particular interfacial forces are involved. This was seen in section 3.7. Thus, it is only necessary to consider drag force and interfacial heat transfer.

The new models for these forces are included in the corresponding branch of the class interfacialModels, and the forces that are not considered in one-dimensional simulations were removed. Along with the new interfacial models, it was included a new model for bubble diameter (section 3.8).

The final interfacialModels class can be seen in figure 5.20. Models highlighted in red are the new models implemented for the one-dimensional simulations. The standard correlations for drag force and interfacial heat transfer which were already included were left remained in the solver for a future use.

The models of the different interfacial forces belong to the same class, called interfacialForces. This class is created specifically for the Eulerian-Eulerian solver, so it is an independent class. Every force requires of different classes from the source of OpenFOAM to work, from which they inherit part of their attributes, as seen in section 4.5. For instance, figure 5.19 shows the relation of the drag force with the different classes that are needed to set the drag model.

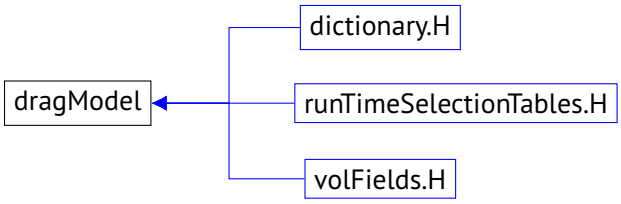


Figure 5.19: Drag model dependency from source functions.

According to figure 5.19 requires of the dictionary.H file, which allows the solver to read input files, the function runTimeSelectionTables.H, which allows the solver to select the model defined by the user among the different models implemented, and the file volFields.H which contains the routines to build the vector of fields in function of the mesh size.

Although these models where implemented to be used within the twoPhaseEulerFoam, these models can be called by different applications, provided that the links are properly defined.

5.4.5 Subcooled boiling model

The model to calculate subcooled boiling was defined in a separate class from the interfacial models. This correlation is written in the file SubcoolingModel.H and it is called from massTransfer.H, where the definition of Γ_k is found. The implementation was undertaken considering the different intervals defined in figure 3.4. Following that flowchart, list 5.21 shows the different conditions defined that control when the subcooled boiling model is calculated.

The different intervals are analyzed within a loop that goes through the all of them for each cell, since it may happen that two adjacent cells are in different regimes. Condensation and post critical heat flux regimes are not implemented, as they are out of the scope of this thesis. This work will be carried out in future steps. The adjustment of the critical heat flux temperature is also considered future work, so in this case an approximate constant value was given for the whole wall. Therefore, in case the fluid meets the conditions of these two regimes, it

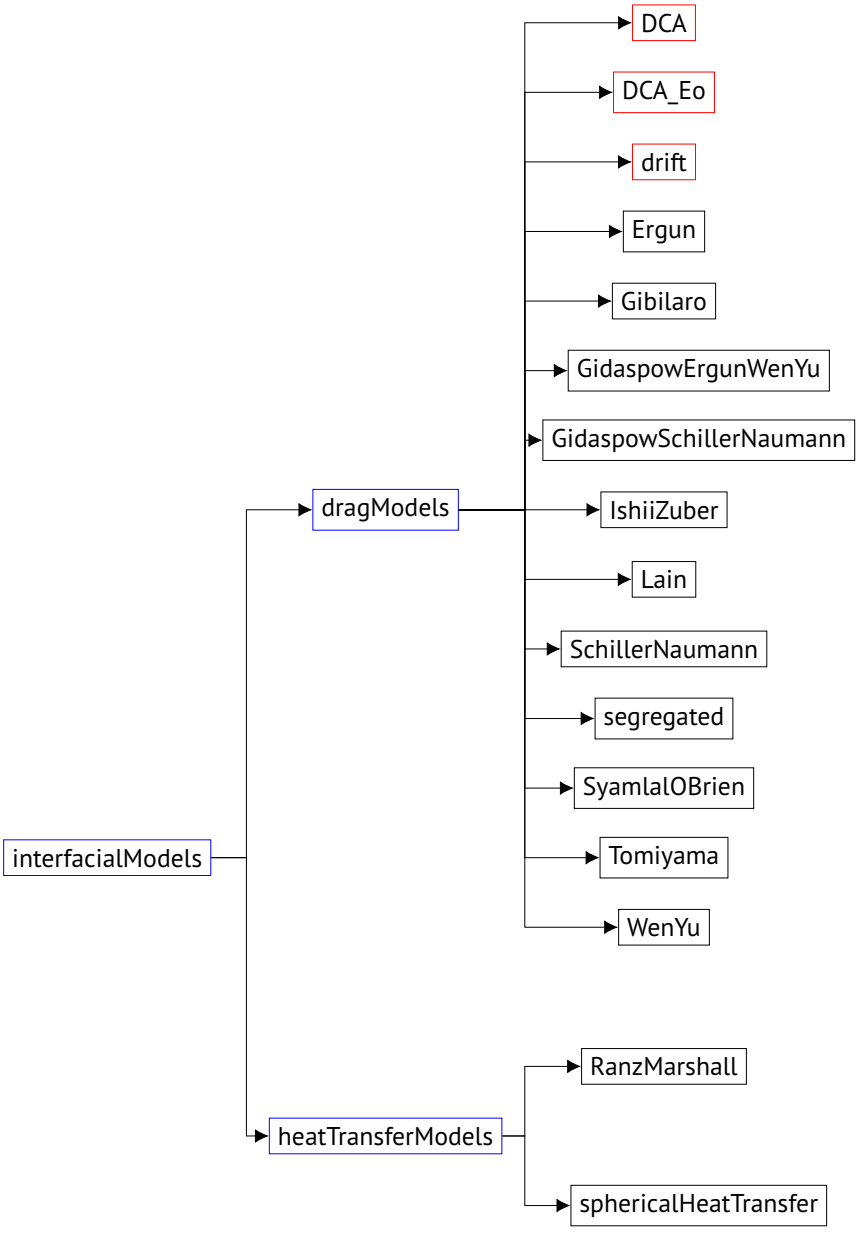


Figure 5.20: Final interfacialModels class.

```
if [(alpha1[celli]>0.8) && (twall[celli]<SatTemp[celli]) &&
(twall[celli]<T1[celli])]
{
   Info« "regime: condensation" « endl;
}
if (alpha1[celli]>0.999)
{
   Info « "regime: single phase vapor" « endl;
   qWGas[celli] = hFC[celli]*(twall[celli]-T1[celli]);
}
if (twall[celli] < tONB[celli])
{
   Info « "regime: forced convection in cell: " « celli « endl;
   qWall[celli] = qWall[celli] * ratio_areas;
}

/*————————-If Twall> Tonb————————*/
if (twall[celli] < tchf)
{
    if ((twall[celli]>SatTemp[celli]) && (twall[celli]>tONB[celli]))
//Evaporation due to subcooled boiling is calculated when the wall
//temperature is larger than tONB
   {
      Info« "regime: subcooled nucleated boiling" « endl;
      ...
   }
}
if (twall[celli] > tchf)
{
   Info « "regime: post critical heat flux" « endl;
}
```

Figure 5.21: Fluid regimes implemented in SubcoolingModel.H.

will be pointed out in which regime is the fluid, but no calculation will be undertaken.

Regarding to the subcooled boiling regime, the variable that determines when this condition is reached is the wall temperature. As explained in section 3.12, wall temperature should overcome saturation and reach the onset temperature (tONB). Cells where wall temperature is very close to the limit between force convection and subcooled boiling presented oscillations from one time step to the following, since wall temperature changed its value from greater to lower than tONB. This is due to the fact that when fluid is in forced convection regime, all the dissipated heat is due to forced convection. However, in subcooled boiling regime, the model implemented to take into account this heat considers different sources of heat, such as the pool boiling heat plus the forced convection. Furthermore, once the different heat sources in the volume are weighted, a portion of this heat is considered as evaporation heat, whereas a second part remains as convective heat and a third part is assumed to be condensed, since a part of the generated vapor will become liquid before reaching the bulk. The final heat due to subcooled boiling led to a heat transfer coefficient of this regime that considers the different phenomena and it is calculated from the global balance.

Even when the heat balance is properly reached, the different behavior between regimes generated instabilities close to the limit value of wall temperature where the regime changes. These oscillations were only overcome by using underrelaxation factors. In particular, a factor of 0.1 was used in this work. This factor was applied to the heat transfer coefficient of the different regimes, so that the new calculation was not so influencing. This was necessary because it is the wall heat transfer the link between fluid and solid, and the oscillations in this term affected directly to the wall temperature, as seen in section 4.3.3.

5.4.6 my1DTPFoam Final Loop

Considering the different sections explained along this thesis, figure 5.22 presents an step-by-step description of the general algorithm presented in this work. The final solver was called *my1DTPFoam*. It displays the order in which the equations are solved and which variables are calculated in each file, for a general time step n+1. This schematic is focused in the relation between variables and models and not in the iterative method itself, so the possible outer and pressure iteration are not displayed. Same final loop is used for both single-phase and two-phase flows, and depending on the temperature and void fraction, it will behave either as a single-phase flow or two-phase flow.

The loop unites the different section of this thesis and includes the equations used in the critical points of the solver. One of the most critical points was the subcooled boiling model, and its relation to the solid wall. When the wall tem-

perature reached the Onset Nucleate Boiling temperature (t_{ONB}) it automatically accessed the subcooled boiling equation and started to transmit heat to the vapor phase. The heat transfer coefficient of the dispersed phase (htc_{Γ}) became non-zero and this new heat distribution affected the wall, which assumed to dissipate a higher heat flux, and the wall temperature got reduced below the Onset Nucleate Boiling temperature next time step. This behavior created oscillations in the wall temperature which only was stabilized by using relaxation factors lower tan 0.1.

Starting from the beginning of the time-step, first task is the numerical setting, by defining the time step size. This can be fixed or adaptive. When the latter option is selected, the time step will be calculated by the code according to the maximum Courant number set at the input. If the former is used, the fixed time step will be used, but the Courant number will still be calculated. Once the time settings are calculated, next stage consists in the update of the saturation properties and the rest of thermophysical parameters in fluidProperties.H file. The inputs to this routine are the pressure and enthalpies of the latter time step. Next, the mass transfer term ($\Gamma_{Evap} + \Gamma_{Cond}$) is updated. However, in order to calculate the real mass transfer, it is necessary to determine whether there is subcooled boiling. Therefore, the subcooled boiling model is called (file SubcooledBoiling.H). Once the amount of subcooled vapor is predicted, the mass equation is solved and the new void fraction obtained. The mass equation is included as an object of the class twoPhaseSystem and it can be solved when this class is included.

Next steps involve the calculation of the conduction and conservation equations. A detail of the second part of the loop is shown in figure 5.23. First, solid conduction equation is solved (WallHeatTransfer.H file). The system is solved after the boundary condition of the inner surface of the solid was updated with the new heat transfer coefficients using equation 4.6. Once the new wall temperature is obtained, the solid properties are updated interpolating the new value of temperature. Following the conduction equation, next task consists in solving the momentum predictor system (UEqn.H). At this point, the wall friction and the interfacial drag closure models are determined. After the predicted velocity calculation the algorithm solves the energy equation in EEqn.H. Before calculating the equations it is necessary to set the interfacial heat transfer closure model. From the energy equation one can obtain the new enthalpies, and from these together with the pressure, the new phase temperatures will be updated in fluidProperties.H.

Finally, the momentum corrector is solved. This equation can be corrected as many times as pressure Correctors are defined in fvSolution file. First, the pres-

sure is calculated. Next, both continuous and dispersed phases are corrected. Compressibility term is updated, as well as density. Last task consists in writing the outputs, when it corresponds. And the time step ends. Up here, the new solver was summarized by briefly describing each main file.

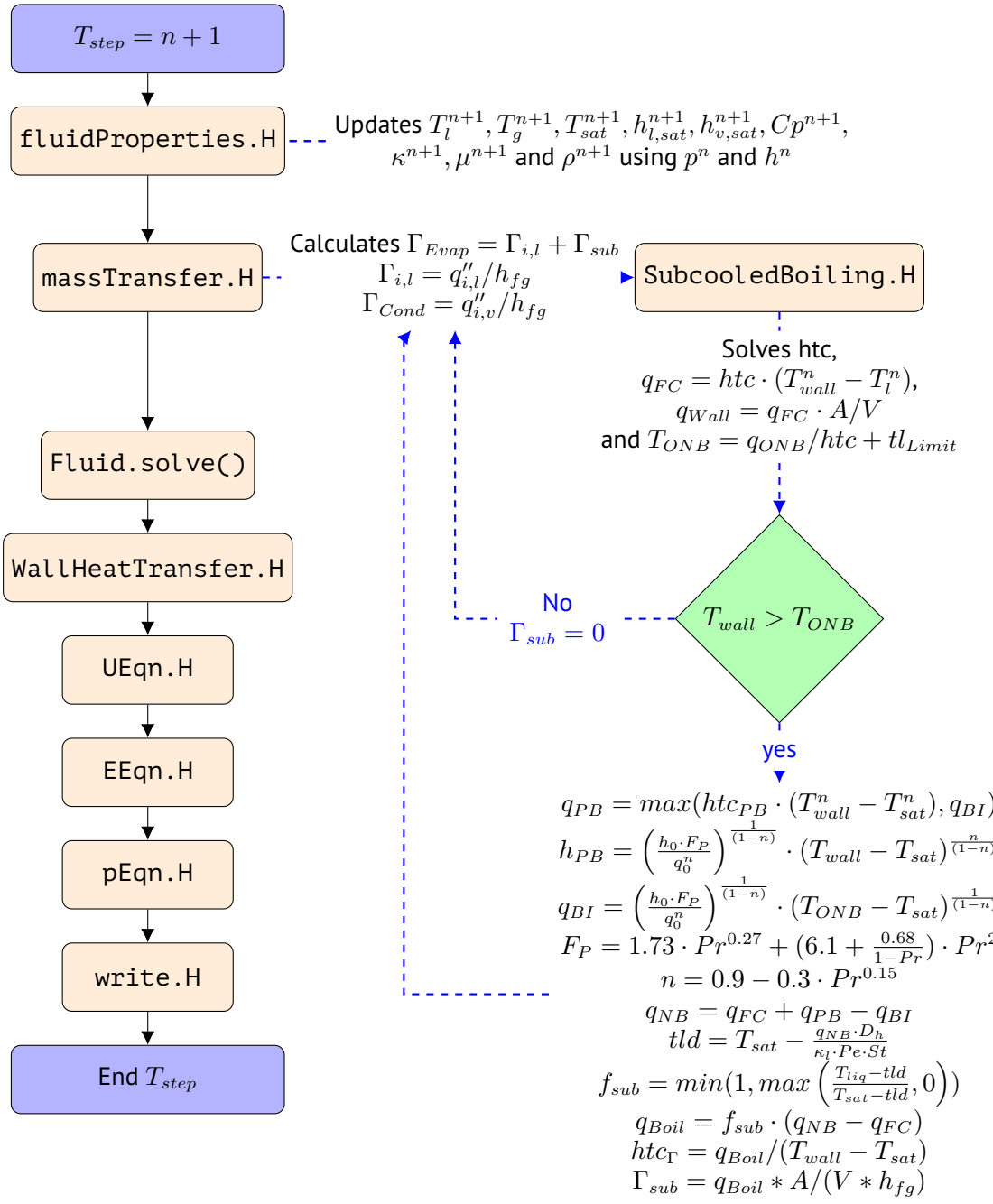


Figure 5.22: Final general loop of the two-phase solver.

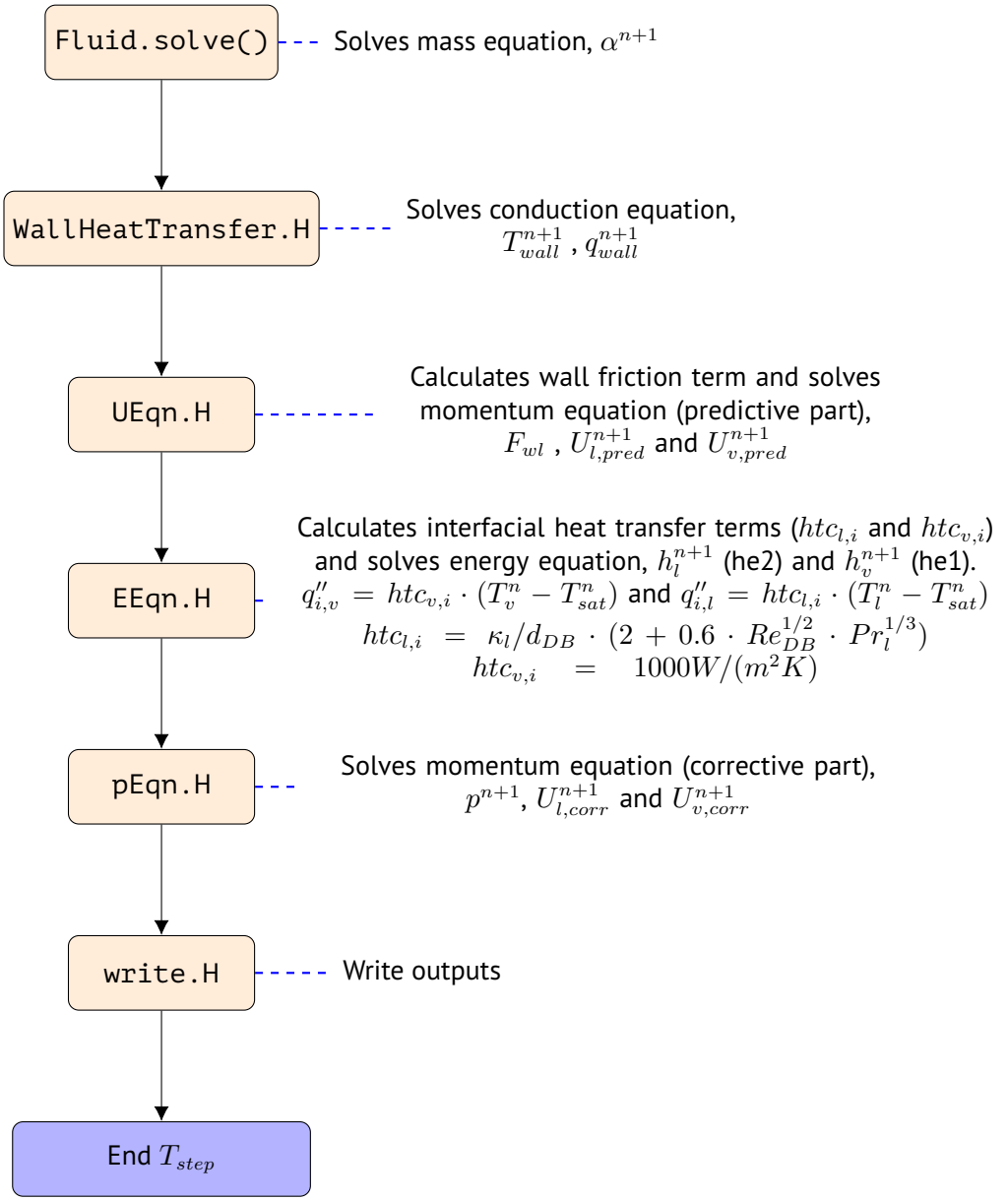


Figure 5.23: Detail of the second portion of the loop.

CHAPTER 6

APPLICATION AND RESULTS

This chapter presents the different test case simulations and the results obtained for each case. First, a single-phase case was tested in order to verify the heat transmission loop. Then, different simulations were performed for two-phase flows. In this case, an adiabatic validation and a system were boiling occurs are presented. The former consists in an experiment with water and air running through a vertical pipe, while the second one predicts the fluid and wall temperature and the void fraction generated in a water flow that runs upwards through a pipe. Results showed the agreement in the interfacial drag in one-dimensional simulation as well as a correct heat transfer modeling.

6.1 Introduction

The different chapters presented up here describes the diverse aspects and features of the new solver *my1DTPFoam*. These were tested by simulating various cases that allowed the verification and validation of the functionalities of this solver. As happened during the solver building, the test simulations were performed in two stages. First, test cases to check single-phase flows simulations and the capability of predicting the proper heat transfer were carried out. Then, once the final solver was built and the new two-phase flow features were set up, simulations with this solver were performed.

The process to set up a case before simulation implies a number of steps that shall be followed in order to give the whole information needed for the solver to

calculate the system. First of all, the main three folders described in section 4.1 must exist. The following step consists in the mesh generation. The information needed to create the mesh is included in the file blockMeshDict, were dimensions, blocks, cell size and boundary conditions must be defined. In case of considering a second mesh to represent a solid, a second file should be included, which must be called blockMeshSolidDict so that the solver can identify it. Along with the general information of the geometry, it is necessary to define the values of the boundary conditions and the initial values. This information is stored in folder 0, where there is a file for each field (liquid velocity, pressure, vapor temperature...). In these files, the initial value for the system as well as the derived type and the numerical value for each boundary condition is defined. With this general information the mesh can be generated.

In this work, the general mesh in all simulations consisted of a vertical pipe. Since the work is one-dimensional, there is one cell for each axial level. And, since the default mesher in OpenFOAM, called blockMeshDict uses only hexahedral cells, the default model in OpenFOAM is a rectangular pipe. Figure 6.1 shows a typical mesh used in this work were the fluid (blue region) is surrounded by a solid (red zone).

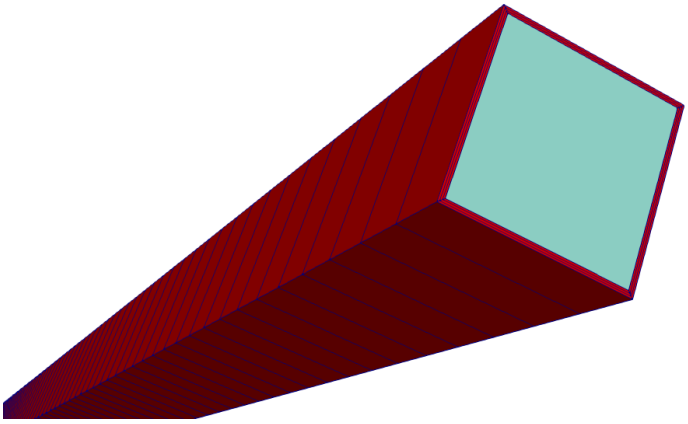


Figure 6.1: Portion of default mesh for the following simulations.

The grid definition plays an important role in terms of time-accuracy balance. It is necessary to find the adequate cell size that allows avoiding numerical diffusion while giving precise results within a reasonably computational time. In one-dimensional simulations this matter is not a limiting constraint since the number of cells is always very low in comparison to three-dimensional geome-

tries, but it is important to avoid the numerical diffusion and to consider this problem in case the system were coupled to a three-dimensional model.

Once the geometry is generated and meshed, the fluid and solid information must be completed. The fluid parameters and solid properties are defined in constant folder. The solver allows the user to define a wide number of different parameters: gravity, fluid transport properties, thermophysical model and interfacial models. The latter are included within a file called phaseProperties that allows selecting the correlation for each one of the interfacial forces. In this case, the user can select the drag force, which is the main force involved in 1D simulations, but he also can select the model to calculate the bubble diameter and the wall drag model.

Last step in the model set up consists in defining the numerical aspects of the case, which are all determined within system folder. This solver allows using all the capabilities of OpenFOAM about numerical aspects. The user can choose different discretization models in fvSchemes file, as well as the type of simulation, different matrix system solvers, number of loop iterations and relaxation values in fvSolution file. The time of the simulation, the time step and the Courant number are indicated in ControlDict file.

The second step is the execution of the solver, and then, once the problem was simulated, the results were post-processed and analyzed in order to verify and validate them. Verification was carried out against different codes, according the case. The main program used as reference to validate the results was TRACE, due to its extended international validation. TRACE allowed us to verify the capability of the new solver against different phenomena from a one-dimensional point of view.

TRACE model was also simulated during this work and the features of this calculation are going to be briefly summarized here. In this work, every TRACE simulation is based on the same general geometry where the features of each element of the model vary according to the case object of analysis. Therefore, the TRACE general schematic is composed of a pipe that simulates the fluid, and two especial components used to define boundary conditions. In particular, there is a FILL, which defines fixed velocity (or mass flow) at the inlet, and a BREAK, which determines a fixed pressure at the outlet. Besides, there is a component called HEAT STRUCTURE that plays the role of solid and allows to set a wall boundary condition at the external surface, either fixed temperature or fixed heat flux. Figure 6.2 shows the general schematic described here.

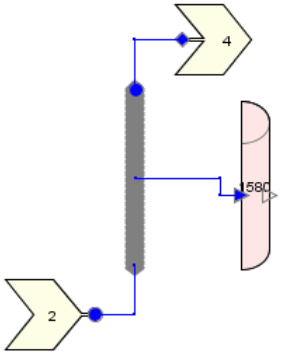


Figure 6.2: General model used for TRACE simulations.

TRACE model presented is valid for both single-phase and two-phase simulations, and it only changes different parameters of the elements included in model of figure 6.2.

In order to get a general perspective of the two models and a better understanding of the different components, figure 6.3 shows an schematic of both systems, where the same components are presented for each model.

In addition to the verification, in those cases were the experimental data is available, validation was carried out by comparison of the simulated results with the experimental values.

6.2 Single-phase flows simulations

First, simulations for single-phase flows were performed in order to test the capability of the solver to represent the heat transfer. The prediction of the temperature of both fluid and solid is one of the main requirements stated for the new solver and the presented case was created to test this capability.

The simulation performed for the single-phase flow verification consisted in a fluid (water) running upwards through a vertical pipe and heated by an external source. The inner pipe diameter is 0.104 m, total height is 4 m and the thickness is 0.003 m. The geometry was divided into 99 uniform axial nodes for the fluid, and the solid mesh was also represented surrounding the fluid and split in 3 reg-

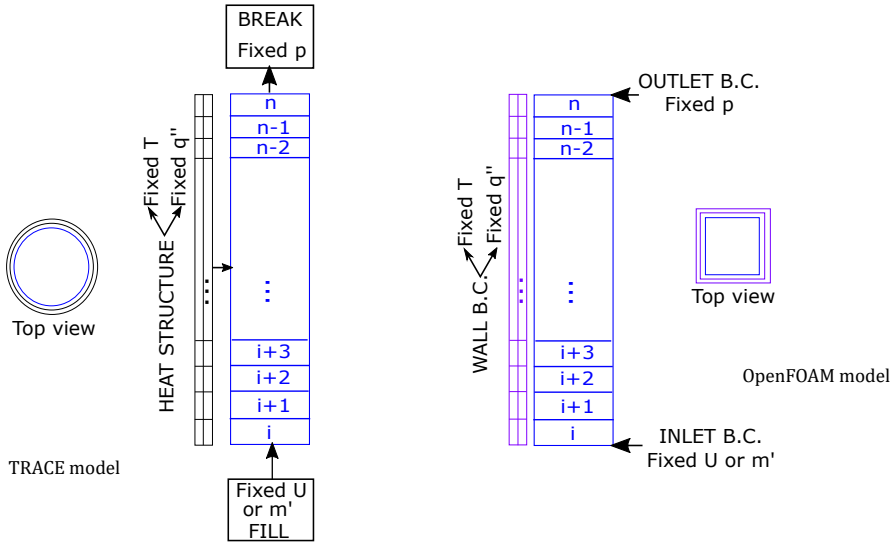


Figure 6.3: Comparison of main elements of TRACE and OpenFOAM models

ular radial nodes. First, a steady state simulation has been executed in order to ensure initial conditions and then a null transient of 100 s has been run. Table 6.1 summarizes the initial conditions for each simulation.

Table 6.1: Initial conditions.

Initial condition	Value	Description
Outlet Pressure (Pa)	2.0725e5	-
Inlet Temperature (K)	294.15	-
Inlet Velocity (m/s)	0.2	-
External heat (W)	1.0e5	Wall B.C Case 1
External wall temperature (K)	375.0	Wall B.C Case 2

Regarding to the solid, it is worthy to mention that the properties are user-defined (section 5.3.3) and the solid material selected for this case is an stainless steel 316. Stainless steel is a material frequently used in nuclear installations, so it is important to know its capability to heat transfer. The reference for this material can be found in Kirillov.

This first part aims at the verification of the solid-liquid heat transfer model and the use of the steam tables application, as well as the single-phase closure equa-

Table 6.2: Temperature-dependent solid properties for single-phase flow simulation (Kirillov, 2008).

Temperature (K)	Conductivity ($\mathrm{W\,K^{-1}\,m^{-1}}$)	Heat Capacity ($\mathrm{kJ\,kg^{-1}\,K^{-1}}$)	Density ($\mathrm{kg\,m^{-3}}$)
300	14.0	502.0	7954.0
400	15.15	516.0	7909.0
500	17.1	529.65	7864.0

tions, so the objective is focused on the verification of the fluid properties and the wall temperatures for a single-phase flow computation. Two different cases are presented so that the heat flux supply can be analyzed, one case with constant external heat flux boundary condition and a second case with a Dirichlet boundary condition (Fixed outer wall temperature). First, the latter case is presented.

6.2.1 Fixed wall temperature boundary condition

Focusing on temperature, figure 6.4 shows the axial evolution of the fluid temperature. Red line corresponds to the new *my1DTPFoam* results, whereas the blue line describes TRACE behavior. This pattern is repeated in all the presented cases. Figure 6.5 shows the axial evolution of the solid temperature at the surface which is in contact to the fluid. Both images show good agreement between codes. Wall temperature presents an average difference of 0.05 K which is constant along the pipe, getting a maximum difference of 0.07 K at the outlet section. Figure 6.6 pressure axial evolution, which also exhibits a good agreement between codes, getting a maximum difference of 13 Pa at the inlet region.

6.2.2 Fixed heat flux boundary condition

Regarding to the case with fixed external heat flux, figures 6.7 and 6.8 presents the comparison of the axial evolution of the liquid and wall temperatures. In this case, the maximum temperature difference at the fluid-solid interface between codes is 0.21 Kelvin and it is found at the inlet, but this difference decreases as the fluid reaches the outlet. The general RMS for this parameter is 0.1 K. Regarding to the fluid temperature, both temperatures agree and follow same evolution along the pipe, having an RMS less than 0.1 K.

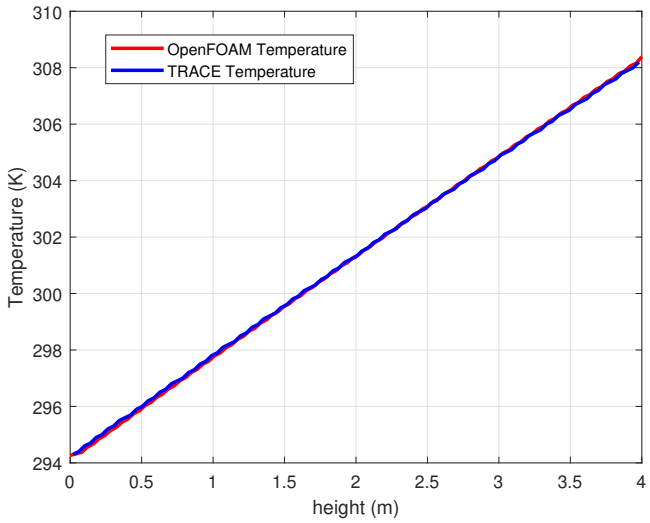


Figure 6.4: Liquid temperature axial evolution.

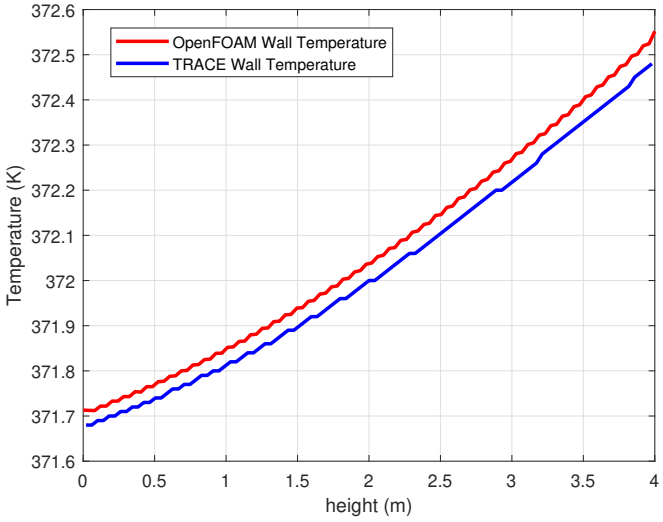


Figure 6.5: Innerwall temperature axial evolution.

Considering both fixed temperature boundary condition and fixed heat flux boundary condition cases, it can be seen that closure models related to fluid works fine,

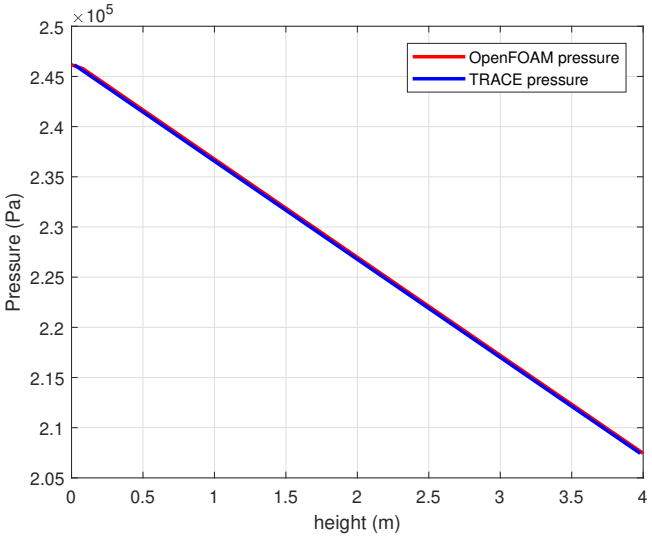


Figure 6.6: Pressure axial evolution.

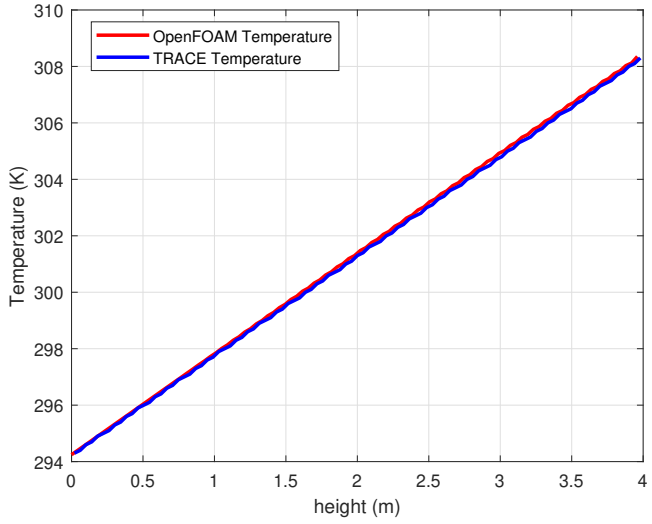


Figure 6.7: Liquid temperature axial evolution.

as well as the acquisition of fluid properties from tables, as shown in figure 6.9,

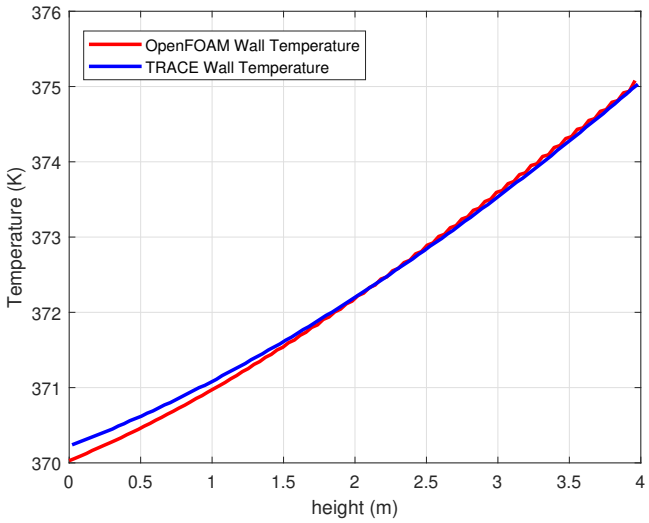


Figure 6.8: Wall temperature axial evolution.

where the comparison of one of the properties taken from steam tables, in this case fluid density, presents a high agreement.

However, there is a slightly difference in the temperature at the interface between solid and liquid. In this point, three variables are contributing, the heat transfer coefficient, the solid properties and the geometry of the pipe. The solid properties can be discarded, since they are equally defined in both codes. Therefore, the temperature small variation may be caused due to a discordance in the heat transfer coefficient or due to discrepancy in the transformation performed between geometries. However, energy conservation is accomplished, since the change in enthalpy in the fluid meets the amount of heat provided to the fluid. Therefore, the variation in the wall temperature at the inlet comes from a variation in the wall heat transfer coefficient model. This model shall be analyzed to find the origin of this difference. Overall, the results can be accepted since the wall temperature difference, which presents the maximum deviation, is irrelevant.

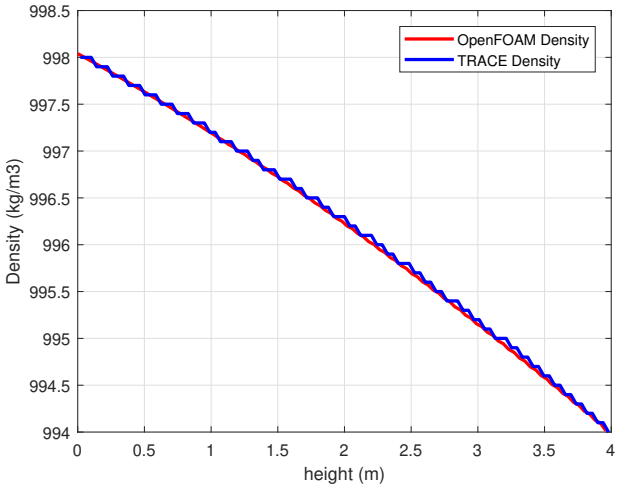


Figure 6.9: Fluid density axial evolution.

6.3 Two-phase flow simulations

In order to check the correct implementation and to verify the two-phase solver, different test cases were used. Some of these were also used as a validation of the system, since the experimental results are also available for comparison with the simulated ones.

In this case, it is necessary to ensure the correct application of different phenomena, such as the interfacial drag force, the interfacial mass transfer and the heat transfer with the solid. It is also necessary to verify the capability of the subcooled boiling model. Since it may happen that not all of them are influencing with the same intensity in every case, it was necessary to use different testcases.

The cases which are going to be introduced are the following: First, an air-water flow system is tested in order to verify the drag force as well as the pressure drop and its influence in the expansion of the air bubbles. Finally, water flow which is heated enough to generate subcooled boiling is presented, in order to test the capability of the solver to simulate subcooled boiling and mass transfer. In this case it is also important the heat transfer through the wall. This routine was already tested in the single-phase case, but the condition at the wall may cause larger gradients with the presence of vapor, so it is necessary to ensure a correct heat balance between fluid and solid.

It is worthy to remind that this cases are simulated for bubbly and slug regime conditions, which is the regime in which this work is focused.

6.3.1 PW-serie experiment

One of the most important aspects in the prediction of heat transfer between two different phases is the interfacial area calculation. This parameter influences the interfacial forces set in general, where it is necessary to determine precisely the interfacial area, but in heat transfer matters, it becomes crucial. The interfacial area is proportionally dependent to the bubble diameter, which also requires an accurate prediction since it affects to the drag force.

Due to the necessity of a correct bubble diameter and the corresponding interfacial area for accurate predictions, different research groups are trying to assess a mechanistic model to define these parameters, to avoid discrepancies between empirical correlations. With the aim of extending the existing information in this field, the work carried out at the Universitat Jaume I in Castellón was taken as a reference. The development of this work can be found in Peña-Monferrer et al. and Peña-Monferrer et al. In this work, only one of the experiments undertaken in the previous references is used.

Particularly, the work presented in this section is the extension of the work presented in (Peña-Monferrer et al., 2018c). The cited work analyses the capability of the one-dimensional code RELAP5 to predict the behaviour of a bubbly flow under low velocity conditions and it proposes a new drag model that considers the bubble size distribution to improve the accuracy of the results. In order to perform the analysis, this work compares the simulation results to the data obtained in an experimental facility located at the Laboratory of Hydraulics of the Universitat Jaume I that consisted in a vertical pipe of $0.052\,\mathrm{m}$ of diameter and $5.5\,\mathrm{m}$ height. Measures were taken at three axial levels of the pipe, z/D = 22.4 (bottom), z/D = 61.0 (middle), and z/D = 98.7 (top). The technique used to obtain the data was Laser Doppler Anemometry (LDA), which is an optical technique to measure velocity and turbulence distribution in both free and internal flows. Details of the experiment and the methodology followed can be found in Monrós-Andreu et al., 2013 and Monrós-Andreu et al., 2017.

The experiment consisted in a water-air fluid running upwards through the pipe. Only the case PW05003 was reproduced in this work and the conditions for the simulation are shown in table 6.3.

The model created in OpenFOAM consisted in a rectangular geometry where the side of the square (similar to the pipe presented in figure 6.1) is taken to keep

Table 6.3: PW05003 flow conditions.

Pressure$_{outlet}$ (Pa)	Vapor Velocity$_{inlet}$ ($\mathrm{m\,s^{-1}}$)	Void fraction$_{inlet}$ (-)
107157.8	0.818	0.0338

the fluid volumen equivalent to the real pipe. The model is $4\,\mathrm{m}$ height (which represents the region where measurements are taken, so it was normalized) and it was divided into 99 axial nodes. This value of axial levels was chosen because the balance between computing time and results accuracy was good enough. Furthermore, this is the maximum number of axial nodes allowed by RELAP5y esta , so using this number also helped to a better comparison of the results. This model does not need solid mesh, since there is no heat transfer from the pipe. In table 6.4, the different B.C defined and the set value is shown.

Table 6.4: PW05003 Fluid Boundary Conditions.

Parameter	Patch	Boundary Condition	Value
Liquid Velocity ($\mathrm{m\,s^{-1}}$)	Inlet	fixedValue	0.5
	Outlet	ZeroGradient	-
	Wall	FixedValue	0
Vapor Velocity ($\mathrm{m\,s^{-1}}$)	Inlet	fixedValue	0.8
	Outlet	ZeroGradient	-
	Wall	FixedValue	0
Pressure (Pa)	Inlet	ZeroGradient	-
	Outlet	FixedValue	1.07e6
	Wall	ZeroGradient	-
Temperature (K)	Inlet	FixedValue	294.25
	Outlet	ZeroGradient	-
	Wall	ZeroGradient	-
Void fraction (-)	Inlet	FixedValue	0.0338
	Outlet	ZeroGradient	-
	Wall	ZeroGradient	-

The simulation consisted in a null transient simulated during 50 seconds. It is also worthy to mention that the drag model used for this simulation was the drag correlation approach (DCA, section 3.7.1) in all codes presented in this comparison. Results obtained with TRACE and RELAP5 are presented in previous works, particularly in Gómez-Zarzuela et al., 2017a and in Peña-Monferrer et al., 2018c, respectively.

The results presented compare the simulation performed by *my1DTPFoam* to the prediction made by TRACE and RELAP5. These results are also compared to the experimental data, in a direct way to undertake verification and validation at the same time.

Figure 6.10 presents the axial pressure evolution along the pipe. In this figure, red, blue and black lines, which are the simulated data, follows a linear trend, starting and finishing from a very close value to the experimental data. However, it can be noticed that the pressure drop in *my1DTPFoam* solver is sligthly greater than in the system codes. This leads to a maximum pressure difference found at the inlet with a value of 0.23 kPa, which is found between TRACE and *my1DTPFoam*. The RMS more critical is again found between TRACE and the new 1D solver, and it is 0.91 kPa. Nonetheless, the pressure drop along the pipe is physically realistic, and the results can be accepted.

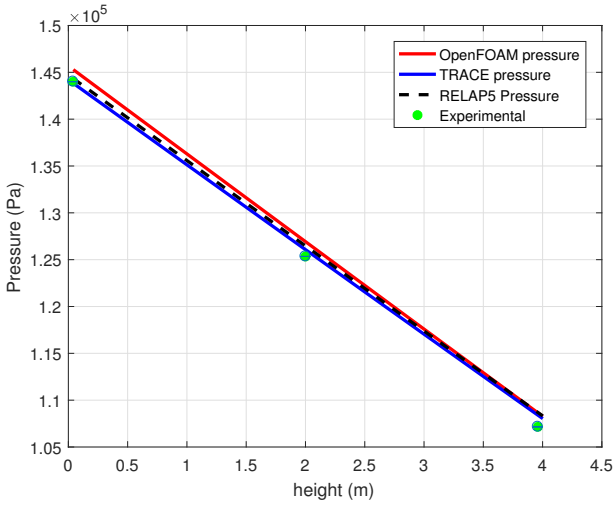


Figure 6.10: Pressure axial evolution in PW experiment case.

Figure 6.11 shows the axial evolution of the dispersed phase velocity. At first sight, the figure presents a constant evolution in all cases. The main difference among them is the axial tendency, where, while the experimental results and the new solver have a very little monotone decreasing slope, TRACE evolves increasing moderately its velocity in the axial direction. RELAP5 also presents an increasing evolution, but the results are lower and the value at the outlet is outside the acceptance interval. The increasing behavior is physically more correct, due to the pressure drop seen in figure 6.10. Looking at the scale at the maxi-

mum difference between the results, it is found at the outlet a variation between results of 0.024 m/s. Focusing on the experimental data, the difference between inlet and outlet is 0.0049 m/s and regarding to the new solver this difference is 0.0069 m/s, so they can be assumed as constant trends. In all cases, the obtained results are within the acceptance interval defined by the error bars with the one exception of the RELAP5 outlet velocity, and *my1DTPFoam* is specially close to the measured values.

The experiment was performed almost at atmospheric conditions, so the pressure drop is very little and its influence in the velocity almost imperceptible. The remaining point lies in the difference between TRACE and *my1DTPFoam* trends, whose RMS is 0.019 m/s. Since all these solver are using the same drag model, one possible term that might influence the axial evolution is the compressibility term, which is calculated differently in the codes, but looking at void fraction calculations, this option may not be the main reason for the difference in the velocity.

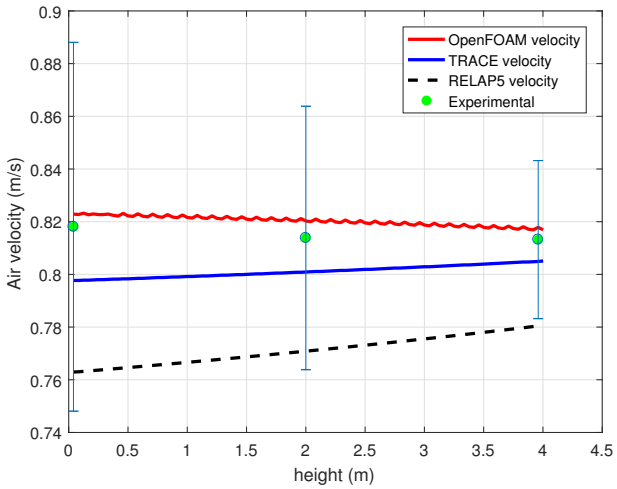


Figure 6.11: Dispersed phase velocity axial evolution in PW experiment case.

Last parameter measured in the experiment is the void fraction. Figure 6.12 shows the results obtained experimentally and by the different programs. Regarding to the experimental results, the image shows the grow up of the void fraction, which, since there is not a heat source to favour a mass exchange, this void fraction increase is caused by the expansion of the bubbles. The void fraction evolution predicted by *my1DTPFoam* starts from a better agreement to the

experimental values than the system codes, even though all simulations predict the same behavior with a maximum difference between RELAP5 and *my1DTPFoam* of 2.4×10^{-3}. In this case, the new solver can be verified, since the calculation agrees with the reference code and the RMS between them is 6.7×10^{-4}, but the axial evolution of both is limited in comparison with the experimental data, where the void fraction measured at the outlet is significantly larger than in the codes.

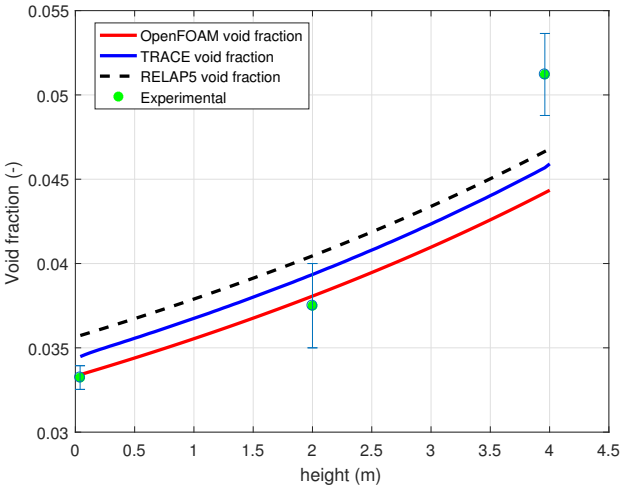


Figure 6.12: Void fraction axial evolution in PW experiment case.

Overall, comparing the axial evolution of velocity and void fraction, one can observe that the compressibility term works properly according to figure 6.12 and the difference in velocity evolution is influenced by a different term. In general, the results of the new 1D solver are physically realistic compared to the experimental results, even though the expansion of the gas is slightly underestimated. Future analysis should be performed about the axial gas expansion.

6.3.2 Bartolomej benchmark

The experiment carried out by Bartolomej G. has been extensively used for validation purposes, examples of this are Krepper, Konar, and Egorov, Corzo et al. or Fu and Anglart works. Therefore, one can find simulations of this experiment performed with a wide variety of programs.

In this work, different programs were used to compare the new solver results for verification purposes. In particular, the results of this work were compared to TRACE and COBRA-TF simulations, which can be considered referent codes in the one-dimensional thermal-hydraulic field. On the other hand, the results of this work were also compared to a three-dimensional solver for two-phase flow called *boilEulerFoam* (Rollins, 2018), which was also developed within OpenFOAM platform and it was previously validated for subcooled boiling for different fluids. Furthermore, this 3D solver was meant as a testing platform for different scenarios, encompassing from critical heat flux situations, DNB calculations and multi-phase flow simulations where boiling occurs. It includes a wide range of interfacial models, allowing the analysis of the influence of these in the final result. It considers the RPI wall heat flux model Kurul and Podowski, 1990 to calculate wall temperature and the fluid properties used are considered constant.

All the simulations presented in this work were performed by the author of this thesis, also in previous cases. However, regarding to the programs which are not object of this work, only a brief description of each model will be presented. More detail about the work undertaken with *boilEulerFoam* can be found in Gómez-Zarzuela et al., 2017b.

Experimental set up

This experiment was carried out in order to study the void fraction and temperature profiles when boiling water subcooled below saturation temperature flows within a vertical heated pipe (Bartolomej G., 1967). For the benchmark calculation, this experiment was performed in a 2 m long heated tube with a inner diameter of $0.0154\,\mathrm{m}$. The heat flux was $5.7 \times 10^5\,\mathrm{W\,m^{-2}}$ and the mass flow of the water at the pressure of $4.5 \times 10^6\,\mathrm{Pa}$ were $900.0\,\mathrm{kg\,s^{-1}\,m^{-2}}$. In the calculations, the inlet subcooling was set to $58.2\,\mathrm{K}$ (Krepper, Konar, and Egorov, 2007).

OpenFOAM Model

Two different geometries were modeled in order to compare results and computing time, an axisymmetric model and a square pipe, including the solid model in each case. Both models were generated with the OpenFOAM built-in mesher *blockMesh*, which, as seen in section 5.3.1, it is able to generate only hexahedral meshes. Thus, in one-dimensional simulations, were each level corresponds to one only cell, these will be cubes and it is not possible to create cylindrical geometries. The alternative to simulate radial actions using the built-in mesher is the use of an axisymmetric mesh.

On the one hand, first geometry was represented as an axisymmetric model, where only a section of 5 degrees of the circular pipe was modeled. Radial symmetry condition was applied to the sides of the pipe section. In order to mesh the geometry, 50 uniform nodes were implemented in the axial direction, whereas 1 radial node was used and also 1 node was utilized to the azimuthal direction. The solid mesh consisted in a second system, which, following the same distribution as fluid geometry, it was distributed in 50 axial levels and 3 radial nodes. Figure 6.13 shows the described model.

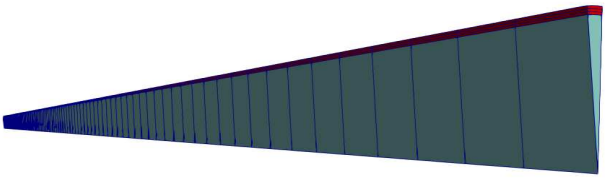


Figure 6.13: Axisymmetric model in OpenFOAM.

Axisymmetric models are known to be faster in time computing than 3D meshes, due to its reduced number of cells, and they have generally less discretization errors in the direction of the circumference, due to the small angle usually used. However, the mesh must be carefully defined, as mentioned in section 5.2.

A null transient of $80\,\mathrm{s}$ was run with an initial time step of $1.0 \times 10^{-4}\,\mathrm{s}$, though this time step is adaptive as long as it satisfies a maximum Courant number of 0.95.

Different cases has been run during the development of this work. First stages did not include subcooled boiling model, so vapor generation started once the continuous phase reached the saturation temperature. However, in this work only simulations with the full solver are presented. In particular, two cases of this benchmark are shown, using different drag correlation models of those which has been implemented (section 3.7.1). First case presented (axisymmetric model) was run using the Drag Coefficient Approach (DCA), while the second case (square model) applies the drift flux correlation. Simulation results are compared with TRACE and COBRA-TF results for the same case.

Axisymmetric model

This simulation was performed using the axisymmetric model. Axial evolution of pressure, liquid temperature, enthalpy and velocity, and void fraction are presented in this section. Only verification was analysed in this case, focusing on one-dimensional thermal-hydraulic codes, in order to compare the capability of the new solver to predict the subcooled boiling in comparison to reference codes in this field. Therefore, the results is this section are presented as a comparison of the outcomes of *my1DTPFoam*, TRACE and COBRA-TF for the same case. For the same reason (thermal-hydraulics analysis), wall temperature is not included in this section.

Figure 6.14 shows the axial evolution of the pressure for the different codes. In this image one can see that pressure drop follows the same trend in all simulations, although *my1DTPFoam* provides the lower values. However, the difference with COBRA-TF pressure at the outlet is about 650 Pa, which is a low difference according to the order of magnitude of the pressure, reaching a difference of 1kPa at the inlet. Furthermore, the value of *my1DTPFoam* agrees with the boundary condition defined at the outlet. TRACE values are about 3kPa superior, which considering only the pipe, is not strange, since there is always a pressure drop between the boundary condition (BREAK) and the first cell. The pressure jump between *my1DTPFoam* and TRACE is kept along the pipe, which indicates that a precise adjustment in the boundary condition could reduce the difference between the results significantly. The RMS error between TRACE and the new solver for this variable is 1.44 kPa, which is close to the average difference axially found.

Next, the void fraction axial evolution is presented in figure 6.15. The vapor starts to occur at a height of 1.1 m, which is the beginning of the upper half of the pipe, approximately. All the codes predict the same point, which suggest a correct prediction of the point where boiling starts. For the first part of the upper half the void fraction calculation continues to match among all the codes, being the prediction of COBRA slightly lower. However, the slope in *my1DTPFoam* is lightly steeper, specially from a height of 1.7 m, where the value of void fraction calculated by *my1DTPFoam* starts increasing significantly, having a larger deviation, and reaching a void fraction at the outlet which is a $12\,\%$ higher than the referent codes. In this case, the RMS error between TRACE and *my1DTPFoam* is 0.0118, but the main difference is found at the outlet section. Since the main contribution in this variable comes from the subcooled boiling model, this approach should be reviewed for the regions where the subcooling temperature get closer to 0.

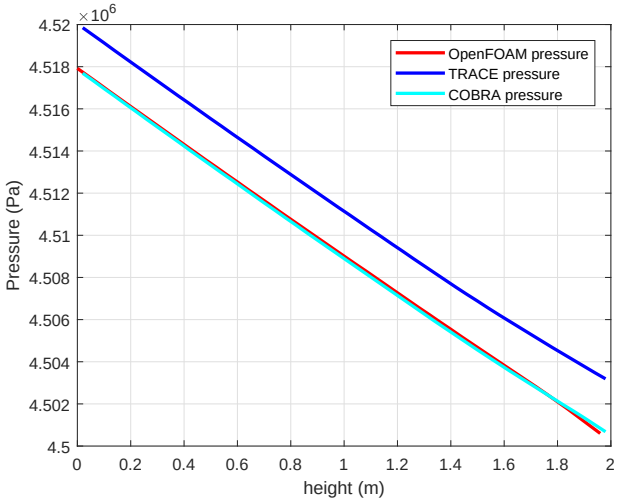


Figure 6.14: Pressure axial evolution at time = 80s.

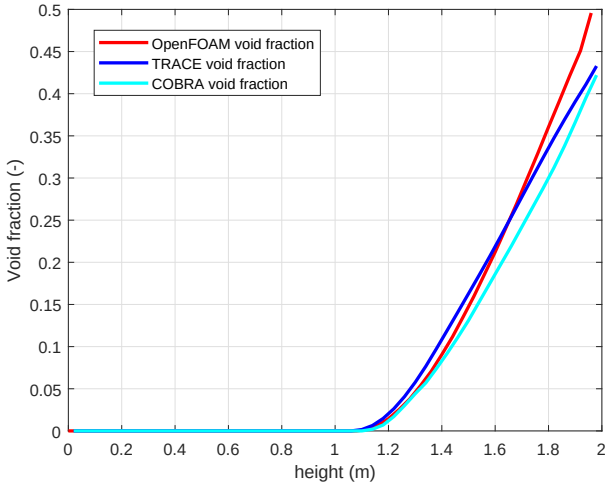


Figure 6.15: Void fraction evolution at time = 80s.

The liquid temperature, which is directly related to the presence of boiling, is shown in figure 6.16. In this case, the comparison is only presented with TRACE, since COBRA-TF provides enthalpies. Hence, the comparison was made separately, using the variable provided by the output of each program. Liquid en-

thalpy is presented in figure 6.17, where both *my1DTPFoam* and COBRA-TF calculations are presented.

Regarding to the liquid temperature, the axial evolution of this parameter agrees in a high level between both codes, as also demonstrated by the RMS error, which is 0.35 K. The main difference can be found at the output section, where according to *my1DTPFoam*, the liquid reaches saturation temperature, while in TRACE this value is always below saturation conditions.

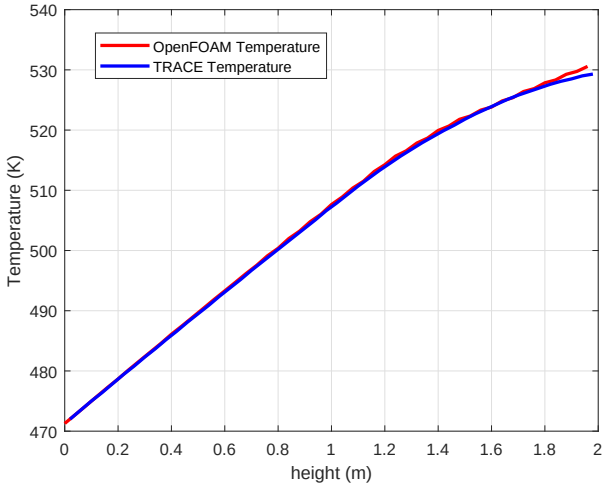


Figure 6.16: Liquid temperature evolution at time = 80s.

On the other hand, if one looks at the enthalpies, the behaviour is very similar to the temperatures. Both trend and values of the codes agrees significantly, being the enthalpy in *my1DTPFoam* slightly higher, also at the outlet section. In this case, the RMS error is 2.81 kJ/kg.

In general, this agreement in the temperature/enthalpy does not correspond to the excess in void fraction at the upper region of the pipe in *my1DTPFoam*. The subcooled boiling model should be analyzed deeper for regions where the fluid temperature approaches to saturation. Furthermore, the model to evaluate the mass transfer term, such as interfacial heat, should be revised.

Finally, the axial evolution of the liquid velocity is presented in figure 6.18. The comparison between the three outcomes shows discrepancies, specially in the case of *my1DTPFoam*. In the lower half of the pipe, where the fluid is single-phase, the liquid velocity remains equal for the different simulations, showing

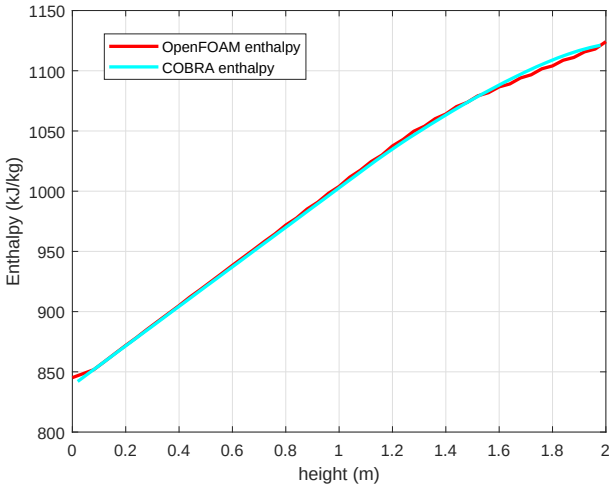


Figure 6.17: Liquid Enthalpy evolution at time = 80s.

an increasing monotone behavior. Then, when the fluid runs through the upper half of the pipe and the fluid becomes two-phase, there is a change of behavior, running faster in *my1DTPFoam* than in the reference codes from a height of h = 1,5m. This difference may come from the interfacial drag force, being more influential than it should in the new solver. Furthermore, the steeper growth in void fraction may also affect to the velocity, giving an excess of relevance to the vapor speed, which, in turn, will influence the void fraction, due to the feedback between these variables.

Summarizing the different results that have been presented, the capability of the solver to simulate thermal properties has been verified. However, the void fraction simulated by the new solver presents a discrepancy with respect to the reference codes. One reason could be the subcooled boiling with fluid temperature close to saturation conditions, originating a greater amount of subcooled boiling. Besides, the higher liquid velocity may also influence the void fraction, since it is involved in the mass continuity equation.

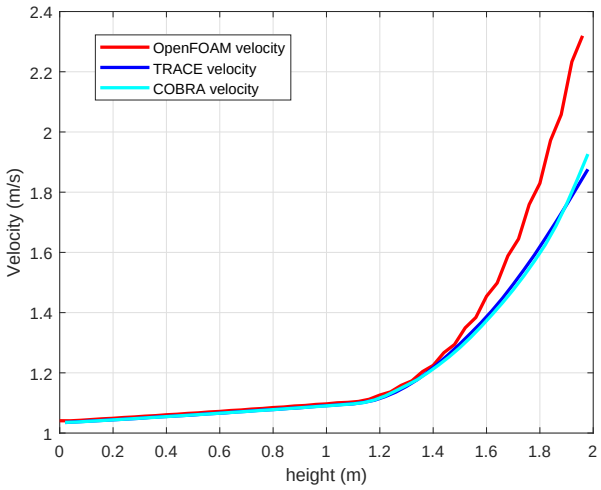


Figure 6.18: Liquid velocity axial evolution at time = 80s.

Square model

The second model consisted of a square pipe of 50 axial levels for both solid and liquid models. This geometry was defined considering same height of the cylindrical pipe and same cell volume. Therefore, in order to preserve the equivalence between the two geometries, the volume was conserved. In order to satisfy volume conservation, the determination of each edge was undertaken by carrying out the following calculation. Assuming same cell volume, the next equality must be satisfied

$$V_{cyl} = V_{sq} \rightarrow \pi * r^2 * h = l^2 * h,$$

where r represents the radius of the cylindrical pipe, l is the side of the square and h is the height. Therefore, since the height of the pipe remains constant for both models, the final relation is

$$\pi * \frac{0.0154^2}{4} = 1.86265e^{-4} = l^2 \rightarrow l = 0.013\,647\,9\,\mathrm{m}$$

Regarding to the solid geometry, this was changed from the original experiment in order to apply a discretization big enough to distinguish the values among

cells. Therefore, whereas the original thickness was $0.35\,\mathrm{mm}$, in this work it was decided to give a thickness of $3\,\mathrm{mm}$ to the solid pipe. This geometry surrounds the solid and it was divided into two radial nodes. The final model is shown in figure 6.19.

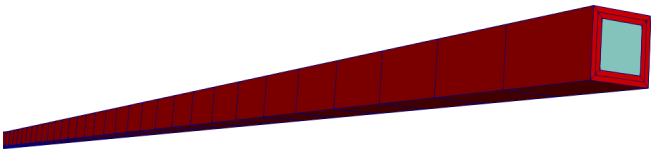


Figure 6.19: Portion of the mesh for Bartolomej benchmark in OpenFOAM.

This model still had a drawback, since the lateral surface of the fluid is not the same as in the cylindrical pipe, and it is necessary to take this fact into account to define the external heat flux boundary condition properly. According to (Bartolomej G., 1967), the heat flux is $5.7 \times 10^5\,\mathrm{W\,m^{-2}}$ for the cylindrical pipe. In order to get equivalent heat for the square model, it is necessary to transform this heat to absolute values (in this case, to Watts), to then transform it to the equivalent heat flux for the square model. Table 6.5 shows the equivalences between surface areas and heat fluxes for square and cylindrical geometries.

Table 6.5: Heat flux equivalences.

Cylindrical Pipe			Square Pipe	
Surface area ($\mathrm{m^2}$)	Heat Flux ($\mathrm{W\,m^{-2}}$)	Total heat (W)	Surface area ($\mathrm{m^2}$)	Heat Flux ($\mathrm{W\,m^{-2}}$)
0.096761	5.7e5	**55154**	0.109183	5.0515e5

The heat flux balance was considered internally, i.e at the volumetric fluid, where the total heat must be the same in both geometries, and in the solid boundary condition, where the external heat flux must be defined accordingly. It does not influence the fluid boundary conditions, as it can be seen in table 6.6, where the list of all boundary conditions for fluid region is shown. In the case of the wall boundary, the *mixed* boundary condition is defined for the temperature. Using

this B.C., the code will take the calculated wall temperature from the solid domain in contact with the fluid and it will assign it as a Dirichlet B.C (fixed value) at the wall of the latter.

Table 6.6: Fluid Boundary Conditions.

Parameter	Patch	Boundary Condition	Value
Velocity (l and v)	Inlet	flowRateInletVelocity	0.1676385
	Outlet	ZeroGradient	-
	Wall	FixedValue	0
Pressure	Inlet	ZeroGradient	-
	Outlet	FixedValue	4.5e6
	Wall	ZeroGradient	-
Temperature	Inlet	FixedValue	471.25
	Outlet	ZeroGradient	-
	Wall	mixed	Code calculated
Void fraction	Inlet	FixedValue	1e-20
	Outlet	ZeroGradient	-
	Wall	ZeroGradient	-

On the other hand, table 6.7 shows solid boundary conditions. The solid region patches were divided into three groups: internal wall, external wall and the rest of surfaces, called *top & bottom walls*. The latter have no impact over the system, so their value depends on the nearest cell. The internal wall was assumed to be in contact to the fluid and will work as a *mixed* boundary condition (see section 3.11) and the external wall is defined as a *externalWallHeatFlux* B.C, which means that it gets the heat flux from an external source (section 4.3.3). In this work, this B.C. was defined as *Total heat* mode, which means that the user defines the absolute value of heat (in W).

Table 6.7: Solid Boundary Conditions.

Parameter	Patch	Boundary Condition	Value
Temperature	Top & Bottom walls	ZeroGradient	-
	InternalWall	mixed	- (code calculated)
	ExternalWall	ExternalWallHeatFlux	62234.4 (W)

The value of the *externalWallHeatFlux* B.C does not match the calculation given in table 6.5. In this case, the value of total heat is given so that the final heat flux of the patch is $5.7 \times 10^5\,\mathrm{W\,m^{-2}}$. The reason for this change lies in the fact that this simulation pretends to get results for the square solid geometry equivalent to those obtained by a cylindrical pipe with same thickness. However, the lateral surface of these geometries are different, so the heat flux needed to reach the same heat at the fluid volume changes, as seen before. Consequently, if less heat flux is needed to transfer the same heat (because the area in contact with the fluid is bigger), the lateral surface does not need to get so hot and its temperature remains lower. Therefore, the results obtained by this solver cannot be the same to the obtained by a solver that works with the values of a cylindrical geometry.

Therefore, in order to demonstrate the proper work of the solid calculation by showing that the same temperature is reached for same boundary conditions and same thickness, the absolute heat equivalent to a heat flux of $5.7 \times 10^5\,\mathrm{W\,m^{-2}}$ was given. It is also worthy to mention that the value given in table 6.7 was calculated considering that the heat is transferred to the external solid surface, which has a larger area than the inner.

The previous justification is valid since the conduction heat transfer is calculated only in radial direction. Thus, in order to calculate the heat transfer at each level, only the external and internal B.C and the own cells of the particular level are considered. Considering the same B.C values and same thickness, the value obtained at the internal wall should be the same regardless the geometry, in a radial calculation.

Regarding the solid geometry, it remains to define the solid properties. The benchmark does not give information about the solid, apart from the material, which is stainless 1CR18Ni9Ti steel. In TRACE solid material database, one can find the Constantan material, which is frequently used in Nichrome coils, used to electrically heat nuclear fuel-rod simulators (NRC, 2013). Therefore, Constantan was used in order to perform these simulations. From the different empirical correlations used to obtain the properties of this material, the values for conductivity, heat capacity and density presented in table 6.8 were calculated. Same values were introduced in both TRACE and *my1DTPFoam* codes.

Finally, in aims of a possible reproducibility, the different algorithms to solve the matrix systems (called *Linear Solvers* in OpenFOAM) are presented here. However, the analysis of the most suitable linear solver is beyond the scope of this thesis. Table 6.9 shows the different algorithms used in this simulation.

Table 6.8: Temperature-dependent solid Properties.

Temperature (K)	Conductivity ($\mathrm{W\,K^{-1}\,m^{-1}}$)	Heat Capacity ($\mathrm{kJ\,kg^{-1}\,K^{-1}}$)	Density ($\mathrm{kg\,m^{-3}}$)
450	29.5	371.0	8393.0
550	30.33	404.33	8393.0
650	30.1	429.61	8393.0

Table 6.9: Linear solvers

System	Linear Solver
Void fraction	PBiCG
Pressure	GAMG
Velocity	symGaussSeidel
Enthalpy	symGaussSeidel

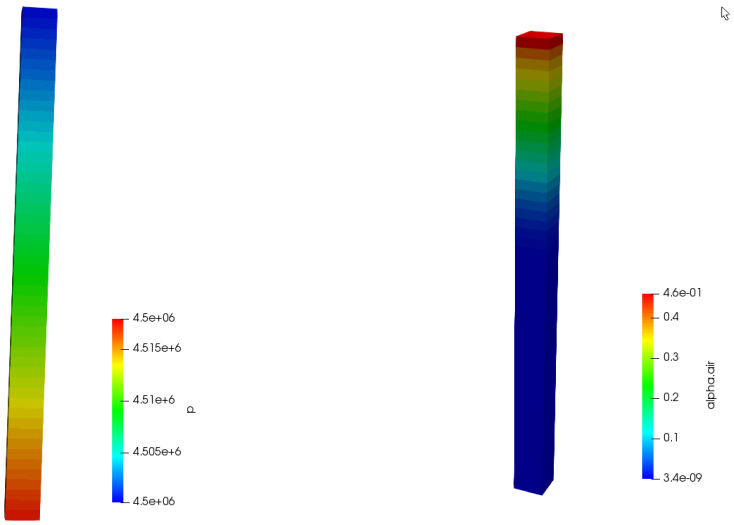
Furthermore, the PIMPLE algorithm was treated as a pure transitory with 2 pressure corrector and 1 outer corrector iterations. Regarding to the relaxation factors, the field *pressure* was given a relaxation factor of 0.3 and the heat transfer coefficient a value of 0.1. The equations of enthalpy and velocity (predictor) were relaxed with values of 0.7 and 0.3, respectively.

The simulation consists in a null transient of 50 seconds with an adaptive time step that satisfies a maximum Courant number of 0.95.

The results of this simulation are presented as a comparison to the results obtained with two different solvers and the experimental data, for verification and validation of the solver. Specifically, the two solver used for comparison are TRACE (NRC, 2013) and a solver developed by the NCSU within OpenFOAM platform for three-dimensional simulations (Rollins, 2018). The first was chosen due to its extended validation in the international scientific community, while the second was selected due to its similarity to the new solver presented in this work and its previous validation, which can be seen as a guide for this work. The model in this solver consisted of an axisymmetric geometry of 200 axial uniform cells, 25 divisions radial direction and 1 azimuthal cells. It is worthy to remark that *boilEulerFoam* results are averaged over each axial level, to get only one value per level.

Results shown in figures 6.20, 6.21, 6.23, 6.24, 6.25, 6.26 and 6.27 represent last step of the simulation, so the results are presented for time = 50s.

First, results obtained by the developed solver are presented in figures 6.20 and 6.21. Here, one can see the main variables of the case represented in *Paraview*, which is a free graphical interface that allows to analyze OpenFOAM data. In particular, pressure, void fraction and liquid and wall temperature are presented. All of them presents a realistic behavior according to the conditions of the case.

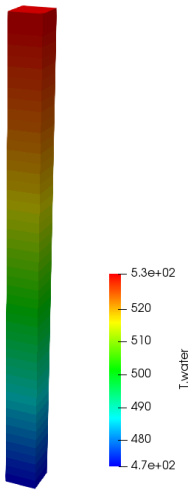


(a) Pressure

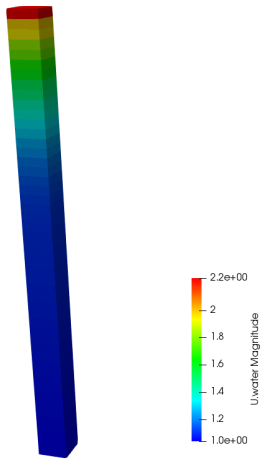
(b) Void fraction

Figure 6.20: Void fraction and pressure axial evolution for time t=50s.

Next, the comparison between different codes is presented. Figure 6.23 shows the axial evolution of the water temperature in its liquid phase for the experiment and for the different programs used for this verification. Green dots are the experimental results, blue line represents TRACE prediction, pink dashes shows the three-dimensional calculation and red line represents the new solver. Overall, all results follow the same trend, with slightly differences. Regarding to the verification, starting from the inlet, the values for the lower part of the geometry, which corresponds to the single-phase part, meets for all cases, although the results for the 3D code evolves with lower results until it reaches the subcooled portion of the pipe. From that moment, the temperature increases faster and it reaches saturation condition before the rest of cases. Regarding to *my1DTPFoam*, the evolution matches the prediction of TRACE until the last two cells, where it reaches saturation conditions while TRACE remains lower. In general, the agreement between these both curves is high except for the outlet section, where the maximum difference is reached with a value of 1.3 K.



(a) Liquid temperature



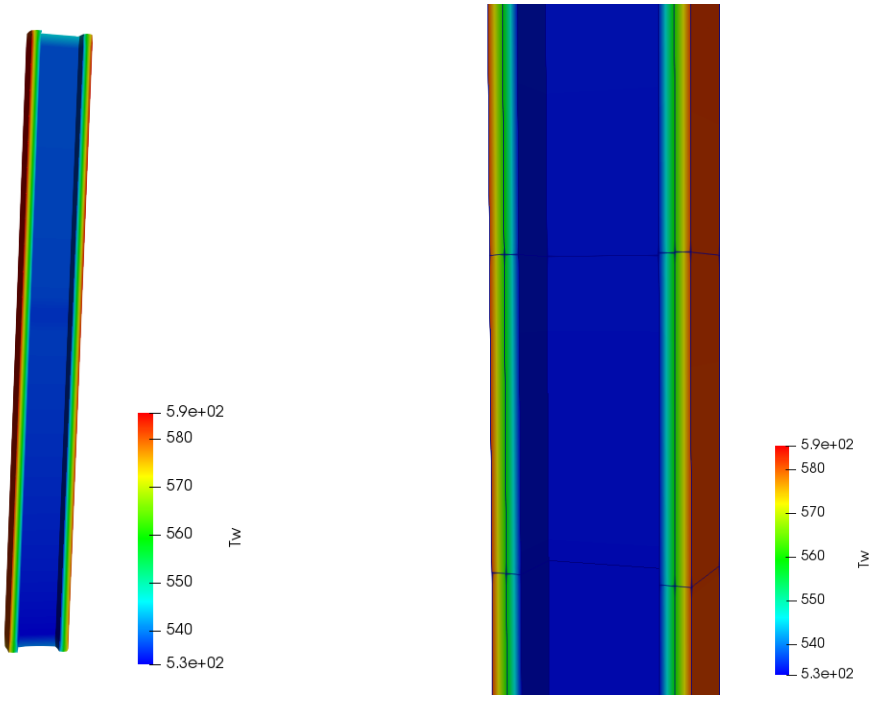
(b) Liquid velocity

Figure 6.21: Liquid temperature and velocity for time t=50s.

Regarding to validation, comparing *my1DTPFoam* to the experimental values, it can be seen that simulated temperature is slightly higher than experimental during the first two thirds of the pipe. The maximum difference along this region is 0.3 K. In the last third, the differences get the maximum value of 2.7 K, but at the outlet section, the final temperature, which is saturation temperature, is reached by both results.

Figure 6.24 shows the results for void fraction parameter. In this figure, the experimental results together with the predictions of TRACE, *boilEulerFoam* and *my1DTPFoam* are presented so that they can be compared. Following the same order as in figure 6.23, regarding to verification, if one compares *my1DTPFoam* with TRACE, one sees that evaporation starts at the same point, which is at the point of height = 1.1 m. Both slopes increase in a linear trend, but *my1DTPFoam* slope is rather steeper, which leads to a higher void fraction at the outlet. However, the main difference between these trends is found at the boiling initiation region, where *my1DTPFoam* does not predicts as much vapor as TRACE, being from that moment, shifted a 2.5% from the reference values.

Analyzing the behavior of the three-dimensional solver, one can see that the evaporation starts earlier, roughly at height = 0.8 m. However, the generation of bubbles is slower at the beginning, so at the last third of the pipe, slope is



(a) Wall temperature

(b) Detail of wall temperature

Figure 6.22: Wall temperature axial evolution and a detail of this result for time t=50s.

very similar to the rest of calculations, with a lower value initially and a steeper slope at the outlet. The value of void fraction at the outlet is even larger than *my1DTPFoam* value.

Comparing the simulated results to the experimental, and focusing on the new solver results, it is clear that void fraction in *my1DTPFoam* is generally lower than expected, until the outlet region, where the slope changes and the void fraction increases rapidly. This last change in the behavior is related to the saturation temperature reaching, as well as the velocity increase, as seen in previous case.

It is also remarkable that, since the fluid reach the saturation temperature at the height of 1.8 m, the vapor generated up to this point is due to subcooled boiling. Therefore, the subcooled model implemented in *my1DTPFoam* is able to predict the bubble generation, although this results show that the calculation is underestimated. Special attention should be paid to the boiling initiation region, where the difference are also meaningfully.

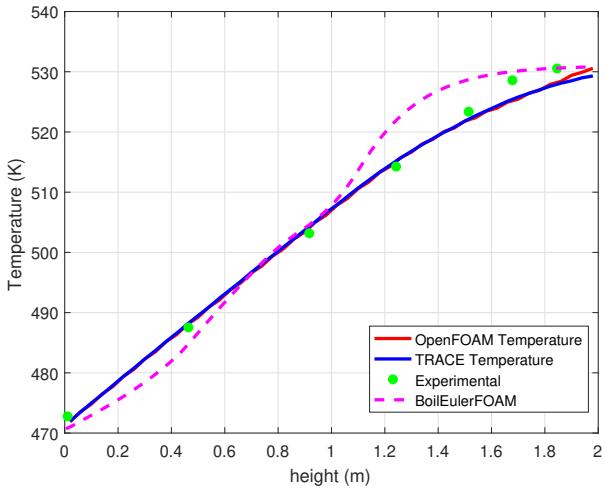


Figure 6.23: Axial liquid temperature evolution for bartolomej benchmark simulated by different programs.

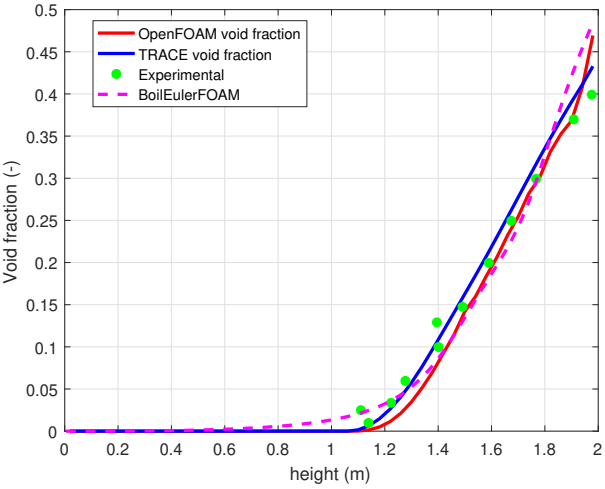


Figure 6.24: Axial void fraction evolution for Bartolomej benchmark simulated by different programs.

Last variable measured in the experiment was wall temperature. The results for this parameter are shown in figure 6.25. Again, all results are shown in the same

graphic to get an overall view. Starting by the verification, it can be seen that wall temperature for TRACE and *my1DTPFoam* simulations match along the pipe. If the simulation of *boilEulerFoam* is added to the comparison, the input of the pipe starts from a colder point, but it heats faster and most of the pipe has a higher temperature.

Similar trend happens when comparing with the experimental result. The experimental values and the three-dimensional solver show a hotter surface than the one-dimensional codes, even though the value approaches whereas move upwards. At the outlet, the temperature difference between *my1DTPFoam* solver and the experimental value is 1 K. However, the differences at the lower part of the pipe are bigger, reaching a maximum of 5 K. In addition, the wall temperature at the second part of the pipe tends to decrease, while in *my1DTPFoam* is increasing monotone along the whole geometry.

As a summary of the validation comparison, one can see that fluid values present a significant agreement in general, although particularly the void fraction is lightly underestimated in *my1DTPFoam* solver than in the measured results. According to the solid results, the wall temperature is also generally lower in the computation, getting the main difference at the bottom region of the pipe. This implies a lower dissipation of heat in the reality than the simulated, where more heat is transferred to the fluid and therefore, the wall temperature remains lower. However, the fluid prediction matches the measured temperature, so the heat found at the fluid is equivalent. Furthermore, subcooled boiling model is able to predict the point where the bubbles generation starts, but the values of void fraction predicted are slightly lower, with the exception found at the outlet section.

In order to provide support to the previous results, additionally information is presented for a better understanding of the predicted results. In particular, pressure and velocity evolution calculated with for *my1DTPFoam* and TRACE codes are presented.

Figure 6.26 presents the axial evolution of the pressure. In this case, both simulations follow the same trend, which can be translated in the same pressure drop calculated by the programs. Wall friction does not influence this case in a significant way, so it may be said that the pressure drop is basically due to the body forces.

Finally, last parameter presented is liquid velocity. This variable is shown in figure 6.27. This result presents various differences between solvers. During the first half of the pipe, while the fluid is liquid, the velocity of both codes is equivalent. Once the fluid becomes two-phase, two different behavior are shown.

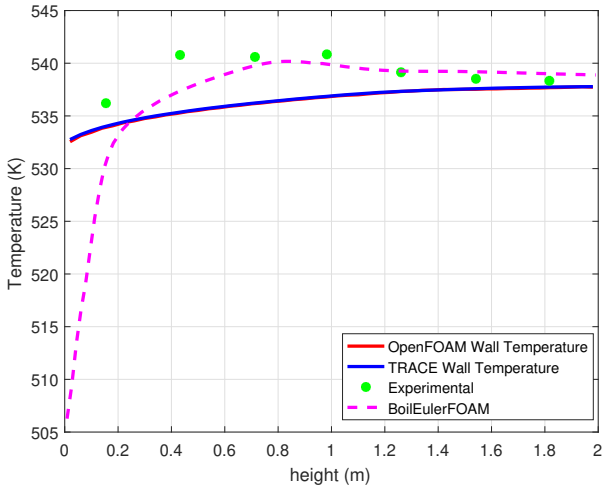


Figure 6.25: Wall temperature axial evolution for Bartolomej benchmark simulated by different programs.

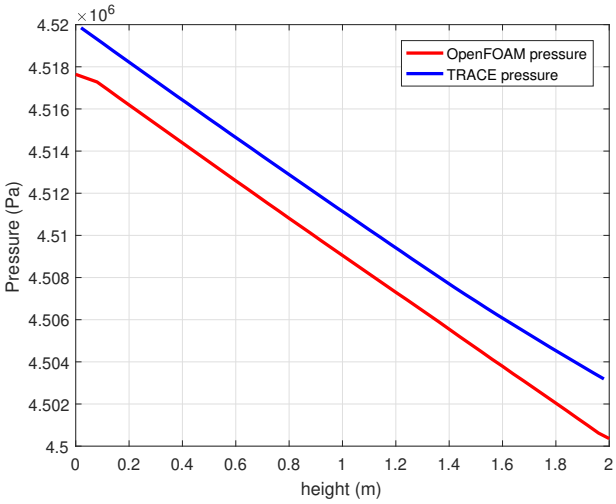


Figure 6.26: Axial pressure evolution for Bartolomej benchmark simulated by OpenFOAM and TRACE.

First, the slope in *my1DTPFoam* simulation is lower than in TRACE. This trend progressively changes with the slope of *my1DTPFoam*, which becomes steeper

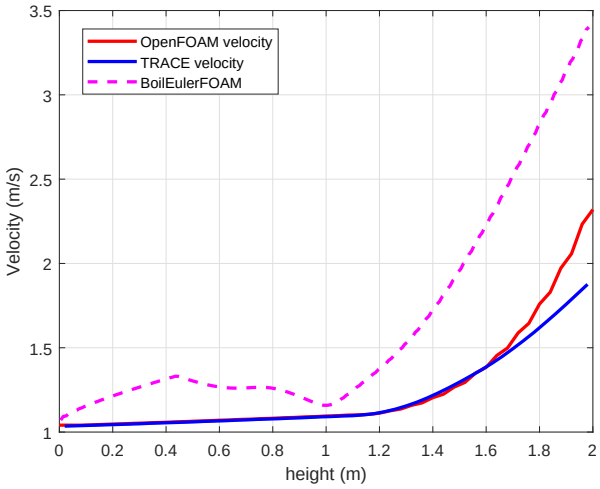


Figure 6.27: Liquid velocity axial evolution for Bartolomej benchmark simulated by OpenFOAM and TRACE.

with the height. The maximum difference is found at the outlet section, where *my1DTPFoam* velocity is a $15\,\%$ higher.

In general, 3D solver numerical solution has a better agreement with experimental data, especially in temperature parameter. However, the mesh is 50 times bigger, consequently it consumes more resources. On the other hand, the results obtained by the 1D solver agrees in general with the experimental data and the system code, consuming less resources. Therefore, the new code presents an alternative to reduce time and obtain correct results in those geometries or portion of geometries where the accuracy on the results in not essential. Hence, this new solver meets the conditions to be coupled to a 3D solver optimizing the computational resources.

Bartolomej case with fixed temperature boundary condition

Besides the validation case, a different case using the Dirichlet boundary condition at the external wall was also simulated. The value of the temperature fixed at the boundary was chosen so that the heat flux provided to the fluid was kept equivalent to the experimental case. Table 6.10 presents the solid B.C for this case. The rest of the conditions remained the same as in the previous case. This performance allows disregarding different potential parameters of failure in or-

der to revise the solver and find the causes of the difference in the results respect TRACE and the experimental values, specially in the void fraction variable.

Table 6.10: Solid Boundary Conditions.

Parameter	Patch	Boundary Condition	Value
Temperature	Top & Bottom walls	ZeroGradient	-
	InternalWall	mixed	- (code calculated)
	ExternalWall	FixedValue	589 (K)

Figure 6.28 shows the axial evolution of the temperature for the last time step. The simulated results are compared to the experimental values, since the equivalent heat flux provided to the fluid remains the same as in the previous case. The evolution between computer codes agrees in a high level, up to a height of 1.8 m, where the fluid in the new solver is overheated, reaching a maximum difference of 1.05 K. This overheating is perceive along the whole pipe, which implies a that the solid is dissipating more heat. However, focusing on the outlet section, one can see that the fluid, according to the experiment, reaches saturation temperature, as happens in the new 1D solver.

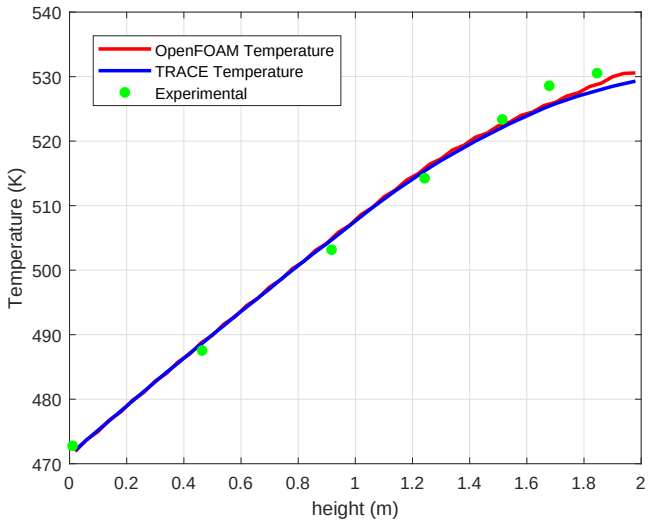


Figure 6.28: Liquid temperature axial evolution with Dirichlet B.C.

Regarding to the behavior found at the surface between liquid and solid, which is shown in figure 6.29, the results calculated by both codes agree in a high level, showing a RMS of $0.045\,\mathrm{K}$. However, when they are compared to the experimental wall temperature, the behavior shown is different, since the wall temperature at the device reaches higher values. This means a larger dissipation of heat in the codes, which do not correspond to the experiment, so the wall heat flux model should be analyzed and compared to different models to look for the most accurate correlation.

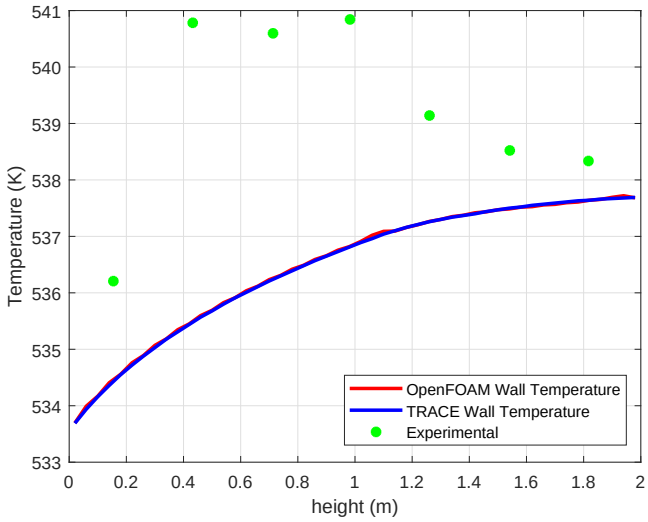


Figure 6.29: Wall temperature axial evolution with Dirichlet B.C.

Lastly, figure6.30 shows the void fraction axial evolution. It can be seen that the new 1D solver rather under predicts this variable, which somewhat differs with the rest of parameters, where the variables are properly adjusted or rather over-predicted. The higher the temperature in the fluid, the higher the vapor generation should be. In this case, the over-prediction can be found at the outlet section of the pipe, when the fluid reaches saturation temperature. Similar behavior can be found when the new 1D solver is compared to the experimental values, which rather approaches to TRACE calculations. The computer calculations are under-predicted except for the outlet section, where the simulation changes the pattern and increases significantly, becoming higher than the experimental value.

Overall, it can be concluded that the case with a fixed temperature boundary condition provides good results in general in comparison to TRACE. This means that closure models are suitable for this simulations and the solver in general can be verified. Subcooled boiling model and mass transfer term in continuity equation should be deeply analyzed, since the differences in the case with fixed heat flux boundary condition could be a consequence of a not adjusted balance.

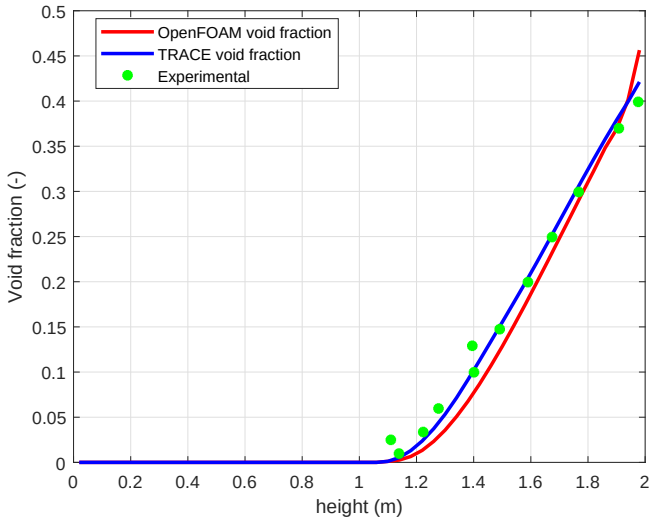


Figure 6.30: Void fraction axial evolution with Dirichlet B.C.

Both cases have a correct trend in all variables, but when the fluid reaches the subcooled nucleate boiling state, there are discrepancies between codes that results in over-predictions in the new solver.

However, as in case with heat flux boundary condition, one can see that the solver presented provides new tools to optimize complex simulations that requires large amount of computational resources by simplifying those parts of the domain where a high precision in the results is not required.

CHAPTER 7

CONCLUSIONS & FUTURE WORK

7.1 Conclusions

This thesis presents the development, verification and validation of a new solver called *my1DTPFoam*. This solver was developed within the OpenFOAM platform and it is meant to one-dimensional simulations for single- and two-phase flows. OpenFOAM library was selected first and mainly, because it is open-sourced. This feature allows the user a full control of the code, and the possibility of modification and improvement of the code. Furthermore, this is a multi-physics platform, since it can perform simulations at different scales among diverse domains.

This work was undertaken in two stages. First, a one-dimensional solver for single-phase flows was developed. The main features of this new solver were: one-dimensional simulations, the use of a wall friction model to consider the axial component of the viscous forces, able to calculate heat transfer effects, and the possibility of adding a solid system to simulate conduction phenomena through it. In order to carry out the work, an OpenFOAM built-in solver was used as a basis and modified to satisfy the new requirements. This document presents the conservation equations used for this implementation as well as the different closure equations used to solve the system. Once the solver was implemented, its verification was performed. Here, a simulation was presented consisting in a 4-meters vertical pipe with a water flow running upwards within it. This prob-

lem allowed the verification of the proper simulation of the pressure drop and the wall heat transfer as well as the liquid and solid temperature distributions. The selected code to verify the new solver was TRACE, since it is considered a reference in one-dimensional thermal-hydraulic simulations. The comparison of the results of both programs shows the capability of OpenFOAM to calculate heat transfer between fluid and solid. Furthermore, the fluid properties were calculated by an external application based on the IAPWS-IF97 steam table that was linked to the new solver and that allows update the properties every time step as a function of pressure and temperature.

The second step consisted in extending the solver to involve two-phase flow simulations. Therefore, a new built-in solver was used as a basis. In this case, the solver *twoPhaseEulerFoam* was chosen. This solver was meant to perform Eulerian-Eulerian calculations for two-phase flows without mass transfer. Thus, in this new stage the main objectives were: performing one-dimensional two-phase flow simulations, considering interfacial mass transfer, and being able to predict subcooled boiling when it happens. In addition, the requirements of the first step must be also satisfied in this solver, so that the same solver is able to simulate single- and two-phase simulations. Various tasks were struggling in this stage: First, it was necessary to modify the steam table external application in order to obtain properties at saturation conditions. Next, the heat balance between fluid and solid when two-phase are participating is extremely sensitive and it was necessary to apply relaxation factors so that the fluid volumetric heat and the wall temperature converged to the stable value and avoid the oscillation of the system. Finally, the interfacial heat transfer together with the subcooled boiling generated much more vapor if the signs criteria was not accurately defined.

The verification and validation of the new *my1DTPFoam* solver was performed by undertaking several simulations and comparing the results with the outcomes of different programs. TRACE code participates in all the presented cases, since it is considered the main reference for this work. However, COBRA-TF is also present to verify the capability of the new solver for one-dimensional simulations. Then, a three-dimensional program developed also within OpenFOAM was used to compare the results of a case because of its similar capabilities to the new solver. This comparison helped to analyze the amount of information that is lost due to the simplification to one dimension. The simulated results were also compared to the experimental data when this was available in order to validate the capabilities of the solver.

Two main experiments were presented in this work to verify and validate the new *my1DTPFoam*. First case consisted in an air-water flow running upward through

a vertical pipe. This simulation was based on an experiment carried out at the *Universitat Jaume I*, Castelló. The experiment was adiabatic and it allowed to determine the capability of the solver to consider the influence of the drag coefficient as well as the expansion of the air due to the pressure drop. Results show a proper simulation of the pressure drop, but the expansion is underestimated. The parameters involved in this phenomena should be revised.

The second experiment, known as Bartolomej benchmark, consisted also in a vertical pipe, but in this case only water flows upwards through it. The fluid was heated externally and boiling occurred. Results obtained by *my1DTPFoam* for this experiment were compared to TRACE and COBRA-TF from the system code side, and a three-dimensional solver developed within OpenFOAM framework, called *boilEulerFoam*. Then, it was validated by comparison of the new 1D CFD solver results with the experimental data. Regarding to the verification, the comparison of *my1DTPFoam* results with TRACE shows a certain underestimation of fluid parameters for the former solver. Water temperature reached saturation at the outlet section, when TRACE did not, and this leads to a greater void fraction at that region. Nonetheless, the rest of the geometry presents a void fraction which is underestimated compared to the reference code and to the experimental data. However, the comparison with the three dimensional code showed an agreement of the results at the outlet section, although the fluid temperature at the inlet did not agree at the same level. The main difference came from the wall temperature prediction, where *my1DTPFoam* values were lower than *boilEulerFoam* solution. In terms of validation, again, fluid temperature agreed at a high level with the experimental value, and void fraction followed a similar trend, being lightly underestimated. On the other hand, the wall temperature in the experimental case was higher than the value obtained by *my1DTPFoam*, which means that the dissipation of heat is lower than simulated. This argument agrees with the value of void fraction predicted, which may be higher due to a greater amount of heat transmitted to the water.

From the results presented in this work, one can conclude that overall, the capability of the new solver to simulate two-phase flows was verified and validated, being able to calculate subcooled boiling and wall heat transfer. Furthermore, it is able to depart from a single-phase simulation, as seen in the last experiment. However, there are terms that should be deeply analyzed, such as the expansion of the gas in the axial direction and the dissipation heat. A sensibility analysis of the proper pipe thickness should be considered, as well as the test of different wall heat transfer coefficient models and the capability of the closure models when the fluid reaches saturation conditions.

Overall, the one-dimensional CFD results agrees with the reference system code, so this new solver presents an alternative for 1D/3D couplings. Using OpenFOAM as the main platform, the coupling would avoid external links that need to translate the different codes, optimizing complex simulations.

7.2 Future work

The use of an open-source code allows a wide range of future contributions, among which, only those that were thought to carry out in a short-term and medium-term future are presented here:

- First, the possibility of interfacial area calculation considering phenomena present in nuclear reactors, like break-up and coalescence using population balance equations.
- The addition of models for the rest of flow regimes, in order to make the solver able to work with all regimes. Along with this, it would be necessary to implement an algorithm to automatically detect the flow regime, as well as a routine to calculate the critical heat flux.
- The integration of this solver within a three-dimensional CFD platform was meant to facilitate future 1D-3D coupling simulations, as mention in section 1.1. Therefore, this task would emphasize the work presented here, so it is a primary future goal.
- Finally, the development of a pseudo-3D solver is also included as a part of future work. The idea of creating an iterative process where first three one-dimensional solver, one for each axis, are calculated and then the calculation is repeated with the cross terms as source terms is considered. The pseudo-3D development will allow also the fully understanding of 1D-3D simulations.

CONTRIBUTIONS

The present work generated different scientific contributions, i.e international and national conferences and scientific articles. The different contributions are listed below, classified according to their category:

- **JCR indexed journal articles**

    1. **Title:** On the One-Dimensional Modeling of Vertical Upward Bubbly Flow.
    Authors: Peña-Monferrer C., Gómez-Zarzuela C., Chiva S., Miró R., Verdú, G. and Muñoz-Cobo J.L.
    Journal: Science and Technology of Nuclear Installations
    Year: 2018
    DOI: 10.1155/2018/2153019

- **International Conferences**

    1. **Title:** Development of a new one-dimensional solver using OpenFOAM.
    Authors: Peña-Monferrer C., Gómez-Zarzuela C., Chiva S., Miró R., and Verdú, G.
    Conference: 4th International Conference on Physics and Technology of Reactors and Applications (PHYTRA 4)
    Venue: Marrakech, Morocco
    Year: September 2018

2. **Title:** Comparison of interfacial drag force models for bubbly regime fluids in vertical pipes between one-dimensional codes and empirical experiment.
   Authors: Gómez-Zarzuela C., Peña-Monferrer C., Chiva S., Miró R., Verdú, G. and Muñoz-Cobo J.L.
   Conference: 8th International Nuclear Atlantic Conference (INAC 2017)
   Venue: Brazil
   Year: October 2017

3. **Title:** Validation of the new multiphase CFD solver BoilEulerFoam for the simulation of flow boiling
   Authors: Gómez-Zarzuela C., Rollins C., Luo H., Abarca A., Miró R., and Verdú, G.
   Conference: 17th International Topical Meeting on Nuclear Reactor Thermal-Hydraulics (NURETH-17)
   Venue: Xi'an (China)
   Year: September 2017

4. **Title:** Ringhals-1 BWR Stability Analysis With TRACE/PARCS
   Authors: Gómez-Zarzuela C., Barrachina T., Abarca A., Miró R., and Verdú, G.
   Conference: 24th International Conference Nuclear Energy for New Europe (NENE 2015)
   Venue: Portoroz (Slovenia)
   Year: September 2015

5. **Title:** Peach Bottom Low Flow Stability Analysis with TRACE/PARCS
   Authors: Gómez-Zarzuela C., Barrachina T., Abarca A., Miró R., and Verdú, G.
   Conference: 23rd International Conference Nuclear Energy for New Europe (NENE 2014)
   Venue: Portoroz (Slovenia)
   Year: September 2014

- **National Conferences**

1. **Title:** Aportaciones a la Simulación del Flujo Bifásico Unidimensional en Régimen Bubbly en Tuberías Verticales

Authors: Gómez-Zarzuela C., Peña-Monferrer C., Chiva S., Miró R., and Verdú, G.
Conference: 44 Reunión Anual de la Sociedad Nuclear Española
Venue: Ávila, Spain
Year: 2018

2. **Title:** Aportaciones al Desarrollo de un Nuevo Módulo para la Simulación de Fluidos en 1D en el Código CFD OpenFOAM
Authors: Gómez-Zarzuela C., Peña-Monferrer C., Chiva S., Miró R., and Verdú, G.
Conference: 44 Reunión Anual de la Sociedad Nuclear Española
Venue: Ávila, Spain
Year: 2018

3. **Title:** Implementación de un módulo 1-D en el código OpenFOAM
Authors: Gómez-Zarzuela C., Abarca A., Barrachina T., Peña-Monferrer C., Chiva S., Miró R.
Conference: 43 Reunión Anual de la Sociedad Nuclear Española
Venue: Málaga, Spain
Year: 2017

4. **Title:** Validación de un nuevo solver implementado en código CFD para la simulación de fluidos en ebullición
Authors: Gómez-Zarzuela C., Rollins C., Luo H., Miró R.
Conference: 43 Reunión Anual de la Sociedad Nuclear Española
Venue: Málaga, Spain
Year: 2017

5. **Title:** Evaluación de los modelos de transferencia de calor sólido/líquido en el código OpenFOAM para el estudio de canales termohidráulicos
Authors: Gómez-Zarzuela C., Peña-Monferrer C., Chiva S., Miró R., and Verdú, G.
Conference: 42 Reunión Anual de la Sociedad Nuclear Española
Venue: Santander, Spain
Year: 2016

6. **Title:** Análisis de estabilidad del reactor BWR Ringhals-I mediante los códigos acoplados TRACE/PARCS
   Authors: Gómez-Zarzuela C., C., Abarca A., Barrachina T., Miró R., Verdú, G.
   Conference: 41 Reunión Anual de la Sociedad Nuclear Española
   Venue: ACoruña, Spain
   Year: 2015

7. **Title:** Visualización en 3D de PARCS y COBRA-TF mediante el uso del paquete de herramientas VTK
   Authors: Gómez-Zarzuela C., C., Abarca A., Barrachina T., Miró R., Verdú, G.
   Conference: 41 Reunión Anual de la Sociedad Nuclear Española
   Venue: ACoruña, Spain
   Year: 2015

8. **Title:** Análisis de estabilidad de Peach Bottom NPP mediante TRACE/PARCS
   Authors: Gómez-Zarzuela C., C., Abarca A., Barrachina T., Miró R., Verdú, G.
   Conference: 40 Reunión Anual de la Sociedad Nuclear Española
   Venue: Valencia, Spain
   Year: 2014

BIBLIOGRAPHY

Ahrens et al. (2005). *ParaView: An End-User Tool for Large Data Visualization*. Ed. by Elsevier (cit. on p. 72).

Anderson, Nolan, Yassin Hassan, and Richard Schultz (2008). "Analysis of the hot gas flow in the outlet plenum of the very high temperature reactor using coupled RELAP5-3D system code and a CFD code". In: *Nuclear Engineering and Design* 238.1, pp. 274–279 (cit. on p. 10).

Angelucci, Morena et al. (2017). "STH-CFD codes coupled calculations applied to HLM loop and pool systems". In: *Sci. Technol. Nucl. Install.* 2017. ISSN: 16876083. DOI: 10.1155/2017/1936894 (cit. on p. 12).

Anglart, Henryk and Olov Nylund (1996). "CFD application to prediction of void distribution in two-phase bubbly flows in rod bundles". In: *Nuclear Engineering and Design* 163.1-2, pp. 81–98 (cit. on p. 14).

Antal, SP, RT Lahey Jr, and JE Flaherty (1991). "Analysis of phase distribution in fully developed laminar bubbly two-phase flow". In: *International Journal of Multiphase Flow* 17.5, pp. 635–652 (cit. on p. 14).

Ates, Ali, Selçuk Darici, and Sefik Bilir (2010). "Unsteady conjugated heat transfer in thick walled pipes involving two-dimensional wall and axial fluid con-

duction with uniform heat flux boundary condition". In: *International Journal of Heat and Mass Transfer* 53.23, pp. 5058 –5064. ISSN: 0017-9310. DOI: https://doi.org/10.1016/j.ijheatmasstransfer.2010.07.059 (cit. on p. 92).

Aumiller, D.L., E.T. Tomlinson, and R.C. Bauer (2001). "A coupled RELAP5-3D: CFD methodology with a proof-of-principle calculation". In: *Nuclear Engineering and Design* (cit. on p. 10).

Azzopardi, B. J. *Sauter Mean Diameter*. Ed. by Thermopedia. URL: http://thermopedia.com/content/1108/ (cit. on p. 47).

Barozzi, G. S. and G. Pagliarini (Feb. 1985). "A Method to Solve Conjugate Heat Transfer Problems: The Case of Fully Developed Laminar Flow in a Pipe". In: *Journal of Heat Transfer* 107.1, pp. 77–83. ISSN: 0022-1481. DOI: 10.1115/1.3247406 (cit. on p. 92).

Bartolomej G., Chanturiya V.M. (1967). "Experimental Study of True Void Fraction when Boiling Subcooled Water in Vertical Tubes". In: *Journal of Thermal Engineering* (cit. on pp. 16, 151, 152, 159).

Basu, Nilanjana, Gopinath R Warrier, and Vijay K Dhir (2002). "Onset of nucleate boiling and active nucleation site density during subcooled flow boiling". In: *Journal of heat transfer* 124.4, pp. 717–728 (cit. on pp. 68, 69).

Baviere, R et al. (2014). "A first system/CFD coupled simulation of a complete nuclear reactor transient using CATHARE2 and TRIO_U. Preliminary validation on the Phénix Reactor Natural Circulation Test". In: *Nuclear Engineering and Design* 277, pp. 124–137 (cit. on p. 10).

Bertodano, MA Lopez de (1993). "Turbulent bubbly two-phase flow in a triangular duct." In: (cit. on p. 48).

Bertolotto, D. et al. (2009). "Single-phase mixing studies by means of a directly coupled CFD/system-code tool". In: *Annals of Nuclear Energy* (cit. on pp. 10, 12).

Bestion, D. et al. (2014). *Extension of CFD Codes Application to Two-Phase Safety Problems - Phase 3*. Tech. rep. OECD (cit. on p. 13).

Cadinu, F., T. Kozlowski, and T-N. Dinh (2007). "Relating System-to-CFD Coupled Code Analyses to Theoretical Framework of a Multiscale Method". In: *Proceedings of ICAPP* (cit. on p. 12).

Caleb S. Brook T. Hibiki, M. Ishii (2012). "Interfacial drag force in one-dimensional two-fluid model". In: *Progress in Nuclear Energy* (cit. on pp. 51, 54).

Choe, WG, L Weinberg, and J Weisman (1978). "Observation and correlation of flow pattern transitions in horizontal, cocurrent gas-liquid flow". In: *Two Phase Transport and Reactor Safety*, pp. 1357–1375 (cit. on p. 58).

Churchill, Stuart W (1977). "Friction-factor equation spans all fluid-flow regimes". In: *Chemical engineering* 84.24, pp. 91–92 (cit. on pp. 28, 62).

Collier, John G. and John R. Thome (1994). *Convective Boiling and Condensation* (cit. on p. 63).

Colombo, M and M Fairweather (2016). "Accuracy of Eulerian Eulerian two fluid CFD boiling models of subcooled boiling flows". In: *International Journal of Heat and Mass Transfer* 103, pp. 28–44 (cit. on p. 15).

Corzo, S et al. (2012). "Numerical simulation of bubbly two-phase flow using eulerian-eulerian model". In: *Asociacion Argentina de Mecanica Computacional* 31, pp. 85–112 (cit. on pp. 15, 151).

Dittus, F. W. and L. M. K. Boelter (1930). *Heat transfer in automobile radiators of the tubular type* (cit. on p. 30).

Drew, Donald A and Lee A Segel (1971). "Averaged equations for two-phase flows". In: *Studies in Applied Mathematics* 50.3, pp. 205–231 (cit. on pp. 14, 32).

Drzewiecki, Timothy J. et al. (2012). "Parameter Sensitivity Study of Boiling and Two-Phase Flow Models in CFD". In: *J. Comput. Multiph. Flows* 4.4, pp. 411–425. ISSN: 1757-482X. DOI: 10.1260/1757-482x.4.4.411 (cit. on p. 15).

Dynamics, Wolf (2019). *Supplement: Meshing with blockMesh*. URL: http://www.wolfdynamics.com/images/OF_intro_training/supplement_blockmesh.pdf (cit. on p. 114).

Fang, Xiande, Yu Xu, and Zhanru Zhou (2011). "New correlations of single-phase friction factor for turbulent pipe flow and evaluation of existing single-phase friction factor correlations". In: *Nuclear Engineering and Design* 241.3, pp. 897–902 (cit. on p. 28).

Fanning, TH, JW Thomas, et al. (2010). *Advances in coupled safety modeling using systems analysis and high-fidelity methods.* Tech. rep. Argonne National Lab.(ANL), Argonne, IL (United States) (cit. on p. 11).

Ferrell, James K. and John William McGee (1966). *Two-phase flow through abrupt expansions and contractions final report volume III on a study of convection boiling inside channels*. Technical Document. North Carolina State University, Department of Chemical Engineering (cit. on pp. 62, 63).

Ferziger, Joel H and Milovan Perić (2002). *Computational methods for fluid dynamics*. Vol. 3. Springer (cit. on p. 77).

Fu, Kai and Henryk Anglart (2017). "Implementation and validation of two-phase boiling flow models in OpenFOAM". In: arXiv: 1709.01783 (cit. on pp. 15, 151).

Garnier, J, E Manon, and G Cubizolles (2001). "Local measurements on flow boiling of refrigerant 12 in a vertical tube". In: *Multiphase Science and Technology* 13.1&2 (cit. on p. 17).

Ghione, A. (2011). "Development and validation of a two-phase CFD model using OpenFOAM". MA thesis. Royal Institute of Technology (cit. on pp. 15, 33–35, 47, 125, 127).

Gnielinski, V. (1976). "New Equations flow regime Heat and Mass Transfer in Turbulent Pipe and Channel Flow". In: *Int. Chem. Eng.* (cit. on p. 30).

Godino, D. M. and D. E. Ramajo S. F. Corzo N. M. Nigro (2018). "CFD simulation of the pre-heater of a nuclear facility steam generator using a thermal coupled model". In: *Nuclear Engineering and Design* (cit. on p. 10).

Gómez-Zarzuela, C et al. (2017a). "Comparison of drift-velocity and drag coefficient approaches for one-dimensional two-fluid models in bubbly flow regime and validation with experimental data". In: (cit. on p. 148).

Nuclear Reactor Thermal Hydraulics (NURETH-17), International Topical Meeting on, ed. (2017b). *Validation of the new Multiphase CFD solver BoilEulerFoam for the simulation of flow boiling* (cit. on p. 152).

Grgic, D. et al. (2002). "Coupled RELAP5/GOTHIC Model For Accident Analysis Of The IRIS Reactor". In: *Technical Meeting on Use of Computational Fluid Dynamics (CFD) Codes for Safety Analysis of Reactor Systems, Including Containment*. Ed. by International Atomic Energy Agency and OECD Nuclear Energy Agency (cit. on p. 10).

Grunloh, T.P. and A. Manera (2016). "A novel domain overlapping strategy for the multiscale coupling of CFD with 1D system codes with applications to transient flows". In: *Annals of Nuclear Energy* (cit. on pp. 10, 12).

H.Anglart (2010). *Thermal-hydraulics in nuclear systems*. URL: `http://kth.diva-portal.org/smash/get/diva2:500651/FULLTEXT01.pdf` (cit. on p. 44).

Hibiki, Takashi and Mamoru Ishii (2002). "Development of one-group interfacial area transport equation in bubbly flow systems". In: *International Journal of Heat and Mass Transfer* 45.11, pp. 2351–2372 (cit. on p. 14).

— (2007). "Lift force in bubbly flow systems". In: *Chemical Engineering Science* 62.22, pp. 6457 –6474. ISSN: 0009-2509. DOI: `https://doi.org/10.1016/j.ces.2007.07.034` (cit. on p. 14).

Holman, Jack P (2010). *Heat transfer*. McGraw-hill (cit. on pp. 30, 31).

Holzinger, Gerhard (2016). *OpenFOAM A little User-Manual*. English. CD-Laboratory - Particulate Flow Modelling Johannes Kepler University, Linz, Austria. 293 pp. (cit. on pp. 38, 85–87, 122).

Hosokawa, Shigeo and Akio Tomiyama (2009). "Multi-fluid simulation of turbulent bubbly pipe flows". In: *Chemical Engineering Science* 64.24, pp. 5308–5318 (cit. on p. 32).

Huber, K. (2017). "A Multiscale Method for Mixed Convective Systems. Coupled calculations with ATHLET and OpenFOAM of the PHENIX NCT". Karlsruhe Institute of Technology (cit. on p. 10).

IAPWS (2019). *The International Association for the Properties of Water and Steam*. URL: `http://www.iapws.org/` (cit. on p. 98).

IDAE (2019). *PNIEC, Plan Nacional Integrado de Energia y Clima 2021 - 2030*. Energy. Ministerio para la Transicón Ecológica Gobierno de España (cit. on p. 1).

Ishii, M and TC Chawla (1979). "Local drag laws in dispersed two-phase flow". In: *Nasa Sti/Recon Technical Report N* 80 (cit. on p. 53).

Ishii, M. and T. Hibiki (2011). *Thermo-Fluid Dynamics of Two-Phase Flow*. Springer (cit. on pp. 14, 45, 46, 48, 51).

Ishii, M. and K. Mishima (1984). "Two-Fluid Model And Hydrodynamic Constitutive Relations". In: *Nuclear Engineering and Design* (cit. on pp. 14, 44, 48, 51).

Ishii, Mamoru and Novak Zuber (1979). "Drag coefficient and relative velocity in bubbly, droplet or particulate flows". In: *AIChE journal* 25.5, pp. 843–855 (cit. on p. 47).

Issa, Raad I (1986). "Solution of the implicitly discretised fluid flow equations by operator-splitting". In: *Journal of computational physics* 62.1, pp. 40–65 (cit. on p. 87).

Ivanov, K. and M. Avramova (2007). "Challenges in coupled thermal-hydraulics and neutronics simulations for LWR safety analysis". In: *Annals of Nuclear Energy* (cit. on p. 10).

Jeong, J. J. et al. (1997). "Assesment of the COBRA/RELAP5 code using the loft L2-3 large-break Loss-Of-Coolant Experiment". In: *Annals of Nuclear Energy* (cit. on p. 10).

Jeong, Ji Hwan and Byoung-Sub Han (2008). "Coolant flow field in a real geometry of PWR downcomer and lower plenum". In: *Annals of Nuclear Energy* 35.4, pp. 610–619 (cit. on p. 8).

Jo, Jong Chull and Dong Gu Kang (Jan. 2010). "CFD Analysis of Thermally Stratified Flow and Conjugate Heat Transfer in a PWR Pressurizer Surgeline". In: *Journal of Pressure Vessel Technology* 132.2. ISSN: 0094-9930. DOI: 10.1115/1.4000727 (cit. on p. 92).

Kataoka, Isao and Mamoru Ishii (1987). "Drift flux model for large diameter pipe and new correlation for pool void fraction". In: *International Journal of Heat and Mass Transfer* 30.9, pp. 1927–1939 (cit. on p. 52).

Kirillov, P. L. (2008). *Thermophysical properties of materials for nuclear engineering* (cit. on pp. 141, 142).

Kliem, S. et al. (1999). *Main Steam Line Break Analysis of a VVER-440 Reactor Using the Coupled Thermohydraulics system/3D-Neutron Kinetics Code DYN3D / ATHLET in Combination with the CFD Code CFX-4*. In: ed. by Ninth International Topical Meeting on Nuclear Reactor Thermal Hydraulics (NURETH-9) (cit. on p. 10).

Kööp, Kaspar (2018). "Application of a system thermal-hydraulics code to development of validation process for coupled STH-CFD codes". PhD thesis. KTH Royal Institute of Technology (cit. on p. 11).

Kowalczuk, Przemyslaw and Jan Drzymala (Oct. 2015). "Physical Meaning of the Sauter Mean Diameter of Spherical Particulate Matter". In: *Particulate Science and Technology* 34, pp. 645–647. DOI: 10.1080/02726351.2015.1099582 (cit. on p. 47).

Krepper, Eckhard, Bostjan Konar, and Yury Egorov (2007). "CFD modelling of subcooled boiling. Concept, validation and application to fuel assembly design". In: *Nuclear Engineering and Design* 237.7, pp. 716 –731. ISSN: 0029-5493. DOI: https://doi.org/10.1016/j.nucengdes.2006.10.023 (cit. on pp. 15, 32, 151, 152).

Kurul, N. and Michael Z. Podowski (1990). "MULTIDIMENSIONAL EFFECTS IN FORCED CONVECTION SUBCOOLED BOILING". In: (cit. on pp. 92, 152).

Kurul, Necdet (1991). "Multidimensional effects in two-phase flow including phase change". In: (cit. on p. 14).

Lahey Jr, Richard T (1978). "A mechanistic subcooled boiling model". In: *International Heat Transfer Conference Digital Library*. Begel House Inc. (cit. on p. 67).

Lee, Gong Hee et al. (2014). "Comparative study on the effect of reactor internal structure geometry modeling methods on the prediction accuracy for PWR internal flow distribution". In: *Annals of Nuclear Energy* 70, pp. 208–215 (cit. on p. 8).

Lemmon, E. W. et al. (2018). *NIST Standard Reference Database 23: Reference Fluid Thermodynamic and Transport Properties-REFPROP, Version 10.0, National Institute of Standards and Technology* (cit. on p. 101).

Levy, S. (1967). "Forced Convection Subcooled Boiling Prediction of Vapor Volumetric Fraction". In: *Int. J. Heat Mass Transfer* (cit. on p. 63).

Li, W. et al. (2014). "Preliminary study of coupling CFD code FLUENT and system code RELAP5". In: *Annals of Nuclear Energy* (cit. on p. 10).

Mahaffy, J. et al. (2007). *Best Practice Guidelines for the use of CFD in Nuclear Reactor Safety Applications*. Tech. rep. NEA/CSNI (cit. on p. 8).

Marić, T, J Hopken, and K Mooney (2014). *The OpenFOAM Technology Primer* (cit. on p. 73).

Márquez-Damián, S. (2013). "An Extended Mixture Model for the Simultaneous Treatment of Short and Long Scale Interfaces". PhD thesis. Universidad Nacional del Litoral (cit. on p. 126).

Michta, Edouard (2011). "Modeling of Subcooled Nucleate Boiling with OpenFOAM". In: February, p. 114 (cit. on p. 15).

Michta, Edouard et al. (2012). "Numerical Predictions of Bubbly Two-Phase Flows with OpenFOAM". In: *J. Comput. Multiph. Flows* 4.4, pp. 351–362. ISSN: 1757-482X. DOI: 10.1260/1757-482X.4.4.351 (cit. on p. 15).

Monrós-Andreu, G. et al. (2013). "Water temperature effect on upward air-water flow in a vertical pipe: Local measurements database using four-sensor conductivity probes and LDA". In: *EPJ Web of Conferences*. Vol. 45. EDP Sciences, p. 01105 (cit. on p. 147).

Monrós-Andreu, G. et al. (2017). "Local parameters of air–water two-phase flow at a vertical T-junction". In: *Nuclear Engineering and Design* 312, pp. 303–316 (cit. on p. 147).

Moukalled, F., L. Mangani, M. Darwish, et al. (2016). "The finite volume method in computational fluid dynamics". In: *An advanced introduction with OpenFOAM® and Matlab®* (cit. on pp. 23, 27, 75, 77).

NRC, U.S. (2013). *TRACE V5.840 Theory Manual* (cit. on pp. 29, 30, 50, 52, 62, 69, 161, 162).

OpenCFD-Ltd (2018). *OpenFOAMv1712 User Guide* (cit. on pp. 21, 73, 74, 90, 121).

openfoamwiki.net (2013). URL: https://openfoamwiki.net/index.php/OpenFOAM_guide/The_PISO_algorithm_in_OpenFOAM (cit. on p. 85).

Papukchiev, A. et al. (2009). *Extension of the Simulation Capabilities of the 1D System Code ATHLET by Coupling with the 3D CFD Software Package ANSYS CFX*. In: ed. by 13th International Topical Meeting on Nuclear Reactor Thermal Hydraulics (NURETH-13) (cit. on p. 10).

Papukchiev, Angel et al. (2015). “Comparison of different coupling CFD–STH approaches for pre-test analysis of a TALL-3D experiment”. In: *Nuclear Engineering and Design* 290, pp. 135–143 (cit. on p. 12).

Pedersen, J. R. et al. (2017). “A New Volume-Of-Fluid Method in Openfoam”. In: *MARINE 2017 Computational Methods in MarineEngineering VII*. Ed. by M. Visonneau, P. Queutey, and D. Le TouzÃ© (cit. on p. 126).

Peña-Monferrer, C. (2017). “Computational fluid Dynamics Multiscale Modelling of Bubbly Flow. A Critical Study and New Developments On Volume of Fluid, Discrete Element and Two-Fluid Methods”. PhD thesis. Universitat Politècnica de València (cit. on pp. 13, 16).

Peña-Monferrer, C. et al. (2018a). “A CFD-DEM solver to model bubbly flow. Part I: Model development and assessment in upward vertical pipes”. In: *Chemical Engineering Science* 176, pp. 524 –545. ISSN: 0009-2509. DOI: https://doi.org/10.1016/j.ces.2017.11.005 (cit. on p. 147).

— (2018b). “A CFD-DEM solver to model bubbly flow. Part II: Critical validation in upward vertical pipes including axial evolution”. In: *Chemical Engineering Science* 177, pp. 537 –556. ISSN: 0009-2509. DOI: https://doi.org/10.1016/j.ces.2017.11.032 (cit. on p. 147).

Peña-Monferrer, C. et al. (Jan. 2018c). “On the One-Dimensional Modeling of Vertical Upward Bubbly Flow”. In: *Science and Technology of Nuclear Installations* 2018, pp. 1–10. DOI: 10.1155/2018/2153019 (cit. on pp. 49, 55–57, 147, 148).

Perelman, T. L. (1961). “On conjugated problems of heat transfer”. In: *International Journal of Heat and Mass Transfer* 3.4, pp. 293–303 (cit. on p. 93).

Petukhov, B.S. (1970). “Heat Transfer and Friction in Turbulent Pipe Flow with Variable Physical Properties”. In: ed. by James P. Hartnett and Thomas F. Irvine. Vol. 6. Advances in Heat Transfer. Elsevier, pp. 503 –564. DOI: https://doi.org/10.1016/S0065-2717(08)70153-9 (cit. on p. 32).

Podowski, M. Z. et al. (1997). *Mechanistic modeling of CHF in forced-convection subcooled boiling*. Tech. rep. Knolls Atomic Power Lab., Schenectady, NY (United States) (cit. on p. 14).

Pye, J. (2013). *Freesteam*. URL: `http://freesteam.sourceforge.net/` (cit. on pp. 98, 99).

RELAP5/MOD3 code manual (1995). *Volume 4, Models and correlations*. Nuclear Regulatory Commission (cit. on pp. 54, 55).

Richardson, J. F. and W. N. Zaki (1997). "Sedimentation and fluidisation: Part I". In: *Chemical Engineering Research and Design* 75, S82–S100 (cit. on p. 60).

Rivard, W. C. and W. C. Torrey (1977). *Assessment of CFD Codes for Nuclear Reactor Safety Problems - Revision 2K-FIX: A computer program for transient, two-dimensional, two-fluid flow*. Tech. rep. Los Alamos National Laboratory, LA-NUREG-6623 (cit. on p. 8).

Rollins, C. (2018). "Development of Multiphase Computational Fluid Dynamics Solver in OpenFOAM". PhD thesis. North Carolina State University (cit. on pp. 15, 47, 48, 125, 127, 152, 162).

Rusche, H. (2002). "Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phase Fractions". PhD thesis. Imperial College of Science, Technology and Medicine (cit. on pp. 33, 36, 39, 122, 125, 127).

Rzehak, Roland (2013). "Closure models for turbulent bubbly flows: A CFD study". In: *Nucl. Eng. Des.* 265, pp. 701–711. ISSN: 00295493. DOI: `10.1016/j.nucengdes.2013.09.003` (cit. on p. 15).

Rzehak, Roland and Sebastian Kriebitzsch (2015). "Multiphase CFD-simulation of bubbly pipe flow: A code comparison". In: *Int. J. Multiph. Flow* 68, pp. 135–152. ISSN: 03019322. DOI: `10.1016/j.ijmultiphaseflow.2014.09.005` (cit. on p. 15).

Schiller, L (1933). "A drag coefficient correlation". In: *Zeit. Ver. Deutsch. Ing.* 77, pp. 318–320 (cit. on p. 47).

Sleicher, C.A. and M.W. Rouse (1975). "A convenient correlation for heat transfer to constant and variable property fluids in turbulent pipe flow". In: *International Journal of Heat and Mass Transfer* 18.5, pp. 677 –683. ISSN: 0017-9310.

DOI: https://doi.org/10.1016/0017-9310(75)90279-3 (cit. on p. 31).

Smith, B.L. et al. (2015). *Assessment of CFD Codes for Nuclear Reactor Safety Problems - Revision 2*. Tech. rep. NEA/CSNI (cit. on p. 8).

Snowsill, W.L. (2010). "Chapter 16 - Particle Sizing". In: *Instrumentation Reference Book (Fourth Edition)*. Ed. by Walt Boyes. Fourth Edition. Boston: Butterworth-Heinemann, pp. 175 –189. ISBN: 978-0-7506-8308-1. DOI: https://doi.org/10.1016/B978-0-7506-8308-1.00016-4 (cit. on p. 56).

Spalding, D. B. (1985). *The numerical computation of multi-phase flows*. Tech. rep., p. 23895 (cit. on pp. 36, 37).

Steiner, Helfried, Alexander Kobor, and Ludwig Gebhard (2005). "A wall heat transfer model for subcooled boiling flow". English. In: *International journal of heat and mass transfer* 48, pp. 4161–4173. ISSN: 0017-9310 (cit. on p. 92).

Taitel, Y., D. Bornea, and A.E. Dukler (May 1980). "Modelling flow pattern transitions for steady upward gas-liquid flow in vertical tubes. [Bubble, slug, churn and dispersed-annular; also existence regions and transitions]". In: *AIChE J.; (United States)* 26. DOI: 10.1002/aic.690260304 (cit. on p. 44).

Thiele, R. and C. Lucas (2016). *IAPWS-IF97-OF* (cit. on p. 98).

Todreas, Neil E and Mujid S Kazimi (2011). *Nuclear Systems Volume I: Thermal Hydraulic Fundamentals*. CRC press (cit. on pp. 27, 40).

Tomiyama, Akio (1998). "Struggle with computational bubble dynamics". In: *Multiphase Science and Technology* 10.4, pp. 369–405 (cit. on p. 47).

Tomiyama, Akio et al. (1998). "Drag Coefficients of Single Bubbles under Normal and Micro Gravity Conditions". In: *JSME Int. J. Ser. B* 41.2, pp. 472–479 (cit. on pp. 47, 55).

Toti, A., J. Vierendeels, and F. Belloni (2016). "Development and Preliminary Validation of a STH-CFD Coupling Method for Multiscale Thermal-Hydraulic

Simulations of the MYRRHA Reactor". In: *2016 24th International Conference on Nuclear Engineering*. American Society of Mechanical Engineers, V004T10A018–V004T10A018 (cit. on p. 11).

— (2017). "Improved numerical algorithm and experimental validation of a system thermal-hydraulic/CFD coupling method for multi-scale transient simulations of pool-type reactors". In: *Annals of Nuclear Energy* (cit. on pp. 10, 12).

Tu, J. Y. and G. H. Yeoh (2002). "On numerical modelling of low-pressure subcooled boiling flows". In: *International Journal of Heat and Mass Transfer* 45.6, pp. 1197 –1209. ISSN: 0017-9310. DOI: https://doi.org/10.1016/S0017-9310(01)00230-7 (cit. on p. 14).

Versteeg, H. K. and W. Malalasekera (2007). *An Introduction to Computational Fluid Dynamics*. Vol. 3. Pearson Education Limited (cit. on pp. 77, 108, 109).

Wagner (2008). "IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam". In: *International Steam Tables: Properties of Water and Steam Based on the Industrial Formulation IAPWS-IF97*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 7–150. ISBN: 978-3-540-74234-0. DOI: 10.1007/978-3-540-74234-0_3 (cit. on p. 98).

Wallis, G. B. (1969). *One-dimensional two-phase flow* (cit. on pp. 28, 43, 54, 60, 63).

Weaver, W. L., E. T. Tomlinson, and D. L. Aumiller (2002). "A generic semi-implicit coupling methodology for use in RELAP5-3D©". In: *Nuclear Engineering and Design* 211.1, pp. 13–26 (cit. on p. 12).

Welch, Samuel W. J. and Thami Rachidi (2002). "NUMERICAL COMPUTATION OF FILM BOILING INCLUDING CONJUGATE HEAT TRANSFER". In: *Numerical Heat Transfer, Part B: Fundamentals* 42.1, pp. 35–53. DOI: 10.1080/10407790190053824 (cit. on p. 92).

Weller, Henry G et al. (1998). "A tensorial approach to computational continuum mechanics using object-oriented techniques". In: *Computers in physics* 12.6, pp. 620–631 (cit. on p. 21).

Weller, H.G. (2005). *Derivation, Modelling and Solution of the Conditionally Averaged Two-Phase Flow Equations* (cit. on pp. 33, 36).

Willerding, G. and W. Baumann (1996). *FLUTAN 2.0, Input Specifications*. Tech. rep. Institut fur Neutronenphysik und Reaktortechnik (cit. on p. 8).

Yao, Wei and Christophe Morel (2004). "Volumetric interfacial area prediction in upward bubbly two-phase flow". In: *International Journal of Heat and Mass Transfer* 47.2, pp. 307–328 (cit. on p. 14).

Yapici, Hüseyin and Bilge Albayrak (2004). "Numerical solutions of conjugate heat transfer and thermal stresses in a circular pipe externally heated with non-uniform heat flux". In: *Energy Conversion and Management* 45.6, pp. 927–937. ISSN: 0196-8904. DOI: https://doi.org/10.1016/S0196-8904(03)00195-X (cit. on p. 92).

Yeoh, G. H. and J.Y. Tu (2006). "Numerical modelling of bubbly flows with and without heat and mass transfer". In: *Applied Mathematical Modelling* 30.10. Special issue of the 12th Biennial Computational Techniques and Applications Conference and Workshops (CTAC-2004) held at The University of Melbourne, Australia, from 27th September to 1st October 2004, pp. 1067–1095. ISSN: 0307-904X. DOI: https://doi.org/10.1016/j.apm.2005.06.012 (cit. on p. 14).

— (2009). *Modelling Subcooled Boiling Flows*. Nova Science Publishers. ISBN: ISBN 978-1-60876-420-4 (cit. on p. 66).

Yildirim, Gürol (2009). "Computer-based analysis of explicit approximations to the implicit Colebrook–White equation in turbulent flow friction factor calculation". In: *Advances in Engineering Software* 40.11, pp. 1183–1190 (cit. on p. 28).

Zalesak, S. (1979). "Fully multidimensional Flux-Corrected Transport algorithms for fluids". In: *Journal of Computational Physics* (cit. on p. 126).

Zuber, Novak and JAa Findlay (1965). "Average volumetric concentration in two-phase flow systems". In: *Journal of heat transfer* 87.4, pp. 453–468 (cit. on p. 50).