CHALMERS

Numerical Simulations of Subcooled Nucleate Boiling for a Power Electronic Device

Master's Thesis in Engineering Mathematics

JONAS PETTERSSON

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2014

Acknowledgements

I want to thank ABB and in particular my supervisors Tor Laneryd and Rebei Bel Fdhila for giving me the opportunity to do my master thesis at ABB corporate research. It has been a great learning experience. Thanks to Henryk Anglart for taking the time to supervise and assist me in my work. His expertise in the field has been invaluable. Thanks to Nils Lavesson for helping me with the detective work required to manoeuvre OpenFOAM and Ulf Sand for his advice on CFD-simulations. Finally I want to thank Vlad Radoi for assisting me with the proof reading of my thesis.

Abstract

The growing demand for power in today's society require large power grids containing high power electronic devices, of which ABB is a leading manufacturer. In these devices power losses inevitably occur, leading to heat generation. The heat eventually causes the device to fail, hence cooling such devices are of great importance. A way to cool such a system is by liquid cooling with phase change (subcooled nucleate boiling) as an energy transport phenomenon. To develop such cooling systems, a mathematical model with a numerical solution procedure is needed to predict the physical properties.

To achieve this, a literature study has been carried out to formulate a closed mathematical system, describing this problem. The formulated model is based on previously verified models for boiling properties, such as bubble diameters, wall super heat, heat flux, mass transfer due to evaporation and condensation. These models were combined with a new approach to calculate wall temperature. The new wall temperature was introduced to save computational time, since this often is an issue when implementing these types of models.

A case with existing experimental data, found in the literature study, was simulated and the results were compared with varying agreement. The vapour content proved to be under predicted by the model and the introduced wall temperature model gave over predicted results. Good agreement could be seen of the evaporation heat fluxes with some dependency of the bubble diameter in the system. Despite over and under predictions, the trends of several properties agreed with the reference data. The developed solver also performed stable computations and was well documented, preparing it for future development.

KEYWORDS: CFD, OpenFOAM, subcooled nucleate boiling flow, wall temperature, bubble diameter model, heat flux partitioning, cooling

Contents

Nomenclature

Roman Symbols

Symbol	Description	Dimension	Base Units
$\boldsymbol{b}$	Body force tensor	–	–
$\boldsymbol{D}$	Rate of strain tensor	–	–
$\boldsymbol{g}$	Gravity vector	–	m/s^2
$\boldsymbol{I}$	Identity tensor	–	–
$\boldsymbol{n}$	Normal vector	–	–
$\boldsymbol{T}$	Stress vector	–	–
$\boldsymbol{u}$	Velocity vector	–	m/s
$\dot{M}$	Mass flow	–	kg/s
$\dot{m}$	Mass flux	–	$kg/s/m^2$
A	Area	–	m^2
C_D	Drag coefficient	–	–
C_p	Specific heat capacity	$J/kg/K$	$m^2/K/s^2$
d_B	Bubble diameter	–	m
d_{Dep}	Bubble detachment diameter	–	m
e_{in}	Internal energy per unit mass	J/kg	m^2/s^2
e_{kin}	Kinetic energy per unit mass	J/kg	m^2/s^2
f	Bubble detachment frequency	Hz	s^{-1}
g	Acceleration of gravity	–	m/s^2
h	Specific enthalpy	J/kg	m^2/s^2
H_{if}	Interfacial heat transfer coefficient	$W/m^2/K$	$kg/s^3/K$
H_{single}	Single-phase forced heat transfer coefficient	$W/m^2/K$	$kg/s^3/K$
k	Turbulent kinetic energy	–	m^2/s^2
L	Latent heat of evaporation	J/kg	$m2/s^2$
N	Nucleation site density	–	m^{-2}
P	Pressure	Pa, bar	$kg/m/s^2$

q	Heat transfer rate	W	$kg \cdot m^2/s^3$
q''	Heat flux	W/m^2	kg/s^3
q_b	Body heat source	–	$kg/s/m^2$
t	Time	–	–
T^*	Dimensionless temperature	–	–
u^+	Dimensionless velocity	–	–
V	Volume	–	m^3
y^*	Dimensionless wall distance according to [17]	–	–
y^+	Dimensionless wall distance	–	–
a_i	Interfacial area concentration	–	m^{-1}

Greek Symbols

Symbol	Description	Dimension	Base Units
α	Void fraction	–	–
$\boldsymbol{\sigma}$	Cauchy stress tensor	–	–
$\boldsymbol{\tau}$	Shear stress tensor	–	–
χ	Thermal diffusivity	–	m^2/s
ϵ	Turbulent dissipation	–	m^2/s^3
Γ_{cond}	Condensation rate per unit volume	–	$kg/m^3/s$
Γ_{evap}	Evaporation rate per unit volume	–	$kg/m^3/s$
κ	Von Karmam's constant $\kappa = 0.41$	–	–
λ	Thermal conductivity	$W/m/K$	$kg \cdot m/s^3/K$
μ	Dynamic viscosity	$Pa \cdot s$	$kg/m/s$
ν	Kinematic viscosity	–	m^2/s
ρ	density	–	kg/m^3
σ	Surface tension	–	kg/s^2
τ	Shear stress	–	–

Dimensionless Numbers

Symbol	Description	Definition

Pr_t	Turbulent Prandtl number	ν^t/χ
Re_L	Global liquid Reynolds number	$d_{pipe}\|\|\boldsymbol{u}_L\|\|/\nu_L$
Re_{d_B}	Bubble Reynolds number	$d_B\|\|\boldsymbol{u}_V - \boldsymbol{u}_L\|\|/\nu_L$
C_f	Fanning friction factor	$2\tau/\rho u^2$
Nu	Nusselt number	Hl/λ
Pr	Prandtl number	ν/χ
St	Stanton number	$H/\rho/\overline{\boldsymbol{u}}/C_p$

Subscripts

Symbol	**Description**
$bulk$	Bulk conditions
dev	Deviatoric
hyd	Hydrostatic
in	Internal
$inst$	Instantaneous
kin	Kinetic
L	Liquid phase
sat	Saturated conditions
$surf$	Surface conditions
V	Vapour phase
$wall$	Wall

Superscripts

Symbol	**Description**
$*$	Dimensionless quantity from [17]
$+$	Standard dimensionless quantity
eff	Effective
$model$	Modelled
t	Turbulent

List of Figures

List of Tables

1 Introduction

1.1 Context

Phase changing phenomenon are present in many engineering applications today. One such phenomena is boiling, where a medium changes from liquid phase to gas phase. This phenomena can be encountered in a variety of systems, like power systems and heat transfer systems [14]. In power systems it occurs in, for example, conventional power plants with boilers and evaporators or in boiling water reactors in nuclear power plants. In heat transfer systems it can be found in different heat exchangers like evaporators and heat pipes.

With the growing demand for power in today's society the need for robust power transmission systems capable of handling large amounts of power are needed .The electrical devices in such systems have to sustain high loads. This results in large development of heat in the components of such devices, causing power losses and possible damage to the system. Cooling such systems properly is therefore of great importance. Prototypes of systems of this complex nature can be both time and cost consuming. Therefore, a proper procedure to model and simulate such systems can be highly beneficial.

One way to use boiling, in particular subcooled nucleate boiling section 2.4, as a cooling system could be to have a liquid flow over a heated surface with conditions, like liquid and pressure, adjusted so that boiling can start at the surface. When phase change occur, a large amount of heat energy is transported, between the phases, in the process. The developed vapour forms bubbles that detaches from the surface and drifts into the bulk, where they condensates back to liquid phase. To model this, a solid mathematical model and a numerical solution to such a model need to be formulated and solved.

1.2 Background

ABB is a large manufacturer and developer of power electronic devices. Many of these devices generate a large amount of heat when they are in service. To cool these components a number of different methods can be considered.

ABB collaborates with professor Henryk Anglart at the department for Nuclear Reactor Technology at KTH in the development of simulation tools for cooling systems. Professor Anglart works with boiling models associated with heat transfer in nuclear reactors. His research includes development of accurate and robust models for subcooled nucleate boiling. In this collaboration, ABB is investigating how to use these models to develop cooling systems for their products. An important part of this investigation is to develop a good model of subcooled nucleate boiling for systems with similar conditions found in their cooling systems. The existing models are usually developed for the nuclear power industry. A variety of models for different physical phenomena have been developed and validated in the past. A lot of research has been done to develop models of bubble diameters, vapour propagation, mass transfer due to evaporation and condensation to name a few. However, these models are often only valid for water and simple geometries, like vertical cylinders. Therefore, to obtain a proper model for ABB's intentions, parts of the existing models need to be changed.

The implementation of such models have been done in different CFD-softwares. One such is OpenFOAM, which is an open source software suitable for developing new solvers. OpenFOAM contains two-phase solvers adjusted to handle materials of several phases like liquid and gas in the same system. In the latest version 2.3.0 of OpenFOAM the *twoPhaseEulerFoam*-solver has been updated with new approaches to calculate some of the two-phase variables, but lack implementation of the phase change.

1.3 Purpose

To develop a robust model and solver for a general liquid and a general geometry is a great task. The purpose of this thesis is therefore restricted to introducing ABB to the theory of subcooled nucleate boiling and to implementation of a such model in OpenFOAM with professor Anglart as the expert in the field.

The developed model will use existing models for all the different aspects of subcooled nucleate boiling except for two parts. The wall temperature and the point where boiling phenomena are initiated will be modelled with a new approach, suggested by professor Anglart. The purpose of the new approach is to reduce the amount of computational resources needed to model this to parts.

The model will be restricted to using water as a liquid and the geometry will be a vertical cylindrical pipe. The result will then be compared to empirical data, to verify if the new modelling, of the ONB, works.

The implementation of the solver will be based on the new *twoPhaseEulerFoam*-solver. The implementation will be carried out in way that makes future adjustments and development, like switching liquid and geometry, easy.

1.4 Limitations

The simulated cases will be restricted to be as simple as possible. As mentioned above the model will only be valid for water as the liquid and the geometry being a vertical cylindrical pipe. The geometry will also be restricted to a 2-D approximation in the form of a wedge of a cylinder to save computational time. Many physical properties will be assumed to be constant, even though they change with temperature, to keep the work load and computation time from growing out of proportion.

2 Theory

In this section basic theory about single-phase fluid mechanics and turbulence are presented. From the theory of single-phase flow the theory about multi-phase flow is developed, according to the Euler method. Basic concepts of heat transfer and a description of the finite volume method are also presented.

2.1 Single-Phase Fluid Mechanics

The main objective of fluid mechanics is to describe and predict physical properties of gas and liquid flows. Systems with such flows possess different properties like pressure, temperature, velocity, density, turbulent/laminar etc. Systems are described by a set of partial differential equations called governing equations derived from basic physical principles. Different approaches can be employed to derive the governing equations and can be found in any basic text on the subject such as [1, 7, 12, 29]. In the following sections the governing equations are derived by defining a so-called control volume.

2.1.1 Continuum Mechanics

To describe the physics of a system it can be modelled by continuum mechanics [18] which was first introduced by the french mathematician Augustion-Louis Cauchy. It models materials as a continuous mass instead of discrete atoms. Hence, it is assumed that the material completely fills the volume it occupies. This approximation gives accurate results, in most systems in engineering applications, due to the large difference between the application scales and the atomic scales.

In addition to the above assumptions three conservation principles are applied, to a closed system, to derive the governing equations. A closed system refers to a system where there is no transfer of matter, momentum or energy with the outside of the system. The conservation principles regard the mass, momentum and energy of the closed system. From these conservation principles three governing equations are derived known as the *continuity equation*, the *momentum equation* and the *energy equation* [1].

2.1.2 Governing Equations

To derive the governing equations the concept of a control volume is introduced. A control volume Ω is defined as a small arbitrary volume with a surface $\partial\Omega$ fixed in space through which the fluid flows. By applying the conservation principles to Ω the governing equations can be derived as done in [1].

Continuity Equation The *continuity equation*, also known as the *balance equation for mass* is based on the physical principle that mass can be neither destroyed nor created [1]. This physical principle states that the mass flow into a control volume equals the rate of change of mass inside the control volume. Let Ω be an arbitrary control volume with the surface $\partial\Omega$ and let $\mathbf{n}$ be the outward unit normal on a small area dS around a point P on $\partial\Omega$. If $\mathbf{u}(u, v, w)$ and ρ is the local velocity vector and density, respectively, then at point P the mass flow $\dot{m}$ through dS can be expressed

$$\dot{M} = \rho\mathbf{u} \cdot \mathbf{n} dS \tag{2.1}$$

The net flow of mass into Ω through $\partial\Omega$ is then given by

$$-\int_{\partial\Omega} (\rho\mathbf{u} \cdot \mathbf{n})\, dS \tag{2.2}$$

Applying the divergence theorem gives

$$-\int_{\partial\Omega} (\rho\mathbf{u} \cdot \mathbf{n})\, dS = -\int_{\Omega} \nabla \cdot (\rho\mathbf{u}) dV \tag{2.3}$$

For an infinitesimal volume element dV inside Ω the mass is given by ρdV, hence the total mass inside Ω is given by

$$\int_\Omega \rho \, dV \tag{2.4}$$

and thus the total net change of mass is given by

$$\frac{\partial}{\partial t} \int_\Omega \rho \, dV = \int_\Omega \frac{\partial \rho}{\partial t} \, dV \tag{2.5}$$

Then by the mass conservation principle

$$-\int_\Omega \nabla \cdot (\rho \mathbf{u}) dV = \int_\Omega \frac{\partial \rho}{\partial t} \, dV \Leftrightarrow \int_\Omega \Big[\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \Big] dV = 0 \tag{2.6}$$

Since Ω can be chosen arbitrary it follows that

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{2.7}$$

which is know as the *continuity equation*.

Momentum Equation The *momentum equation*, also known as the *balance equation of momentum*, is based on Newton's second law of motion, that is, the sum of all forces acting on a body is equal to the rate of change of momentum of it (*mass* $\times$ *acceleration*) [1]. The forces acting on the body can be divided into to groups. The first is the body forces $\mathbf{b}$ acting inside the body e.g. centrifugal or gravitational forces. The second group is the surface forces acting on the surface of the body e.g. pressure or viscous forces.

Similarly to the continuity equation the rate of change of total momentum inside Ω is given by

$$\frac{\partial}{\partial t} \int_\Omega \rho \mathbf{u} dV \tag{2.8}$$

Since Ω depend on t, applying Leibniz's integral rule and then the divergence theorem, Equation 2.8 can be rewritten according to

$$\frac{\partial}{\partial t} \int_\Omega \rho \mathbf{u} dV = \int_\Omega \frac{\partial (\rho \mathbf{u})}{\partial t} dV + \int_{\partial \Omega} \mathbf{u} \rho \mathbf{u} \cdot \mathbf{n} dS = \int_\Omega \frac{\partial (\rho \mathbf{u})}{\partial t} dV + \int_\Omega \nabla \cdot \rho \mathbf{u} \mathbf{u} dS \tag{2.9}$$

The forces acting on the surface $\partial \Omega$ is represented by the stress vector $\mathbf{T}$. The stress vector is obtained via the Cauchy stress tensor $\boldsymbol{\sigma}$ according to

$$\mathbf{T} = \mathbf{n} \cdot \boldsymbol{\sigma} \tag{2.10}$$

hence the total contribution to the total momentum from the surface forces are given by

$$\int_{\partial \Omega} \mathbf{n} \cdot \boldsymbol{\sigma} dS \tag{2.11}$$

where $\mathbf{n}$ is the outwards unit normal. Applying the divergence theorem to Equation 2.11 gives

$$\int_{\partial \Omega} \mathbf{n} \cdot \boldsymbol{\sigma} dA = \int_\Omega \nabla \cdot \boldsymbol{\sigma} dV \tag{2.12}$$

The total momentum due to body forces is given by

$$\int_\Omega \rho \mathbf{b} dV \tag{2.13}$$

where $\mathbf{b}$ is the body force per unit mass. The governing equation for momentum can then be expressed

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = \nabla \cdot \boldsymbol{\sigma} + \rho \mathbf{b} \tag{2.14}$$

Assuming isotropic Newtonian viscous fluids [18] the stress tensor $\boldsymbol{\sigma}$ can be divided into two parts, a *hydrostatic* part $\boldsymbol{\sigma}_{hyd}$ and a *deviatoric* part $\boldsymbol{\sigma}_{dev}$. The hydrostatic stress represents the stresses normal to the surface of the body , i.e the negative mechanical pressure P. The hydrostatic stress is defined as one third of the trace of $\boldsymbol{\sigma}$ times the unit tensor:

$$\boldsymbol{\sigma}_{hyd} = \frac{1}{3}\mathrm{tr}(\boldsymbol{\sigma})I = -P \tag{2.15}$$

The deviatoric stress consists of distorting stresses and are assumed to depend on the velocity gradient through the *rate of strain tensor* $\boldsymbol{D}$ according to

$$\boldsymbol{\sigma}_{dev} = 2\mu \mathrm{dev}(\boldsymbol{D}) \tag{2.16}$$

where

$$\boldsymbol{D} = \frac{1}{2}\big(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^{\top}\big) \tag{2.17}$$

The operator *dev* is the deviatoric tensor operator defined as

$$\mathrm{dev}\big(\boldsymbol{D}\big) = \boldsymbol{D} - \frac{1}{3}\mathrm{tr}\big(\boldsymbol{D}\big)\boldsymbol{I} \tag{2.18}$$

Expressed in terms of velocity and pressure gradients the Cauchy stress tensor then reads

$$\boldsymbol{\sigma} = -P\boldsymbol{I} + \mu\big(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^{\top} - \frac{2}{3}(\nabla \cdot \boldsymbol{u})\boldsymbol{I}\big) \tag{2.19}$$

Combining Equation 2.19 with Equation 2.14 gives the following version of the momentum equation

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = -\nabla P + \nabla \cdot \mu\big(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^{\top} - \frac{2}{3}\mathrm{tr}\big(\nabla \boldsymbol{u}\big)\boldsymbol{I}\big) + \rho \mathbf{b} \tag{2.20}$$

Energy Equation The *energy equation*, also known as the *balance equation of energy*, is based on the conservation of energy principle. This principle states that energy can neither be created nor destroyed, it can only change form. It follows from the first law of thermodynamics and states that the rate of change of total energy is equal to the rate heat is added minus the rate of work done [1].

The total energy is given by

$$\int_{\Omega} \rho\big(e_{in} + e_{kin}\big)dV \tag{2.21}$$

hence the rate of change of total energy is given by

$$\frac{\partial}{\partial t}\int_{\Omega} \rho\big(e_{in} + e_{kin}\big)dV = \int_{\Omega} \frac{\partial\big(\rho\big(e_{in} + e_{kin}\big)\big)}{\partial t}dV + \int_{\Omega} \nabla \cdot \boldsymbol{u}\rho\big(e_{in} + e_{kin}\big)dV \tag{2.22}$$

where e_{in} is the internal energy per unit mass and e_{kin} is the kinetic energy per unit mass.

The heat added to the system can be divided into two parts. The first part is the heat added to the system through the surface $\partial\Omega$ and is given by

$$-\int_{\partial\Omega} \boldsymbol{q}_s \cdot \boldsymbol{n}dS = -\int_{\Omega} \nabla \cdot \boldsymbol{q}_s dV \tag{2.23}$$

where $\boldsymbol{q}_s$ is the surface heat flux vector. The second part is the contribution from body heat sources q_b given by

$$\int_{\Omega} q_b dV \tag{2.24}$$

The rate of work done on the system can also be partitioned into two parts. The first part is work done due to surface forces, i.e. the pressure and viscous forces

$$\int_{\partial\Omega} P(\boldsymbol{u}\cdot\boldsymbol{n})dS - \int_{\partial\Omega}(\boldsymbol{\tau}\cdot\boldsymbol{u})\cdot\boldsymbol{n}dS = \int_{\Omega}\nabla\cdot(P\boldsymbol{u})dV - \int_{\Omega}\nabla\cdot(\boldsymbol{\tau}\cdot\boldsymbol{u})dV \tag{2.25}$$

where $\boldsymbol{\tau}$ is the shear stress tensor defined as $\text{dev}(\boldsymbol{\sigma})$. The second part accounts for the body forces $\boldsymbol{b}$, e.g. to gravitation, and is given by

$$\int_{\Omega}\rho(\boldsymbol{b}\cdot\boldsymbol{u})dV \tag{2.26}$$

Hence the governing equation for the total energy can be expressed

$$\frac{\partial(\rho(e_{in}+e_{kin}))}{\partial t} + \nabla\cdot\boldsymbol{u}\rho(e_{in}+e_{kin}) = -\nabla\cdot\boldsymbol{q}_s + q_b + \nabla\cdot(P\boldsymbol{u}) - \nabla\cdot(\boldsymbol{\tau}\cdot\boldsymbol{u}) + \rho(b\cdot\boldsymbol{u}) \tag{2.27}$$

The kinetic energy can also be obtained by taking the dot product of the momentum equation and the velocity vector. Subtracting the obtained expression for the kinetic energy from Equation 2.27 yields an equation for the internal energy e_{in}. Introducing the concept of enthalpy defined as

$$h = e_{in} + \frac{P}{\rho} \tag{2.28}$$

the energy equation can be expressed in terms of the specific enthalpy according to

$$\frac{\partial\rho h}{\partial t} + \nabla\cdot(\rho h\boldsymbol{u}) = -\nabla\cdot\boldsymbol{q}_s + q_b + \frac{\partial P}{\partial t} + P\nabla\cdot\boldsymbol{u} + \boldsymbol{\tau}:\nabla\boldsymbol{u} \tag{2.29}$$

where operator : is the double dot product [12, 18]. From the enthalpy, the temperature can be obtained through

$$T = \frac{h}{C_p} \tag{2.30}$$

assuming a calorically perfect gas [1].

2.1.3 Turbulence

In most engineering applications the flow of interest is turbulent, that is, properties can change in a chaotic way and there can be very large velocity gradients, pressure gradients and time fluctuations. There is no formal definition of a turbulent flow, but it has a number of characteristics, see [7]:

Irregular The flow is irregular and chaotic but is governed by the momentum equation Equation 2.20. The flow consist of eddies which are not defined in detail but are assumed to have three characteristic scales: length (diameter), velocity and time of existence. The largest scales are restricted by the geometry in which the flow evolves. The largest eddies are assumed to get their energy from the mean flow and pass it to smaller eddies in the so called cascade process. The smallest scale eddies are assumed to be destroyed by dissipation.

Dissipation: The destruction of the small scaled eddies is called turbulent dissipation. After the energy is transferred to the smallest eddies in the cascade process it is transformed into thermal energy by the turbulent dissipation.

Three-dimensional: Turbulence always occurs in three dimension and is unsteady. If the governing equations are time averaged it can be treated as two-dimensional.

Diffusivity: Turbulence increase the diffusivity in a flow. Hence it increases the exchange of momentum in boundary layers and increases the wall friction and heat transfer.

Continuum: The smallest turbulent scales are much larger than the atomic scales hence the principles of continuum mechanics are valid.

High Reynolds numbers: Turbulence only occurs at high Reynold numbers. The Reynolds number is a dimensionless quantity that can be viewed as a comparison of the inertial forces to the viscous forces.

Reynolds Average Navier-Stokes Equations Solving all scales in a turbulent flow is only possible in very simple cases with the computer power available today. For most practical purposes it is enough to calculate the time-averaged properties of a flow. This can be solved by solving a time-averaged version of the Navier-Stokes equations, the theory to derive such an equation can be found in any basic book in the subject like [7, 29] that were used in the following derivations. To do this a property ζ of a flow is divided into a time-averaged part $\overline{\zeta}$ and a fluctuating part ζ' according to

$$\zeta = \overline{\zeta} + \zeta' \tag{2.31}$$

The first step is to expressing the properties in the continuity equation Equation 2.7 and momentum equation Equation 2.20 according to Equation 2.31. Then taking the time average of each equation yields the so-called *Reynolds averaged Navier-Stokes equations* (RANS). After some algebra and dropping the time average bar the RANS equations states

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{2.32}$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = -\nabla P + \nabla \cdot \mu\big(\nabla \boldsymbol{u} + \big(\nabla \boldsymbol{u}\big)^\top - \frac{2}{3}\mathrm{tr}\big(\nabla \boldsymbol{u}\big)\boldsymbol{I}\big) + \rho \mathbf{b} + \nabla \cdot \big(- \rho \overline{\boldsymbol{u'} \otimes \boldsymbol{u'}}\big) \tag{2.33}$$

where $\otimes$ is the outer product. The last term in Equation 2.33 is called the *Reynold stresses* and account for the turbulent part of the flow. This term need to be modelled to get a valid representation of a turbulent flow.

Turbulence Modelling To model the Reynold stresses several methods are available. In this thesis the standard $k - \epsilon$-model [7, 12, 29] will be used which is based on the *Boussinesq assumption* [31], which reads

$$-\overline{u' \otimes u'} = \nu^t\big(\nabla \boldsymbol{u} + \big(\nabla \boldsymbol{u}\big)^\top\big) - \frac{2}{3}\big(k + \nu^t \mathrm{tr}\big(\nabla \boldsymbol{u}\big)\big)\boldsymbol{I} \tag{2.34}$$

where k is the turbulent kinetic energy and ν^t is a proportionality constant known as the *eddy viscosity*. Both these properties have to be modelled. The turbulent kinetic energy is modelled with the equation

$$\frac{\partial \rho k}{\partial t} + \nabla \cdot (\rho k \boldsymbol{u}) = \nabla \cdot \Big(\rho\Big(\nu + \frac{\nu^t}{\sigma_k}\Big)\nabla k\Big) + 2\rho\nu^t\big(\nabla \boldsymbol{u} + \big(\nabla \boldsymbol{u}\big)^\top\big) \cdot \nabla \boldsymbol{u} - \rho\epsilon \tag{2.35}$$

where ϵ the dissipation mentioned in 2.1.3. The turbulent dissipation is calculated with the governing equation

$$\frac{\partial \rho \epsilon}{\partial t} + \nabla \cdot (\rho \epsilon \boldsymbol{u}) = \nabla \cdot \Big(\rho\Big(\nu + \frac{\nu^t}{\sigma_\epsilon}\Big)\nabla \epsilon\Big) + C_{1\epsilon}\frac{\epsilon}{k}2\rho\nu^t\big(\nabla \boldsymbol{u} + \big(\nabla \boldsymbol{u}\big)^\top\big) \cdot \nabla \boldsymbol{u} - C_{2\epsilon}\rho\frac{\epsilon^2}{k} \tag{2.36}$$

With k and ϵ obtained from Equation 2.35 and Equation 2.36 the eddy viscosity can be calculated according to

$$\nu^t = C_\mu \frac{k^2}{\epsilon} \tag{2.37}$$

The equations contain five adjustable constants $C_\mu, \sigma_k, \sigma_\epsilon, C_{1\epsilon}, C_{2\epsilon}$ which usually are set to $0.09, 1.00, 1.30, 1.44$ and 1.92 respectively.

Near Wall Treatment of Turblent Quantities The area close to the wall need to be handled in a specific way to accurately predict the velocity. The region closest to the wall can be divided into a number of smaller layers. The distance to the wall is usually expressed in the non-dimensional unit y^+ depending on the turbulent kinetic energy [7] according to

$$y^+ = \frac{C_\mu^{1/4} k^{1/2} y}{\nu} \tag{2.38}$$

where C_μ is the constant mentioned in section 2.1.3 equal to 0.09. The velocity of significance close to the wall region is the streamwise velocity denoted u^+ which is defined in terms of y^+.

In the region adjacent to the wall, approximately $y^+ \lesssim 5$, the flow is considered laminar and the turbulent quantities are set to zero and the velocity is approximated by $u^+ = y^+$. The region $30 \lesssim y^+ \lesssim 3000$ is called the *log-law-region* in which the velocity is approximated by

$$u^+ = \frac{1}{\kappa} \ln\left(E y^+\right) \tag{2.39}$$

The constants E is usually set to 9.793 and κ is Von Karman's constant which is equal to 0.4187 according to [8]. The region between the viscous region and the log-law region is called the buffer layer. In this region there is a transition of the velocity from being well approximated by $u^+ = y^+$ to being well approximated by Equation 2.39. At $y^+ \approx 11$ both these laws are equal and a switch is usually carried out in the model.

2.2 Multi-Phase Fluid Mechanics

In multi-phase fluid mechanics flows consisting of materials in different phases are subject to study. The phases can be gas, liquid , solid or materials of different chemical properties as water and oil.

An example of a system with multi-phase flow is a pipe with heated walls where enough heat is added for vapour bubbles to develop. Such a boiling system is classified as a two-phase gas-liquid flow and the single-phase framework is no longer sufficient to describe it. The theory presented in this section only concern two-phase flow but could easily be expanded to any number of phases. The theory in this thesis will be based on the so-called Euler-Euler model [8].

2.2.1 Euler-Euler Model

The Euler-Euler model is an approach that describe a two-phase (or multi-phase) system were each phase is represented by a complete set of governing equations. With the appropriate closure equations and equations for turbulence modelling such a system can be described properly. The governing equations are similar to Equation 2.7, Equation 2.20 and Equation 2.29, with some modifications. The modifications introduces new concepts as void fractions and accurate representation of the terms for exchange of mass, momentum and energy between the phases.

Void Fraction The void fraction α_k is defined for each phase k as a number between 0 and 1 giving the fraction of space occupied by phase k. Consequently the sum of all void fractions equal 1. The void fractions can be solved for each phase through the continuity equation. In a two-phase flow system it follows that $\alpha_k = 1 - \alpha_i$, hence it is enough to solve for one void fraction. The subscripts k and i represent the two separate phases.

Continuity Equation The adjusted continuity equation of each phase k according to [10, 11, 14, 20, 21] reads

$$\frac{\partial \alpha_k \rho_k}{\partial t} + \nabla \cdot \left(\alpha_k \rho_k \boldsymbol{u}_k\right) = \Gamma_{ki} - \Gamma_{ik} \tag{2.40}$$

where Γ_{ik} account for the mass transferred from phase k to phase i and Γ_{ki} the mass transferred from i to k. How the mass transfer terms are defined depends on the physical system that is modelled and the choice of model.

Momentum Equation Similarly to the continuity equation the momentum equation is also adjusted by the void fraction. Additional terms are also added, to account for the exchange of momentum due to phase transfer between the phases. A general version of the momentum equation for a multi-phase flow with the Boussinesq assumption according to [10, 14, 20] reads

$$\frac{\partial \alpha_k \rho_k \boldsymbol{u}_k}{\partial t} + \nabla \cdot \left(\alpha_k \rho_k \boldsymbol{u}_k \boldsymbol{u}_k\right) = -\alpha_k \nabla P + \nabla\left(\alpha_k\left(\tau_k + \tau_k^t\right)\right) + \Gamma_{ki}\boldsymbol{u}_i - \Gamma_{ik}\boldsymbol{u}_k + \alpha_k \rho_k \boldsymbol{g} + F_k \quad (2.41)$$

The second term on the R.H.S. is the effect of the Reynolds viscous stresses and the turbulent stresses. The term $\tau_k + \tau_k^t$ is the combined Reynolds viscous and turbulent stresses given by

$$\tau_k + \tau_k^t = \rho_k \nu_k^{eff}\left(\nabla \boldsymbol{u}_k + \left(\nabla \boldsymbol{u}_k\right)^\top - \frac{2}{3}\boldsymbol{I}\nabla \cdot \boldsymbol{u}_k\right) - \frac{2}{3}\boldsymbol{I}\rho_k k_k \quad (2.42)$$

where $\boldsymbol{I}$ is the identity tensor and k is the turbulent kinetic energy. The term ν_k^{eff} is the effective kinematic viscosity and is the sum of different viscosities of the flow. The third and the fourth term accounts for the loss and gain of momentum due to the phase change. The fifth term is gravitational term and the last term F_k on the R.H.S. accounts for the interfacial forces and is subject for further modelling. The gravitation and the interfacial forces are assumed to be the only body forces.

Energy Equation As with the single-phase flow the energy can be expressed in terms of different quantities. The modified energy equation for the two-phase flow will also be given in terms of enthalpy. If the effects of the shear stress to the energy conservation is neglected the enthalpy equation states [14]

$$\begin{aligned}\frac{\partial \alpha_k \rho_k h_k}{\partial t} + \nabla \cdot \left(\alpha_k \rho_k \boldsymbol{u}_k h_k\right) &= -\nabla \cdot \alpha_k\left(\boldsymbol{q}_k + \boldsymbol{q}_k^t\right) \\ &+ \alpha_k \frac{\partial P}{\partial t} + \alpha_k \boldsymbol{u}_k \cdot \nabla P + \Gamma_{ki} h_i - \Gamma_{ik} h_k + q_b \end{aligned} \quad (2.43)$$

The first term $\boldsymbol{q}_k + \boldsymbol{q}_k^t$ account for the sum of the molecular and turbulent heat fluxes of each phase. The second and third term is the material derivative of the pressure. The fourth and fifth term accounts for the contribution of enthalpy from phase change. The last term is a the source term for body sources of enthalpy.

2.2.2 Interfacial Forces

In Equation 2.41 the interfacial forces account for the forces arising at the interface between the phases. A number of such forces with varying impact on the solution can be considered for a boiling flow. A few examples are

Drag force: The drag force is the most contributing force to the momentum. It is the force felt by an object, like a bubble or drop, when it moves steadily through the surrounding fluid. There exist several formulations of this force, including the so-called *drag force coefficient* of this force. The formulations often focuses on how to model the drag force coefficient.

Virtual mass force: When an object accelerates through a fluid it accelerates some of the surrounding liquid. This results in an interaction force called the virtual mass force.

Phase change force: The force due to phase change accounts for the effect on the momentum at the interface when the state of a phase changes to another state.

Wall lubrication force: In case of boiling at a heated wall the wall lubrication force acts normal to the surface and preventing the generated vapour bubbles from accumulating.

Turbulent dispersion force: The turbulent dispersion force accounts for the effects from the fluctuating turbulent velocities on the vapour bubbles.

Shear lift force: The shear lift force is a force that acts on bubbles moving in a flow and pushes smaller bubbles to low-velocity regions, i.e. to the wall and larger bubbles to the regions with higher velocities.

2.2.3 Interfacial Area Concentration

The interfacial area concentration a_i is defined as the total interfacial area between two phases divided by the total volume of the system, that is, for a boiling system the area of vapour bubbles per unit volume. This quantity can be modelled with more advanced modelling including effects of bubble break-up and bubbles merging. The advanced models can be carried out with a separate transport equation, to be solved in parallel with the adapted model. Simpler models can be adopted if these effects are neglected and the average bubble size is given. In this case the interfacial area concentration can easily be obtained from the void fraction and the diameter according to

$$a_i = 6\frac{\alpha_k}{d_k} \tag{2.44}$$

where d_k denotes the average bubble diameter for phase k.

2.3 Heat Transfer

In heat transfer the exchange of thermal energy in a physical system is the subject of study. Heat exchange always occur when there exist a temperature difference in a medium or between media. Heat transfer is categorized depending on the conditions in which it take place [13].

Conduction: Conduction refers to the heat exchange taking place in stationary mediums as solids or fluids.

Convection: Convection describes the heat transfer between a surface and a moving fluid or gas, if they are at different temperatures.

Radiation: Radiation is the heat transfer due to emitted energy in the form of electromagnetic waves from a heated surface.

In convection heat transfer the fluid motion is the dominating mechanism that transfer the heat. Convection can further be divided into *free convection* and *forced convection*. In a forced convection system the fluid motion is maintained by external force, e.g. a pump or a fan whereas in free convection it is maintained by buoyancy forces. The system that will be modelled in this thesis consist of forced convection in a vertical pipe with heated wall. So far these terms only describe heat transfer in single-phase systems, i.e. no phase mass transfer like boiling or condensation is accounted for.

2.4 Boiling

If enough heat is added to a liquid in a system a phase transfer from the liquid phase to the gas phase will occur. This phenomena is called boiling and can be divided into two categories:

Pool boiling: In pool boiling the fluid is at rest and the only motion taking place is due to free convection and to bubbles growing and detaching. This corresponds to the single-phase free convection conditions.

Forced convection boiling: In forced convection boiling the fluid motion is induced by free convection, bubbles growth and by external means. It corresponds to the single-phase forced convection conditions.

The boiling heat transfer is divided into regimes depending on the behaviour of the boiling. It is illustrated in Figure 2.1 on a $log(q'') - log(\Delta T_{sup})$ plane where ΔT_{sup} is the wall super heat defined as the difference between the temperature of the heated surface and the saturation temperature for the system. The heat flux q'' represent the heat transferred to the system at different ΔT_{sup}.

In the first regime, up to point B, no boiling occurs and heat transfer only consist of convection. After point B until point C small isolated vapour bubbles are developed at the heated surface which detach and condense in the bulk liquid. This regime is referred to as *subcooled nucleate boiling* and will be subject of study in this thesis. The point B at which boiling starts is referred to as *onset of nucleate boiling* (ONB). Between point C and E increasing number of larger bubbles develop and start to merge with each other and form columns of vapour. This regime is called *saturated nucleate boiling*.

The next regime is the *transition boiling*, between point E and G. In this region bubbles grow so fast that a film of vapour is created over the surface. Since the thermal conductivity of vapour is much smaller than for the liquid the heat transferred to the system decreases in this regime. After point G the *stable film boiling* regime starts. In this regime the surface is totally covered by vapour and the heat is transferred though conduction and radiation through the vapour.

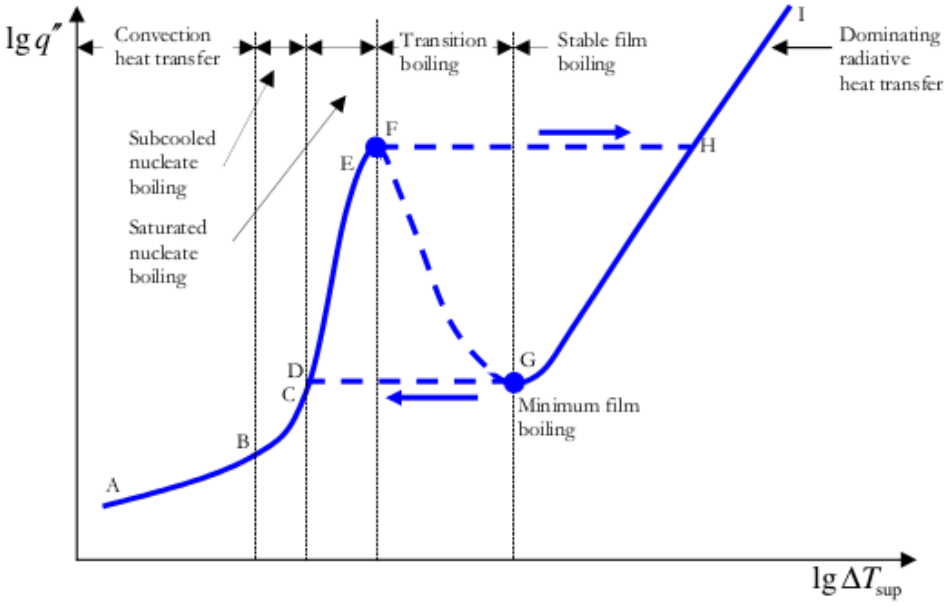

Figure 2.1: Boiling curve illustrating different boiling regimes as the wall super heat at different heat fluxes from [12].

2.5 Non-Dimensional Numbers

In the study of systems with turbulent flow and heat transfer there are many ways to estimate basic concepts. For this purpose there exist a wide range of dimensional numbers that describe the different properties of a system [13]. The theory in this thesis will make use of the following dimensionless numbers:

Reynolds number: The Reynolds number (Re) is defined as the ratio between inertial forces and viscous forces. It can be calculated by $\mathrm{Re} = \overline{\boldsymbol{u}}l/\nu$, were $\overline{\boldsymbol{u}}$ is the mean velocity of the fluid and l is the characteristic length, e.g. pipe diameter. The Reynolds number is often used to estimate if a flow is turbulent or laminar.

Prandtl number: The Prandtl number (Pr) gives the ratio between momentum diffusivity, i.e. kinematic viscosity, and thermal diffusivity. It can be calculated according to $\mathrm{Pr} = \nu/\chi = C_p\mu/\lambda$.

Nusselt number: The Nusselt number (Nu) is defined as the ratio between convective and conductive heat transfer across a boundary, that is, it is the dimensionless temperature gradient normal to the heated surface. It can be calculated by $\mathrm{Nu} = Hl/\lambda$ where H is the convective heat transfer coefficient and λ is the thermal conductivity.

Stanton number: The Stanton number (St) is defined as the ratio between heat transferred into a fluid and the thermal capacity of the fluid. It can be calculated by $\mathrm{St} = H/(\rho\overline{\boldsymbol{u}}C_p)$ where C_p is the specific heat capacity of the fluid.

2.6 Finite Volume Method

The finite volume method is a technique to go from continuous differential equation to a discrete algebraic equation[7, 29].The fundamental idea is to divide the computational domain into a number of smaller control volumes of geometrical shapes like cuboid or tetrahedrons. This is done by defining a number of node points in the domain and then divide the domain between the node points resulting in a structure called *mesh*. In the solution procedure, the refinement of the mesh is a compromise between the solution accuracy and the computational cost.

The differential equation is then discretized by integration over each control volume. The integrations make use of Gauss theorem stating that

$$\int_{\Omega} \nabla \cdot \Psi dV = \oint_{\partial\Omega} \Psi \cdot \boldsymbol{n} dS \tag{2.45}$$

With this technique the integrated terms of the differential equation only needs to be given at the boundary of the control volume. The boundary is divided into faces defined as parts of the boundary where the normal do not change. Since the values at the faces are normally not known they are interpolated from the node points. This is done with an interpolation scheme of which there exist a wide range to chose from depending on what is suitable for the equation to be solved.

As an example, to solve the one-dimensional differential equation

$$\frac{d^2\psi}{dx^2} + S = 0 \tag{2.46}$$

over a domain D first divide D into a suitable number of control volumes, see Figure 2.2. At each control volume by Gauss theorem it follows that

$$\int_{\Omega} \Big[\frac{d}{dx}\Big(\frac{d\psi}{dx}\Big) + S\Big] dV = \int_{\Omega} \Big[\frac{d}{dx}\Big(\frac{d\psi}{dx}\Big)\Big] dV + \int_{\Omega} S dV =$$

$$\oint_{\partial\Omega} \Big(\frac{d\psi}{dx}\Big) dS + \overline{S}\Delta x = \Big(\frac{d\psi}{dx}\Big)_e - \Big(\frac{d\psi}{dx}\Big)_w + \overline{S}\Delta x = 0 \tag{2.47}$$

The subscripts e and w corresponds to the faces between node P and node E and W respectively and $\overline{S}$ is the average of S over the control volume. To achieve the finial discretized equation the first order derivative terms need to be interpolated on the faces in terms of the node points. One such scheme is the central differencing scheme which gives

$$\Big(\frac{d\psi}{dx}\Big)_e = \frac{\psi_E - \psi_P}{\delta x_e}\,,\ \Big(\frac{d\psi}{dx}\Big)_w = \frac{\psi_P - \psi_W}{\delta x_w} \tag{2.48}$$

Inserting Equation 2.48 in the expression in Equation 2.47 and rearranging yields

$$a_E\psi_E - a_P\psi_P + a_W\psi_W = -S_u \tag{2.49}$$

where

$$a_E = \frac{1}{\delta x_e}\,,\ a_W = \frac{1}{\delta x_w}\,,\ a_P = a_E + a_W.$$

Performing the previous discretization at each node gives a linear system of equation, with an equation per node. Such a system of equation can be solved by existing algorithms like TDMA and Gauss-Seidel.

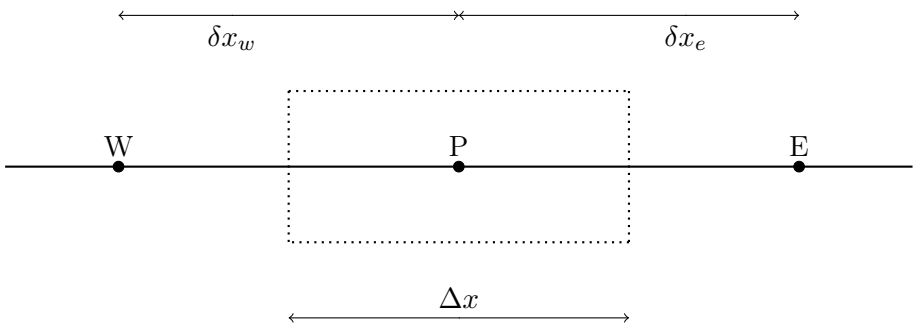

Figure 2.2: An example of a one-dimensional control volume.

3 Boiling Model

In this section a model of a subcooled nucleate boiling system is specified. The model that was used to simulate the boiling is based on the model published by Kurul *et al.* [21]. It is based on the two-fluid model Eulerian-Eulerian approach, where each phase is modelled with its own set of transport equations. The two phases are distinguished by the subscripts L and V representing Liquid and Vapour, respectively.

The system, subject for examination was a vertical pipe with an upward flow of water, heated from the wall to the point where phase change, in the form of evaporation, started at the wall. The evaporation was in the form of subcooled nucleate boiling. The vapour, in the form of bubbles, detaching from the wall then condensates in the subcooled liquid.

The momentum of the fluid and the vapour was modelled with the momentum equation section 3.2.2. Factors contributing to the momentum were interfacial forces and turbulence, as described in section 3.3 and section 3.4.

The propagation of heat was calculated by the heat equation described in section 3.2.3. The accurate temperature at the wall had to be calculated separately with the law-of-the-wall explained in section 3.5.2.

The boiling phenomena was modelled to start at a certain global condition described in section 3.5. Once boiling has started at the wall, the total heat flux from the wall is partitioned into two parts as explained in section 3.6. One part that goes into a continued heating of the liquid and one part that is consumed in the phase change from liquid to vapour.

The heat flux, going into the vapour phase, was calculated with the diameter of the vapour bubbles detaching from the wall, the number of bubbles growing on the wall per unit area and the frequency at which bubbles were developed. These models are described in section 3.7.

The mass transfer term describing the evaporation was calculated as a quantity depending on the heat flux going into the vapour phase. The model for this mass transfer is described in section 3.8.1.

When the vapour bubbles drifted into the subcooled bulk they were cooled down and condensed back to liquid. This phase change was modelled with the condensation mass transfer term described in section 3.8.2.

The condensation rate was dependent on the size of the bubbles in the bulk flow. This requires a model for the bubble diameter in the bulk, which is described in section 3.9.

In Figure 3.1 the internal flow in a vertical pipe with walls heated, to the point where boiling starts, is depicted.

3.1 Assumptions and Simplifications

The model consist of two key parts. The first part is the model of the departure diameter of the bubbles published by Ünal [32]. The second part is the model of the wall super heat according to Jens-Lottes correlation [12].

To obtain a solvable model a number of assumptions were made to simplify it.

1. The liquid phase was considered incompressible. Though in reality the density changes with temperature.

2. Properties like viscosity and thermal conductivity were also assumed to be constant, despite the temperature differences.

3. The vapour phase was assumed to be at saturated conditions, that is, the vapour was assumed to always be at saturated temperature and pressure. Consequently the vapour phase was treated as incompressible. This also simplified the way the condensation was modelled, since no cooling of the vapour inside the bubbles had to be considered.

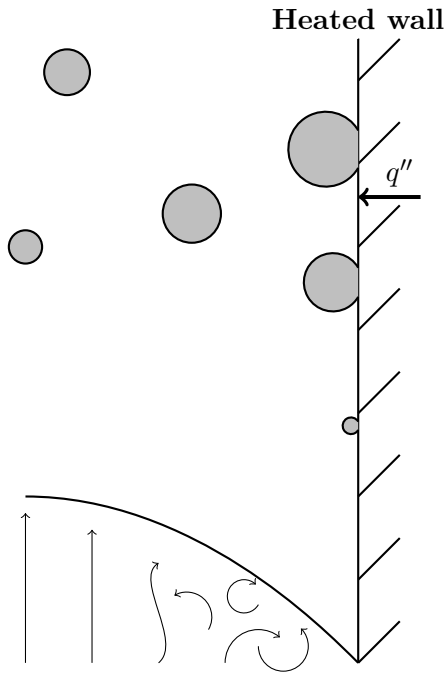

Figure 3.1: Graphical illustration of a vertical pipe with a wall, heated to the point that boiling starts.

4. The vapour phase was assumed to be laminar, i.e. effects of turbulence inside the bubbles were neglected
5. The bubbles were assumed to be perfect spheres to simplify the modelling of the bubble dimensions.
6. Single bubbles were assumed, hence, no effects caused by bubbles merging or breaking up were accounted for.

3.2 Governing Equations

The governing equations used in the model is based on the equations formulated in section 2.2. With the above assumptions, the appropriate governing equations were formulated according to the following sections.

3.2.1 Continuity Equation

Since the flow was assumed to be incompressible, the continuity equations were only used as transport equations for the void fraction and for the pressure correction algorithm. The equations read

$$\frac{\partial \rho_L \alpha_L}{\partial t} + \nabla \cdot (\rho_L \alpha_L \mathbf{u}_L) = \Gamma_L \tag{3.1}$$

for the liquid phase and

$$\frac{\partial \rho_V \alpha_V}{\partial t} + \nabla \cdot (\rho_V \alpha_V \mathbf{u}_V) = \Gamma_V \tag{3.2}$$

for the vapour phase. The source term Γ_V is given by evaporation rate subtracted by the condensation rate $\Gamma_{evap} - \Gamma_{cond}$ and $\Gamma_L = -\Gamma_V$.

To obtain the void fractions it was enough to solve the transport equation for α_V. Since there were only two phases α_L could be obtained through

$$\alpha_L = 1 - \alpha_V \tag{3.3}$$

Hence, the only equation to be solved for the void fraction were Equation 3.2.

3.2.2 Momentum Equation

For each phase a separate momentum equation was formulated. Since the vapour phase was assumed to be laminar no turbulent quantities were included, in the corresponding momentum equation. Moreover, since the liquid phase was treated as incompressible and the vapour phase was assumed to be saturated and, hence, incompressible the density could be moved to the R.H.S of the equations. The governing equation for the vapour phase reads as

$$\frac{\partial \alpha_V \boldsymbol{u}_V}{\partial t} + \nabla \cdot \left(\alpha_V \boldsymbol{u}_V \boldsymbol{u}_V\right) = -\frac{\alpha_V}{\rho_V}\nabla P + \frac{1}{\rho_V}\nabla\left(\alpha_V \tau_V\right) + \alpha_V \boldsymbol{g} + \frac{1}{\rho_V} F_V \tag{3.4}$$

where

$$\tau_V = \rho_V \nu_V \Big(\nabla \boldsymbol{u}_V + \left(\nabla \boldsymbol{u}_V\right)^\top - \frac{2}{3}\boldsymbol{I}\nabla \cdot \boldsymbol{u}_V\Big) \tag{3.5}$$

For the liquid phase, which accounted for turbulent effects, the momentum equation stated

$$\frac{\partial \alpha_L \boldsymbol{u}_L}{\partial t} + \nabla \cdot \left(\alpha_L \boldsymbol{u}_L \boldsymbol{u}_L\right) = -\frac{\alpha_L}{\rho_L}\nabla P + \frac{1}{\rho_L}\nabla\left(\alpha_L\left(\tau_L + \tau_L^t\right)\right) + \alpha_L \boldsymbol{g} + \frac{1}{\rho_L} F_L \tag{3.6}$$

where

$$\tau_L + \tau_L^t = \rho_L \nu_L^{eff}\Big(\nabla \boldsymbol{u}_L + \left(\nabla \boldsymbol{u}_L\right)^\top - \frac{2}{3}\boldsymbol{I}\nabla \cdot \boldsymbol{u}_L\Big) - \frac{2}{3}\boldsymbol{I}\rho_L k_L \tag{3.7}$$

Both equations account for the stress due to pressure through the first term on the R.H.S. of Equation 3.4 and Equation 3.6. The second term in Equation 3.4 represents the viscous stresses and in Equation 3.6 it stands for the viscous and turbulent stresses. The third and fourth term are the gravitational force and the interfacial forces. The last two terms needed further modelling to achieve a closed system of equations.

3.2.3 Energy Equation

Since the vapour phase is assumed to be at saturated conditions no energy equations needs to be solved for it. For the liquid phase the energy equation is stated in terms of the specific enthalpy according to

$$\begin{aligned}\frac{\partial \alpha_L h_L}{\partial t} + \nabla \cdot \left(\alpha_L \boldsymbol{u}_L h_L\right) = -\frac{1}{\rho_L}\nabla \cdot \left(\alpha_L\left(\boldsymbol{q}_L + \boldsymbol{q}_L^t\right)\right) + \frac{\alpha_L}{\rho_L}\frac{\partial P}{\partial t} + \frac{\alpha_L}{\rho_L}\boldsymbol{u}_L \cdot \nabla P \\ + \frac{\Gamma_{cond} h_{V,sat} - \Gamma_{evap} h_L}{\rho_L} + \frac{\boldsymbol{q}_{wall} A_{wall}}{\rho_L}\end{aligned} \tag{3.8}$$

Comparing to Equation 2.43, the source term has been specified as the wall heat flux in Equation 3.13. Using Fourier's law

$$\boldsymbol{q} = -\lambda \nabla T = -\frac{\lambda}{C_p}\nabla h \tag{3.9}$$

it follows that

$$\boldsymbol{q}_L = -\lambda_L \nabla T_L = -\frac{\lambda_L}{C_{p,L}}\nabla h_L \tag{3.10}$$

$$\boldsymbol{q}_L^t = -\lambda_L^t \nabla T_L = -\frac{\lambda_L^t}{C_{p,L}}\nabla h_L \tag{3.11}$$

where the turbulent thermal conductivity λ_L^t can be obtained through the Prandtl number section 2.5 according to

$$\lambda_L^t = \frac{\rho_L C_{p,L} \nu_L^t}{\mathrm{Pr}_L^t} = \frac{\rho_L C_{p,L}(\nu_L^{eff} - \nu_L)}{\mathrm{Pr}_L^t} \tag{3.12}$$

where ν_L^{eff} is the effective kinematic viscosity. The turbulent Prandtl number Pr_L^t was set to 0.85, which is within the range of common values obtained form different empirical and analytical relations as in [2, 15]. Inserting Equation 3.10, Equation 3.11 and Equation 3.12 in Equation 3.8 gives the following energy equation

$$\frac{\partial \alpha_L h_L}{\partial t} + \nabla \cdot \big(\alpha_L \boldsymbol{u}_L h_L\big) = \nabla \cdot \big(\alpha_L \big(\frac{\lambda_L}{\rho_L C_{p,L}} + \frac{1}{Pr_L^t}\big(\nu_L^t - \nu_L\big)\nabla h_L\big)\big)$$
$$+ \frac{\alpha_L}{\rho_L}\frac{\partial P}{\partial t} + \frac{\alpha_L}{\rho_L}\boldsymbol{u}_L \cdot \nabla P + \frac{\Gamma_{cond} h_{V,sat} - \Gamma_{evap} h_L}{\rho_L} + \frac{\boldsymbol{q}_{wall} A_{wall}}{\rho_L} \tag{3.13}$$

3.3 Turbulence Model

Since turbulence was only taken into account for the liquid phase, only one set of turbulence equations were needed. The turbulence equations were taken from the OpenFOAM's multi-phase version of the $k-\epsilon$ turbulence model. It is based on the original $k-\epsilon$-model, adjusted with the void fraction and suitable near wall treatment.

3.3.1 Multi-Phase Turbulence Model

The $k-\epsilon$ turbulence model in OpenFOAM can be derived from the standard $k-\epsilon$-model by first. The multi-phase turbulence can be modelled with the standard equations [17]

$$\frac{\partial k_k}{\partial t} + (\boldsymbol{u}_L \cdot \nabla) k_k = \nabla \cdot \Big(\Big(\frac{\nu_L^{eff}}{\sigma_{k_k}}\Big)\nabla k_k\Big) + G - \epsilon_k \tag{3.14}$$

$$\frac{\partial \epsilon_k}{\partial t} + (\boldsymbol{u}_L \cdot \nabla) \epsilon_k = \nabla \cdot \Big(\Big(\frac{\nu_L^{eff}}{\sigma_{\epsilon_k}}\Big)\nabla \epsilon_k\Big) + C_{1\epsilon_k}\frac{\epsilon_k}{k_k} G - C_{2\epsilon_k} \tag{3.15}$$

where $\nu_L^{eff} = \nu_L + \nu_L^t$ and G is the production of k caused by viscous forces and reads

$$G = \nu_L^t \big(\nabla \boldsymbol{u}_L : \mathrm{dev}\big(\nabla \boldsymbol{u}_L + \big(\nabla \boldsymbol{u}_L\big)^\top\big)\big) \tag{3.16}$$

It is suggested that multi-phase turbulence can be solved with the equations [2]

$$\frac{\partial \rho_L \alpha_L k_L}{\partial t} + \nabla \cdot (\rho_L \alpha_L k_L \boldsymbol{u}_L) =$$
$$\nabla \cdot \Big(\rho_L \alpha_L \Big(\nu + \frac{\nu_L^t}{\sigma_{k_L}}\Big)\nabla k_L\Big) + \alpha_L \rho_L P_L - \alpha_L \rho_L \epsilon_L \tag{3.17}$$

for the turbulent kinetic energy and

$$\frac{\partial \rho_L \alpha_L \epsilon_L}{\partial t} + \nabla \cdot (\rho_L \alpha_L \epsilon_L \boldsymbol{u}_L)$$
$$= \nabla \cdot \Big(\rho_L \alpha_L \Big(\nu + \frac{\nu_L^t}{\sigma_{\epsilon_L}}\Big)\nabla \epsilon_L\Big) + C_{1\epsilon_L}\rho_L \alpha_L \frac{\epsilon_L}{k_L} P_L - C_{2\epsilon_L}\rho_L \alpha_L \frac{\epsilon_L^2}{k_L} \tag{3.18}$$

for the dissipation. Contributions to k and ϵ, due to inter-phase transfer, have been omitted in this model. P_L is the production term of turbulent kinetic energy due to viscous forces. It is defined as

$$P_L = \nu_L^t \big(\nabla \boldsymbol{u}_L + \big(\nabla \boldsymbol{u}_L\big)^\top\big)\nabla \boldsymbol{u}_L - \frac{2}{3}\nabla \boldsymbol{u}_L \big(k + \nu_L^t \nabla \cdot \boldsymbol{u}_L\big)\boldsymbol{I} \tag{3.19}$$

where a factor 3 in the last term, due to frozen stress, has been neglected in accordance with [2]. The production term P_L can be rewritten in terms of the production term G [17]. To do this it is first noted that

$$G = \nu_L^t \big(\nabla \boldsymbol{u}_L : \mathrm{dev}\big(\nabla \boldsymbol{u}_L + \big(\nabla \boldsymbol{u}_L\big)^\top\big)\big)$$

$$= \nu^t\big(\nabla \boldsymbol{u}_L + \big(\nabla \boldsymbol{u}_L\big)^\top\big)\nabla \boldsymbol{u}_L - \frac{2}{3}\nu^t \nabla \boldsymbol{u}_L\big(\nabla \cdot \boldsymbol{u}_L\big)\boldsymbol{I} \tag{3.20}$$

hence it follows that

$$P_L = \nu_L^t\big(\nabla \boldsymbol{u}_L : \mathrm{dev}\big(\nabla \boldsymbol{u}_L + \big(\nabla \boldsymbol{u}_L\big)^\top\big)\big) - \frac{2}{3}\nabla \boldsymbol{u}_L k = G - \frac{2}{3}\nabla \boldsymbol{u}_L k \tag{3.21}$$

Inserting the obtained expressions for the product term P_L into Equation 3.17 and Equation 3.18 gives the final version of the $k-\epsilon$-equations used in the boiling model:

$$\frac{\partial \rho_L \alpha_L k_L}{\partial t} + \nabla \cdot (\rho_L \alpha_L k_L \boldsymbol{u}_L) =$$

$$\nabla \cdot \Big(\rho_L \alpha_L \Big(\nu + \frac{\nu_L^t}{\sigma_{k_L}}\Big)\nabla k_L\Big) + \alpha_L \rho_L G - \alpha_L \rho_L \frac{2}{3}\nabla \boldsymbol{u}_L k - \alpha_L \rho_L \epsilon_L \tag{3.22}$$

for the turbulent kinetic energy and

$$\frac{\partial \rho_L \alpha_L \epsilon_L}{\partial t} + \nabla \cdot (\rho_L \alpha_L \epsilon_L \boldsymbol{u}_L) = \nabla \cdot \Big(\rho_L \alpha_L \Big(\nu + \frac{\nu_L^t}{\sigma_{\epsilon_L}}\Big)\nabla \epsilon_L\Big)$$

$$+ C_{1\epsilon_L} \rho_L \alpha_L \frac{\epsilon_L}{k_L} G - C_{1\epsilon_L} \rho_L \alpha_L \frac{\epsilon_L}{k_L}\frac{2}{3}\nabla \boldsymbol{u}_L k - C_{2\epsilon_L} \rho_L \alpha_L \frac{\epsilon_L^2}{k_L} \tag{3.23}$$

for the turbulent dispersion.

3.3.2 Near-Wall Treatment of Turbulent Quantities

The area close to the wall needs special attention to accurately predict the velocity as mentioned in section 2.1.3. The near wall treatment used in this model was chosen from the existing models in OpenFOAM. The *kqRWallFunction* was used for the turbulent kinetic energy, *epsilonWall-Function* for the dissipation and *nutkWallFunction* for the eddy viscosity. By applying these wall functions the velocity at the wall was implicitly modelled according to the approximations explained in section 2.1.3.

3.4 Interfacial Forces

The same interfacial forces were considered as in the model by Kurul *et al.* [21]. The total force consisted of the drag force, the virtual mass force and the force due to phase change. The term F_V in Equation 3.4 was then given by

$$F_V = F_V^{drag} + F_V^{vm} + F_V^{\Gamma} \tag{3.24}$$

and F_L in Equation 3.6 given by

$$F_L = -F_V$$

The suggested model ignored other momentum forces like the wall lubrication force and turbulent dispersion force. The force due to phase change $\Gamma_{ki}\boldsymbol{u}_i - \Gamma_{ik}\boldsymbol{u}_k$ is already given explicitly in Equation 3.4 and Equation 3.6. It depends on the mass transfer terms, which will be described later.

3.4.1 Drag Force

Kurul *et al.* [21] defined the drag force in Equation 3.24 according to

$$F_V^{drag} = \frac{1}{2} a_i \rho_L C_D |\mathbf{u}_L - \mathbf{u}_V|\big(\mathbf{u}_L - \mathbf{u}_V\big) \tag{3.25}$$

and

$$F_V^{drag} = -F_L^{drag} \tag{3.26}$$

where a_i denote the interfacial area concentration and C_D is the drag coefficient. The formulation used in this model differs a bit from Kurul *et al.* and is based in the Ishii-Zuber formulation [2] which reads

$$F_V^{drag} = \frac{3}{4}\rho_L \alpha_L \frac{C_D}{d_B}|\mathbf{u}_V - \mathbf{u}_L|\big(\mathbf{u}_V - \mathbf{u}_L\big) \tag{3.27}$$

Where d_B is the bubble diameter. The drag coefficient C_D is calculated with the Reynold number and states [16]

$$C_D = \begin{cases} \frac{24}{\mathrm{Re}_{d_B}}\big(1 + 0.15\mathrm{Re}_{d_B}^{0.687}\big)\ , \ \mathrm{Re}_{d_B} < 1000 \\ 0.44\mathrm{Re}_{d_B}\ , \ \mathrm{Re}_{d_B} > 1000 \end{cases} \tag{3.28}$$

where Re_{d_B} is the Reynolds number based on the bubble diameter and the vapour-liquid relative velocity calculated by

$$\mathrm{Re}_{d_B} = \frac{d_B|\mathbf{u}_V - \mathbf{u}_L|}{\nu_L}. \tag{3.29}$$

3.4.2 Virtual Mass Force

The virtual mass force from Equation 3.24 can be expressed as

$$F_V^{vm} = \alpha_V \rho_L C_{vm}\Big[\Big(\frac{\partial \mathbf{u}_V}{\partial t} + \mathbf{u}_V \cdot \nabla \mathbf{u}_V\Big) - \Big(\frac{\partial \mathbf{u}_L}{\partial t} + \mathbf{u}_L \cdot \nabla \mathbf{u}_L\Big)\Big] \tag{3.30}$$

according to [2, 21, 26] and

$$F_V^{vm} = -F_L^{vm} \tag{3.31}$$

where C_{vm} is the virtual mass coefficient and was set to 0.5.

3.4.3 Force due to phase change

The force due to phase change in Equation 3.24 was calculated based on established models [27, 10, 11, 20]. It was given by

$$F_V^{\Gamma} = \Gamma_{evap}\boldsymbol{u}_L - \Gamma_{cond}\boldsymbol{u}_V \tag{3.32}$$

3.5 Onset of Nucleate Boiling

The local condition to decide if the flow still was in the single-phase convective heating regime or had reached a two-phase boiling regime was modelled with a new approach suggested by professor Anglart. The idea was to formulate an explicit limit for the local liquid wall temperature, $T_{wall,L}$, above which the flow was treated as a two-phase subcooled nucleate boiling flow. This approach was adopted in an effort to reduce the need of computational power.

The idea was to model the limit of the wall super heat, $\Delta T_{sup} = T_{wall,L} - T_{sat}$, at which subcooled nucleate boiling starts. Since the saturation temperature, T_{sat}, was known from tabulated data [13] the modelled limit of ΔT_{sup} gave the corresponding limiting value of the wall temperature, denoted $T_{wall,L}^{limit}$. Then the calculated local $T_{wall,L}$ was compared with $T_{wall,L}^{limit}$ to determine if phase-change (boiling) was to be calculated at the wall.

To close the system, ΔT_{sup} and $T_{wall,L}$ needed to be modelled. Many models for the wall super heat have been presented and in this model the Jens-Lottes correlation [12] was adopted, see below. The wall temperature was calculated from the temperature in the cell adjacent to wall by a law-of-the-wall for temperature, see below.

3.5.1 Jens-Lottes Correlation

The wall super heat ΔT_{sup} was modelled by the Jens-Lottes correlation. The correlation models ΔT_{sup}, at which subcooled nucleate boiling starts, according to

$$\Delta T_{sup}^{J-L} = 25e^{-P/62}\Big(\frac{q''_{wall}}{10^6}\Big)^{0.25} \tag{3.33}$$

From Equation 3.33 the wall temperature the limiting value of the wall temperature was obtained according to

$$T_{wall,L}^{limit} = T_{sat} + \Delta T_{sup}^{J-L} \tag{3.34}$$

The local wall temperature, $T_{wall,L}$, from the solution was compared to $T_{wall,L}^{limit}$ and if $T_{wall,L} \geq T_{wall,L}^{limit}$, ONS was considered to be reached and boiling properties were calculated. Since Jens-Lottes is a global correlation and does not give any detailed information about the behaviour of the flow the condition was checked in each cell at each time step to determine if boiling occurred.

3.5.2 Wall Temperature

The law-of-the-wall from the CFD software Fluent was adopted in this model [8]. It is based on the theories proposed by Jayatilleke [4], Lauder *et al.* [17] and Rubesin *et al.* [30]. According to the theory, the law-of-the-wall is defined with respect to a point P next to wall. The dimensionless distance to the wall is a scaling of y and defined as

$$y^* = \frac{\rho_L 0.09^{1/4} k_P^{1/2} y_P}{\mu_L} \tag{3.35}$$

were y_P is the distance from point P to the wall and k_P is the turbulent kinetic energy at the point P. Further more a temperature quantity T^* is defined as

$$T^* \equiv \frac{\big(T_{wall} - T_P\big)\rho_L C_{pL} 0.09^{1/4} k_P^{1/2}}{q''_{wall}} \tag{3.36}$$

With Equation 3.35 the law-of-the-wall models T^* as

$$T^* = \begin{cases} \mathrm{Pr} y^* \,,\; y^* < y_T^* \\ \mathrm{Pr}_t\Big(\frac{1}{\kappa}\ln(9.793 y^*) + P\Big) \,,\; y^* > y_T^* \end{cases} \tag{3.37}$$

where Pr_t is the turbulent Prandtl number equal to 0.85, κ is the con Kármán constant equal to 0.4187 and P is defined as

$$P = 9.24\Big(\Big(\frac{\mathrm{Pr}}{\mathrm{Pr}_t}\Big)^{3/4} - 1\Big)\Big(1 + 0.28e^{-0.007\mathrm{Pr}/\mathrm{Pr}_t}\Big) \tag{3.38}$$

The variable y_T^* is defined as the y^* where the linear law and the logarithmic law intersect. With the modelled T^* the wall temperature can be calculated from Equation 3.36.

To test that the modelling of the wall temperature was modelled accurately a rough estimate of the wall temperature was calculated with the Dittus-Boelter equation [28]. The Dittus-Boelter equation calculates the Nusselt number according to

$$\mathrm{Nu} = 0.023\mathrm{Re}^{4/5} Pr^{2/5} \tag{3.39}$$

With the definition of the Nusselt number and Newton's law of cooling the Dittus-Boelter wall temperature was calculated according to

$$T_{wall}^{D-B} = T_{bulk} + \frac{q''_{wall} D}{\mathrm{Nu}\lambda_L} \tag{3.40}$$

3.6 Wall Heat Flux Partitioning

Kurul *et al.* suggested a partitioning of the total wall heat flux into different parts when boiling occurs [21]; a single-phase part, a quenching part and a evaporation part. This model of the heat flux partitioning was simplified to save computational resources. The heat flux was divided into two parts, the evaporation heat flux q''_{evap} and the single-phase heat flux q''_{single}. The heat not going to evaporation of the liquid was assumed to heat the liquid phase. Since the total heat flux were given by q''_{wall}, the single-phase heat flux could be expressed

$$q''_{single} = q''_{wall} - q''_{evap} \tag{3.41}$$

The heat flux q''_{evap} is one of the key components of boiling models and is used to define the mass transfer from liquid to vapour.

3.7 Evaporation Wall Heat Flux

The evaporation heat flux q''_{evap} was modelled with the same expression as suggested by [21] and read

$$q''_{evap} = \frac{\pi}{6} d^3_{Dep} \rho_V L N f \tag{3.42}$$

where d_{Dep} is the departure diameter of the vapour bubbles, N is the nucleation site density, f is the bubble frequency and L is the latent heat of evaporation. The terms N, f and d_{Dep} needed appropriate modelling to get a closed system.

3.7.1 Nucleation Site Density

The nucleation site density N was also modelled according to Kurul *et al.* [21] and read

$$N = \big(210(T_{wall,L} - T_{sat})\big)^{1.805} \tag{3.43}$$

The nucleation site density depends on the wall temperature and the saturation temperature, i.e, the wall super heat. The input temperatures has to be given in Kelvin or Celsius in order to use Equation 3.43 and the result is given in m^{-2}.

3.7.2 Bubble Detachment Frequency

The bubble detachment frequency is given as a relation between the gravity g, the bubble departure diameter and the density of both phases [6]. It reads

$$f = \sqrt{\frac{4}{3} \frac{g\big(\rho_L - \rho_V\big)}{\rho_L d_{Dep}}} \tag{3.44}$$

Since both the bubble detachment frequency and the evaporation heat flux in the model depended on the bubble detachment diameter a robust model for this quantity was of significant importance.

3.7.3 Bubble Departure Diameter

The calculation of the diameter of the departing bubbles was based on the theory presented by Ünal [32]. He proposed a way to analytically derive the average bubble detachment diameter and growth time. The theory was based on a number of assumptions [32]:

1. Subcooled nucleate boiling is a transition from the forced convection to the fully developed boiling regimes.

2. For constant operating conditions and geometry the bubbles emerging in the subcooled nucleate boiling flow regime have diameters and growth times that stretches over a range of values. A statistical approach is used and the result to get average values over the whole population for both the bubble detachment diameter and growth time.

3. The spherical or ellipsoidal bubbles growing on the surface grows on a thin liquid film with a partially dried section.

4. The instantaneous bubble volume is the same as a sphere with the same instantaneous bubble diameter d_{inst}.

5. The partially dried area, of the thin liquid film, is circle shaped.

6. The thin liquid film covers an area equal to $\pi d_{inst}^2(1 - d_{inst}/d_{dry})/4$ where d_{dry} is the diameter of the circle shaped dry area of the liquid film. The fraction d_{inst}/d_{dry} is assumed to be constant for a given pressure.

7. Inertia-controlled bubble growth has been neglected.

8. Contribution to the bubble growth from the super heated liquid layer can be neglected due to the large ratio between the maximum diameter and the thickness of the super heated layer. Consequently it is justified to assume that the heat input from the liquid film layer is large enough to neglect the contribution from the super heated layer.

9. Heat is assumed to dissipate to the liquid through condensation at the upper-half of the bubbles surface. The bottom half is assumed not to face the cold liquid hence not contributing to the dissipation of heat.

10. Bubble growth is assumed to be an isobaric process, i.e., it takes place at constant pressure.

11. Concerning bubbles with maximum diameter they can be divided into two groups. The first group is the bubbles formed at high subcooling. They reach their maximum diameter at the surface then slide along the heated surface without leaving it or collapses. The second group is formed at low- and medium subcooling and leave the heated surface when achieving maximum diameter. For both groups the diameter, at the point of detachment, can be described by $\frac{d(d_{inst})}{dt} = 0$.

He formulated a linear first order differential equation for the instantaneous bubble diameter, which read

$$\frac{d(d_{inst})}{dt} = D_a \omega t^{-1/2} - D_c \Phi D_b d_{inst} \tag{3.45}$$

where t is the bubble growth time. Solving Equation 3.45 lead to the expression

$$d_{inst}(t) = \frac{2D_a \omega t^{1/2}(1 + \frac{1}{3} D_b D_c \Phi t)}{1 + D_c \Phi D_b t} \tag{3.46}$$

Since $\frac{d(d_{inst})}{dt} = 0$ and $d_{inst}(t_dep) = d_{dep}$ at the time of departure t_{dep} he showed that the bubble departure diameter could be expressed as

$$d_{Dep} = \frac{2.42 \times 10^{-5} P^{0.709} D_a}{(D_b \Phi)^{1/2}} \tag{3.47}$$

where Φ depend on the bulk velocity according to

$$\Phi = \begin{cases} \left(\frac{\overline{\mathbf{u}}_L}{0.61}\right)^{0.47} & for \quad \overline{\mathbf{u}}_L < 0.61 m/s \\ 1 & otherwisw \end{cases} \tag{3.48}$$

The term D_b was defined

$$D_b = \frac{T_{sat} - T_{bulk}}{2(1 - \rho_V/\rho_L)} \tag{3.49}$$

and

$$D_a = \frac{(q''_{wall} - H_{single}(T_{sat} - T_{bulk}))^{1/3}\lambda_L}{2D_c^{1/3}L\rho_V\sqrt{\pi\lambda_L/\rho_L C_{pL}}}\sqrt{\frac{\lambda_{surf}C_{surf}\rho_{surf}}{\lambda_L\rho_L C_{pL}}} \tag{3.50}$$

where H_{single} is the single-phase forced convection heat transfer coefficient for heated surface. The terms λ_{surf}, C_{surf} and the ρ_{surf} is the thermal conductivity, specific heat and density of the heated surface, respectively. In Equation 3.50 λ_L is the thermal conductivity of the liquid and C_{pL} is the specific heat of the liquid. The coefficient D_c was given by

$$D_c = \frac{L\mu_L\left[\frac{C_{pL}}{0.013L\mathrm{Pr}^{1.7}}\right]^3}{\sqrt{\frac{\sigma}{(\rho_L-\rho_V)g}}} \tag{3.51}$$

were σ is the surface tension.

The single-phase heat transfer coefficient H_{single} was modelled in terms of the local Stanton number according to [21]

$$H_{single} = \mathrm{St}\rho_L C_{p,L}\overline{\mathbf{u}}_L \tag{3.52}$$

The Stanton number was calculated with the same method as Michta [20]. In that method the St is calculated in terms of the Fanning friction factor C_f according to

$$\mathrm{St} = \frac{C_f^2}{1 - 1.783C_f} \tag{3.53}$$

where the friction factor is defined implicitly through

$$C_f = \frac{1}{\frac{\ln(Re_L C_f)}{0.435} - 5.05} \tag{3.54}$$

and obtained by iteration of Equation 3.54 given a starting value of 0.062 for C_f. In Equation 3.54 Re_L is the global Reynolds number for the liquid phase calculated according to

$$Re_L = \frac{d_{pipe}||\boldsymbol{u}_L||}{\nu_L} \tag{3.55}$$

This model has a limited range of applicability and is not valid in the general case. To accurately predict d_{dep} the following conditions, of the physical system, must be fulfilled:

$0.1 < P < 17.7\ MPa$

$0.47 < q''_{wall} < 10.64\ MW/m^2$

$0.08 < \overline{\mathbf{u}}_L < 9.15\ m/s$

$3 < T_{sat} - T_{bulk} < 86\ K$

$0.08 < d_{Dep} < 1.24\ mm$

3.8 Interfacial Mass Transfer

3.8.1 Evaporation

The interfacial mass term due to evaporation was defined in the same manner as in [11]. Since the transition from liquid to vapour only took place at the heated wall, of the pipe, it is a mass flux boundary condition. However, to implement such a boundary condition was out of the scope of this thesis. Instead the evaporation was defined as a production term and was defined as zero in the whole domain, except at the boundary cells adjacent to the heated wall [11, 20].

Since the heat flux, going to the evaporation, was known (according to section 3.7) it could be defined as

$$\Gamma_{evap} = \frac{q''_{evap}}{L + C_{p,L}(T_{sat,L} - T_{bulk})} \frac{A_{wall,i}}{V_{wall,i}} \tag{3.56}$$

where $A_{wall,i}$ and $V_{wall,i}$ are the wall area and volume of the i'th wall cell. The fraction $\frac{A_{wall,i}}{V_{wall,i}}$ transforms the expression into a volumetric source term for the evaporation.

3.8.2 Condensation

Since the vapour was assumed to be at saturated conditions there were no temperature gradients within the vapour bubbles. Consequently, the heat transfer to the liquid phase could only take place in form of condensation at the interface between the phases [5, 13].

The mass flow rate per unit volume due to condensation for an arbitrary geometry can be described by [13]

$$\Gamma_{cond} = \frac{\dot{m}_{cond}}{V} \tag{3.57}$$

where $\dot{m}_{cond}$ is the mass flow rate due to condensation and V is the volume of the system. Moreover the mass flow rate can be defined as

$$\dot{m}_{cond} = \frac{q}{L} \tag{3.58}$$

where q is the heat transfer rate given in W. Using Newtons law of cooling expressed as

$$q = H_{if} A_{if} \big(T_{sat} - T_L\big) \tag{3.59}$$

where A_{if} is the total interfacial area between the two phases of the system it follows that

$$\dot{m}_{cond} = \frac{H_{if} A_{if} \big(T_{sat} - T_L\big)}{L} \tag{3.60}$$

By definition $a_i = A_{if}/V$, hence the interfacial mass transfer rate per unit volume due to condensation can be expressed as[11, 20]

$$\Gamma_{cond} = \frac{H_{if} a_i \big(T_{sat} - T_L\big)}{L} \tag{3.61}$$

where a_i was given by Equation 2.44.

The term H_{if} in Equation 3.61 is the interface heat transfer coefficient and was calculated with the bubble Nusselt number Nu_b according to

$$H_{if} = \frac{\mathrm{Nu}_b \lambda_L}{d_B} \tag{3.62}$$

where d_B is the bubble diameter in the bulk of the liquid. The bubble Nusselt number was obtained with the Ranz-Marchall correlation [25] as

$$\mathrm{Nu}_{d_B} = 2 + 0.6 \mathrm{Re}_{d_B}^{1/2} Pr_L^{1/3} \tag{3.63}$$

where the bubble Reynold number was calculated based on the relative velocity between the phases according to

$$\mathrm{Re}_{d_B} = \frac{\|\mathbf{u}_V - \mathbf{u}_L\| d_B}{\nu_L} \tag{3.64}$$

An alternative way to calculate H_{if}, that was used by Michta [20], reads

$$H_{if} = \rho_L C_{p,L} \sqrt{\frac{\pi}{4} \frac{\|\mathbf{u}_V - \mathbf{u}_L\|}{d_B} \frac{\lambda_L}{\rho_L C_{p,L}} \frac{1}{1 + \lambda_L^t / \lambda_L}} \tag{3.65}$$

3.9 Bubble Diameter in Bulk

The condensation term depended on an accurate description of the bubble diameter in the liquid bulk. This could be done in several ways. Kurul *et al.* [21] developed the following correlation for the bulk bubble diameter

$$d_b = 10^{-4}\big(T_L - T_{sat}\big) + 0.0014 \tag{3.66}$$

Another way to calculate the diameter of the bubbles is implicitly through the interfacial area concentration Equation 2.44. Since the void fraction was solved with the continuity equation the bulk diameter could be calculated in the bulk cells if a_i was known.

The interfacial area concentration can be modelled by solving a separated differential equation. Ishii *et al.* [19] suggested the transport equation

$$\frac{\partial a_i}{\partial t} + \nabla \cdot (a_i \mathbf{u}_V) = \frac{2}{3} \frac{a_i}{\alpha_V} \Big(\frac{\partial \alpha_V}{\partial t} + \nabla \cdot (\alpha_V \mathbf{u}_V) \Big) + \frac{1}{3\psi} \Big(\frac{\alpha_V}{a_i} \Big)^2 \big(R_{TI} + R_{RC} + R_{WE} \big) + R_{NUC} \tag{3.67}$$

for a_i where ψ accounts for the shape of the particles of interest. The terms R_{TI}, R_{RC} and R_{WE} are sink and source terms accounting for break up due to turbulence, coalescence due to turbulence and coalescence due to acceleration of a bubble in the wake of a preceding bubble, respectively. Due to the assumptions in the model, these three terms were neglected. The last source term R_{NUC} was not present in [19], it accounts for the contribution for a_i from the bubbles developing at the wall. It was in [27] and read

$$R_{NUC} = \pi d_{Dep}^2 N f \frac{A_{wall,i}}{V_{wall,i}} \tag{3.68}$$

The definitions of R_{TI}, R_{RC} and R_{WE} were taken as they were defined in [19].

3.10 Summary of Model

In Figure 3.2 an the boiling system depicted in Figure 3.1 is evolved to account for different parts of the presented boiling model.

In the magnified picture of the detaching bubble, the modelled heat flux partition in Equation 3.41 is depicted with the arrows (a), (b) and (c). The total wall heat flux is represented by (a) and the single phase heat flux is illustrated with (c). The arrow (b) represents the evaporation heat flux going into the vapor phase in the form of evaporation, modelled by Equation 3.56. The bubble detachment diameter, modelled with Equation 3.47, is depicted with the dashed arrow (d).

The magnified image of the bulk bubble depicts the condensation rate, modelled according to Equation 3.61, with (e). The interfacial area concentration, Equation 2.44, is represented by (f) and the bulk bubble diameter d_B is illustrated with (g).

The heated wall, heating the fluid, is represented by (h). The point (i) illustrates the bulk of the flow, where the properties α_L, α_V, h_L and T_L are solved by the governing equations Equation 3.1, Equation 3.2, Equation 3.13 and Equation 2.30, respectively. Properties defined at the surface of the wall is represented by (j). The properties T_{wall}, ΔT_{sup}, N, f and q''_{evap} are calculated according Equation 3.36, Equation 3.33, Equation 3.43, Equation 3.44 and Equation 3.42, respectively, at the wall surface. At the point (k), the ONB is illustrated.

In the magnified picture in the lower left corner, the interfacial forces are illustrated by the arrow around the bubble (l). In this region the flow is influenced by the interfacial forces F_V^{drag}, F_V^{vm} and F_V^{Γ} calculated according to Equation 3.25, Equation 3.30 and Equation 3.32, respectively. The vapor velocity $\boldsymbol{u}_V$, calculated by Equation 3.4, is illustrated at (m).

The region surrounding (n) represents the velocity profile of $\boldsymbol{u}_L$, given by Equation 3.6. The turbulent properties k_L, calculated according to Equation 3.22, and ϵ_L, calculated according to Equation 3.23 , are illustrated at (o). The references to properties on Figure 3.2 are listed in Table 3.1

(a) : q''_{wall}	(i) : α_L, α_V, h_L and T_L
(b) : q''_{evap}	(j) : T_{wall}, ΔT_{sup}, N, f and q''_{evap}
(c) : q''_{single}	(k) : ONB
(d) : d_{Dep}	(l) : F_V^{drag}, F_V^{vm} and F_V^{Γ}
(e) : Γ_{cond}	(m) : $\boldsymbol{u}_V$
(f) : a_i	(n) : $\boldsymbol{u}_L$
(g) : d_B	(o) : k_L and ϵ_L
(h) : wall	

Table 3.1: Node references for Figure 3.2.

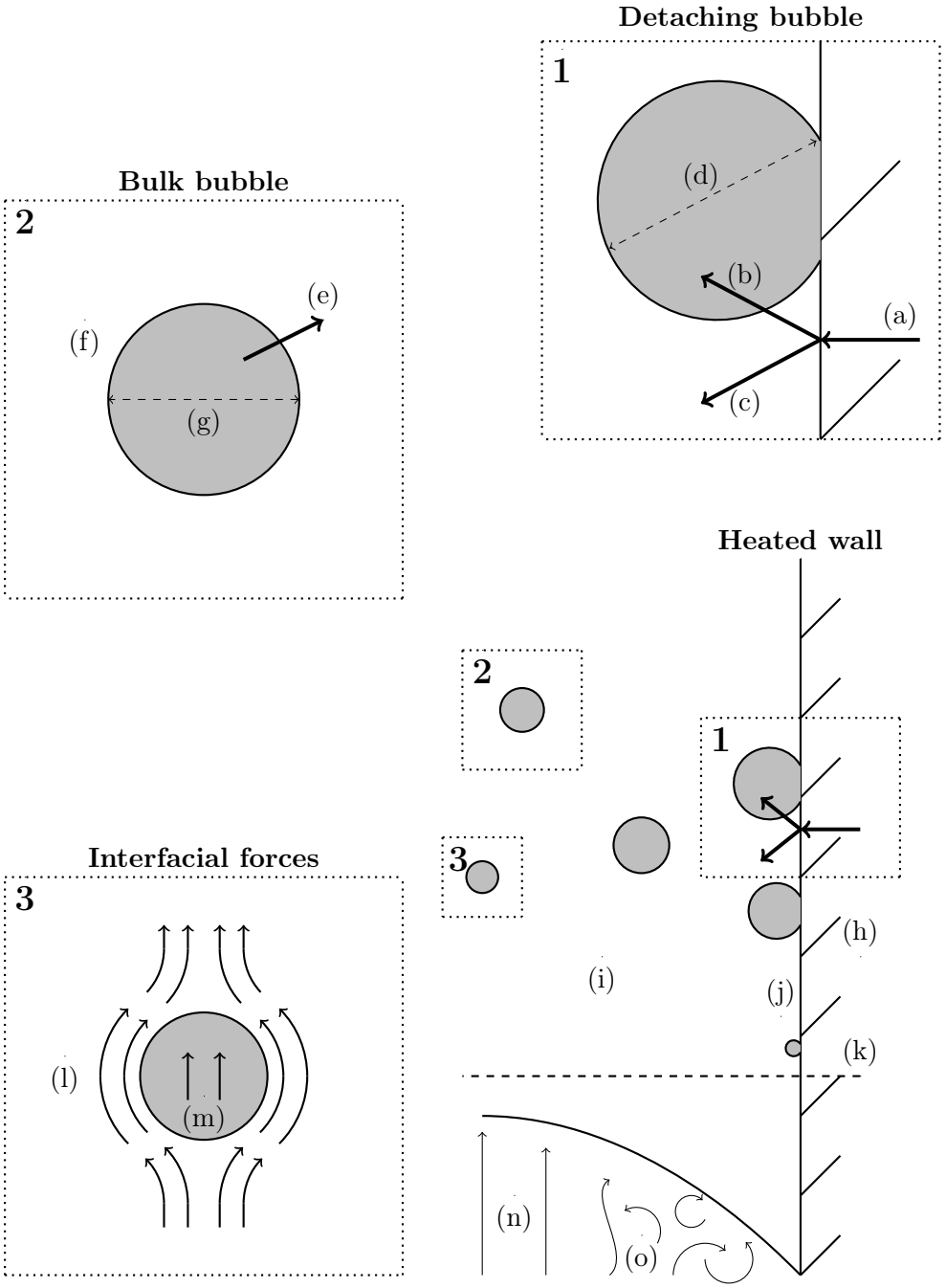

Figure 3.2: Graphical illustration of a vertical pipe with a wall, heated to the point that boiling starts. The major processes present in a boiling system, like evaporation, condensation, bubbles and interfacial forces, are illustrated with a magnified image to show more details.

In summary the model consist of 7 independent transport equations

- 2 equation for the mass conservation of both phases, Equation 3.1 and Equation 3.2
- 2 equations for the momentum conservation of both phases, Equation 3.6 and Equation 3.4
- 1 equation for the enthalpy of the liquid phase, Equation 3.13
- 2 equations for the turbulent quantities, Equation 3.22 and Equation 3.23

With the closure laws and correlations the complete model depends on 7 variables

- the void fraction α_V
- the velocities of the vapour and the liquid $\boldsymbol{u}_V$ and $\boldsymbol{u}_L$
- the liquid enthalpy h_L
- the turbulent kinetic energy k
- the turbulent dissipation ϵ
- the pressure P

This defines a closed system of equations suitable for modelling subcooled nucleate boiling.

4 Methods and Materials

This section covers the implementation of the model into a solver in the CFD software OpenFOAM. First, a general description of the structure of OpenFOAM is given. Basic concepts of developing solvers and setting up cases are presented. Then an explanation of how the different parts of the solver were implemented is given with derivations of more stable formulations of some equations that were better suited for numerical reasons.

4.1 OpenFOAM

OpenFOAM is a CFD software consisting of a large library of different functionalities, implemented in the C++ programming language. OpenFOAM is preferably used in a Linux environment. It is an open-source software and hence free of charge and the user is able to adjust the software in any way. This was an important reason for choosing OpenFOAM since most other software available on the market lack the possibility to adjust the solvers to the extent needed in this thesis [23].

4.1.1 General about OpenFOAM

The OpenFOAM software contains tools for both pre-processing along with the solver tools. Useful features in the pre-processing tools are the meshing tool and the utilities to create a wedge shaped mesh, see Appendix A for more details on the procedure to create such a mesh.

In the solving procedure in OpenFOAM there are two main parts. The first is the *applications* which are executables referred to as *solvers* and *utilities*. The solvers are algorithms for solving systems of differential equations mainly with the finite volume method. The utilities are functions for data manipulation [22, 23].

To solve a problem in OpenFOAM, the problem is defined in an entity called *case*. In a case, the details of the problem are given to OpenFOAM together with information about which solver to be used and, hence, which differential equations to be solved [22, 23]. The cases and solvers will be described in more detail in the following sections.

OpenFOAM does not include any post-processing software so that functionality has to be provided by another software. A common tool to use with OpenFOAM is the software ParaView [23]. In this thesis ParaView and Matlab have been the main post-processing tools.

4.1.2 Solvers

The core of a solver is defined in a .C-extension file in which the solution algorithm is implemented. It is often supported by header files, so called .H-extensions to obtain good readability of the code. The solvers can also call other .C-extensions for functions defined in separate classes. Each solver needs a *Make* folder containing the two files *options* and *files* that incorporates paths to libraries and files that are necessary for the solver to work. An example of a file structure of a solver named *newApp* can be seen in Figure 4.1 [23]

The solvers are implemented so that they solve a specific set of differential equations. For example, the *laplacianFoam*-solver solves the heat equation

$$\frac{\partial T}{\partial t} = \chi \Delta T \tag{4.1}$$

The syntax used to implement differential equation makes it very readable for the user. For example Equation 4.1 is implemented with

```
solve
(
    fvm::ddt(T) - fvm::laplacian(chi, T)
);
```

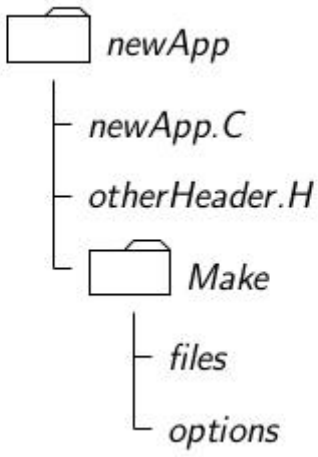

Figure 4.1: An example of a directory structure in a OpenFOAM solver [23].

Compilation of the solver files are executed conveniently with the command *./Allwmake* which compiles all the code needed to get the executable file to run cases with [22, 23].

4.1.3 Cases

When solving a problem, the solvers provide the general parts of the problem like the system of equation. The cases provide the details that defines the specific problem to be solved. A case is set up as a folder with an appropriate name containing the three sub folders *constant*, *system* and *0* [23].

The *constant*-folder contains the mesh and several other problem specific files defining constant properties of the case [23]. In this folder the mesh is stored in a directory named *polyMesh*. The other files, that depend on the specific problem, can be *turbulenceProperties* or *transportProperties*. For example, in this thesis the turbulence model for the phases were defined in files called *turbulenceProperties.liquid* and *turbulenceProperties.vapor*.

The *system* directory contains files relevant to the solver and usually includes *controlDict*, *fvSchemes* and *fvSolution*. The file *controlDict* includes controls for the solution like the solver to be used, starting time, ending time, time step and Courant number. The schemes used to discretize the derivatives in the differential equations are given in the file *fvSchemes*. The file *fvSolution* contains instructions for the solution algorithm like pressure correction procedures and relaxation [23].

In the directory *0* initial conditions and boundary conditions for all the fields in the problem are defined. The commonly used boundary condition types are predefined in OpenFOAM and can easily be set with keywords. For example a homogeneous von Neaumann condition has the keyword *zeroGradient* [22, 23].

4.2 Implementation

The implementation of the model was based on the existing solver *twoPhaseEulerFoam* which solves a two-phase flow according to the Euler-Euler approach. To this solver the physical quantities representing the subcooled nucleate boiling were added. An overview of how the new solver called *myTwoPhaseEulerFoamBoiling* was structured can be seen in Figure 4.2.

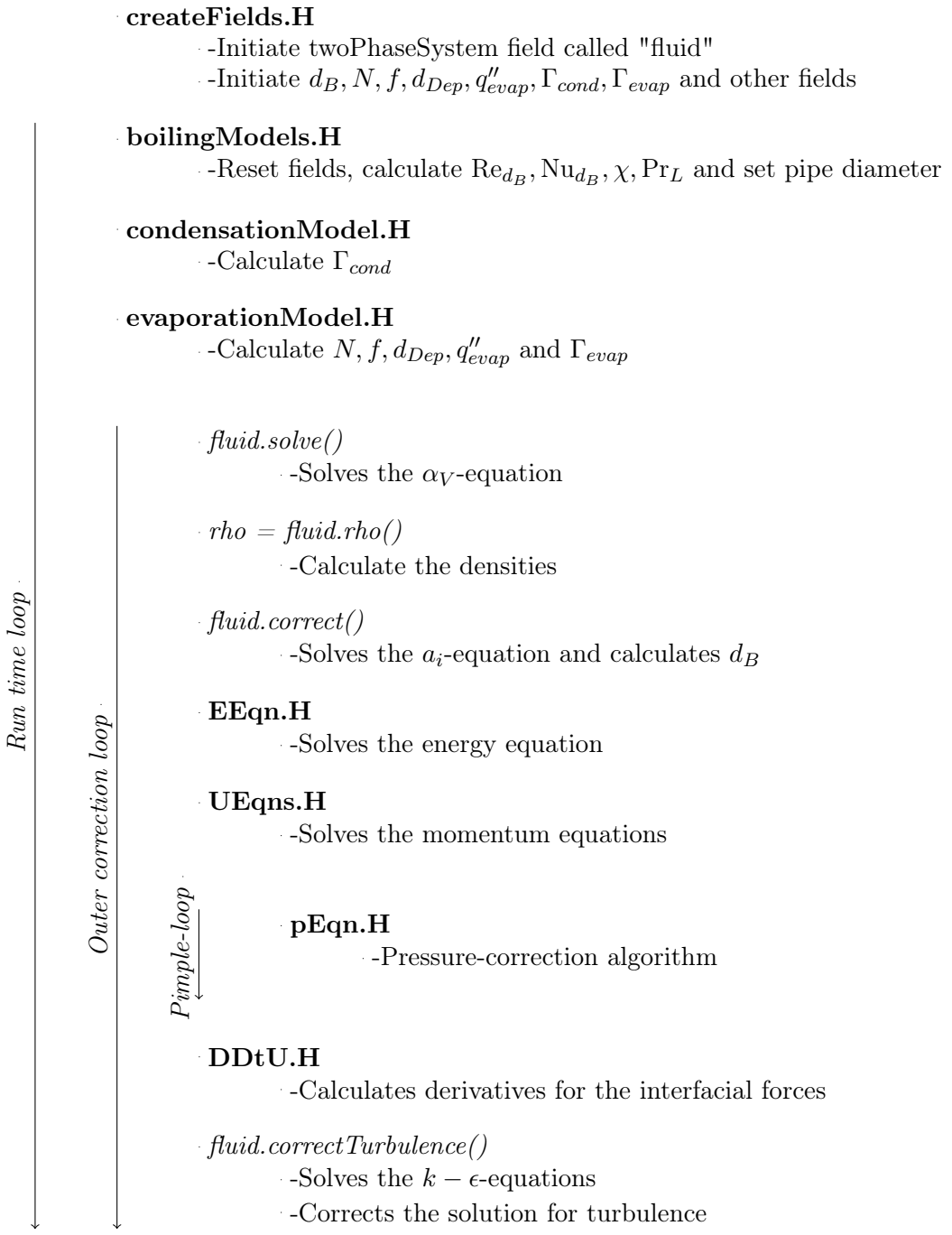

Figure 4.2: Graphic overview the algorithm implemented in the solver. The order in which the key parts are solved and the different loops used in the solver are presented.

4.2.1 Continuity Equation

The implemented continuity equation was a reformulated version of Equation 3.2. The reformulation was necessary to obtain a stable expression [10]. Due to the incompressibility of both phases, the continuity equation for phase k can be written

$$\frac{\partial \alpha_k}{\partial t} + \nabla \cdot (\alpha_k \mathbf{u}_k) = \frac{\Gamma_k}{\rho_k} \tag{4.2}$$

Introducing the relative velocity $\mathbf{u}_r = \mathbf{u}_V - \mathbf{u}_L$ and the combined velocity $\mathbf{u}_c = \alpha_V \mathbf{u}_V + \alpha_L \mathbf{u}_L$, the velocities of each phase can be expressed $\mathbf{u}_L = \mathbf{u}_c - \alpha_V \mathbf{u}_r$ and $\mathbf{u}_V = \mathbf{u}_c + \alpha_L \mathbf{u}_r$. Inserting these expressions in Equation 4.2 gives

$$\frac{\partial \alpha_V}{\partial t} + \nabla \cdot (\alpha_V \mathbf{u}_c) + \nabla \cdot (\alpha_V \alpha_L \mathbf{u}_r) = \frac{\Gamma_V}{\rho_V} \tag{4.3}$$

$$\frac{\partial \alpha_L}{\partial t} + \nabla \cdot (\alpha_L \mathbf{u}_c) - \nabla \cdot (\alpha_V \alpha_L \mathbf{u}_r) = \frac{\Gamma_L}{\rho_L} = -\frac{\Gamma_V}{\rho_L} \tag{4.4}$$

To guarantee boundedness of α_V [10], Equation 4.3 and Equation 4.4 are summed. Since $\alpha_L + \alpha_V = 1$, the resulting expression reads

$$\nabla \cdot \mathbf{u}_c = \frac{\Gamma_V}{\rho_V} - \frac{\Gamma_V}{\rho_L} \tag{4.5}$$

Rewriting

$$\nabla \cdot (\alpha_k \mathbf{u}_c) = \alpha_k \nabla \cdot \mathbf{u}_c + \mathbf{u}_c \cdot \nabla \alpha_k \tag{4.6}$$

in Equation 4.3 and inserting Equation 4.5 gives

$$\frac{\partial \alpha_V}{\partial t} + \mathbf{u}_c \cdot \nabla \alpha_V + \nabla \cdot (\alpha_V \alpha_L \mathbf{u}_r) = \alpha_V \Big(\frac{\Gamma_V}{\rho_L} - \frac{\Gamma_V}{\rho_V} \Big) + \frac{\Gamma_V}{\rho_V} \tag{4.7}$$

Applying Equation 4.6 again gives the following representation of the continuity equation

$$\frac{\partial \alpha_V}{\partial t} + \nabla \cdot (\alpha_V \mathbf{u}_c) - \alpha_V (\nabla \cdot \mathbf{u}_c) + \nabla \cdot (\alpha_V \alpha_L \mathbf{u}_r) =$$

$$\alpha_V \Big(\frac{\Gamma_V}{\rho_L} - \frac{\Gamma_V}{\rho_V} \Big) + \frac{\Gamma_V}{\rho_V} \tag{4.8}$$

In accordance with the void fraction originally implemented in the *twoPhaseEulerFoam*-solver, the L.H.S of Equation 4.8 is rewritten a second time in the same way. The resulting equation then reads

$$\frac{\partial \alpha_V}{\partial t} + \nabla \cdot (\alpha_V \mathbf{u}_c) + \nabla \cdot (\alpha_V \alpha_L \mathbf{u}_r) =$$

$$2\alpha_V (\nabla \cdot \mathbf{u}_c) + 2\alpha_V \Big(\frac{\Gamma_V}{\rho_L} - \frac{\Gamma_V}{\rho_V} \Big) + \frac{\Gamma_V}{\rho_V} \tag{4.9}$$

which is the finial version that was implemented in the solver. The only terms to implement were the source terms due to the mass transfer $\Gamma_V = \Gamma_{evap} - \Gamma_{cond}$.

When solving for a state variable ψ, the source term S is usually divided into an explicit part and an implicit part

$$S = S_P \psi_P + S_u. \tag{4.10}$$

The implicit part only contains negative entries and is added to the coefficient matrix of ψ_P and the explicit part is added to the source vector. Since both Γ_{evap} and Γ_{cond} were positive, the source term could be rewritten according to

$$2\alpha_V \Big(\frac{\Gamma_V}{\rho_L} - \frac{\Gamma_V}{\rho_V} \Big) + \frac{\Gamma_V}{\rho_V} = 2\alpha_V \Big(\frac{\Gamma_{evap} - \Gamma_{cond}}{\rho_L} - \frac{\Gamma_{evap} - \Gamma_{cond}}{\rho_V} \Big) + \frac{\Gamma_{evap} - \Gamma_{cond}}{\rho_V}$$

$$= S_p \alpha_V + S_u \tag{4.11}$$

where

$$S_P = -2\frac{\Gamma_{cond}}{\rho_L} - 2\frac{\Gamma_{evap}}{\rho_V} - \frac{\Gamma_{cond}}{\rho_V \alpha_V}$$

and

$$S_u = 2\alpha_V \Big(\frac{\Gamma_{evap}}{\rho_L} + \frac{\Gamma_{cond}}{\rho_V} \Big) + \frac{\Gamma_{evap}}{\rho_V}$$

The terms S_P and S_u were implemented as the implicit and explicit source terms, respectively. The modifications made in the code to add the source terms can be found in Appendix D.

4.2.2 Momentum Equation

The momentum equations used in the solver differed from the formulations Equation 3.6 and Equation 3.4 in the model. A rewritten form of the L.H.S. of the equations were used in the original solver. These formulations were kept in the solver.

Expanding the derivatives of the L.H.S. of the equation for the phase k gives

$$\frac{\partial \alpha_k \boldsymbol{u}_k}{\partial t} + \nabla \cdot \big(\alpha_k \boldsymbol{u}_k \boldsymbol{u}_k\big) = \alpha_k \frac{\partial \boldsymbol{u}_k}{\partial t} + \boldsymbol{u}_k \frac{\partial \alpha_k}{\partial t} + \alpha_k \boldsymbol{u}_k \nabla \cdot \big(\boldsymbol{u}_k\big) + \boldsymbol{u}_k \nabla \cdot \big(\alpha_k \boldsymbol{u}_k\big) \Rightarrow$$

$$\frac{\partial \alpha_k \boldsymbol{u}_k}{\partial t} + \nabla \cdot \big(\alpha_k \boldsymbol{u}_k \boldsymbol{u}_k\big) - \Big(\frac{\partial \alpha_k}{\partial t} + \nabla \cdot \big(\alpha_k \boldsymbol{u}_k\big)\Big)\boldsymbol{u}_k = \alpha_k \frac{\partial \boldsymbol{u}_k}{\partial t} + \alpha_k \boldsymbol{u}_k \nabla \cdot \big(\boldsymbol{u}_k\big) \tag{4.12}$$

The L.H.S of Equation 4.12 states the L.H.S of the implemented momentum equation for both phases. The gravitation and pressure term were not implemented in the momentum equation, instead these contributions were accounted for in the pressure equation, as implemented in the original solver. The drag force was divided into two parts, one was implemented in the momentum equation and one in the pressure equation, as in the original solver. This set up was kept and the forces due to phase change were added to the momentum equations, see Appendix F for the code.

4.2.3 Energy Equation

The energy equation was changed in a similar manner as the momentum equation with a rewritten implementation of the L.H.S according to

$$\frac{\partial \alpha_L h_L}{\partial t} + \nabla \cdot \big(\alpha_L h_L h_L\big) - \Big(\frac{\partial \alpha_L}{\partial t} + \nabla \cdot \big(\alpha_L h_L\big)\Big) h_L \tag{4.13}$$

The remaining parts of the equation was implemented in a similar way as done by Michta [20] with the effective kinematic viscosity calculated by the original equations in the turbulence models provided in OpenFOAM, see Appendix E.

4.2.4 Pressure Correction

The pressure and velocity were solved in parallel with the iterative PIMPLE-algorithm in OpenFOAM. The PIMPLE algorithm is a mixture between the well known SIMPLE algorithm and the PISO algorithm [22, 23, 29]. The algorithm is based on a specific equation for the pressure which is consistent with the continuity to calculate the pressure field.

The algorithm starts by solving the momentum equations without the pressure term, giving initial guesses of the velocity fields which do not obey the continuity equation. Then an initial guess for the pressure field is defined and the PIMPLE-loop is entered. In the loop the pressure is updated by solving the pressure equation which ensures that continuity is obeyed. The momentum equations are then solved with the updated pressure field to obtain better approximations

of the velocity fields. The PIMPLE-loop is then restated with the obtained velocity and pressure fields. This process is iterated until convergence is achieved.

In the pressure equation, implemented in the original *twoPhaseEulerFoam*-solver, the contribution from the mass transfer terms in the continuity equation had to be added. According to [10] the continuity constraint was taken for the mixture since there are two phases present. The pressure equation can be derived from the Equation 4.5 by replacing the velocity with the combined *flux* ϕ_c according to

$$\nabla \cdot \boldsymbol{\phi}_c = -\frac{\alpha_V}{\rho_V}\frac{D(\rho_V)}{dt} - \frac{\alpha_L}{\rho_L}\frac{D(\rho_L)}{dt} + \frac{\Gamma_V}{\rho_V} - \frac{\Gamma_V}{\rho_L} \tag{4.14}$$

The flux is defined in OpenFOAM as $\phi = \boldsymbol{S} \cdot \boldsymbol{u}_f$, where $\boldsymbol{S}$ is the surface area vector and $\boldsymbol{u}_f$ is the velocity interpolated on the cell faces [22]. In the original solver code the corresponding equation without the mass terms was implemented as

$$\text{pEqnComp1()} + \text{pEqnComp2()} + \text{pEqnIncomp()}{=}0 \tag{4.15}$$

where pEqnIncomp() corresponds to $\nabla \cdot \boldsymbol{\phi}_c$, pEqnComp1() to $\frac{\alpha_V}{\rho_V}\frac{D(\rho_V)}{dt}$ and pEqnComp2() to $\frac{\alpha_L}{\rho_L}\frac{D(\rho_L)}{dt}$. Adding the mass transfer terms to the same row as Equation 4.15 were the only adjustment needed.

4.2.5 Boiling Terms

The boiling terms of the model were implemented in the solver straight forward. The constant values like material properties and relaxation factors were implemented to be read from the case files. The only concern were the evaporation terms that only should be calculated at the wall cells of the mesh. This called for access to specific cells of the fields in the solver which is not the intended way to use OpenFOAM. It was done in a similar manner as done by Michta [20]. The code can be seen in Appendix C. The bulk temperature and velocity was calculated by integration over the cross-section of the pipe. This was done since the methods of taking the bulk values from a certain distance from the wall could cause large jumps in values due to the course mesh that was used.

The implementation of the a_i-equation already existed in the original solver. The only difference was that it was implemented in terms of the *interfacial curvature* κ_i, defined as $\kappa_i = a_i/\alpha$, and hence $\kappa_i = 6/d_B$. The only adjustment needed was correcting an error in the implementation highlighted in a bug report [9] and adding the source term due bubble development at the wall Equation 3.68.

Evaporation The evaporation term was calculated with a for-loop that steps through each wall cell and calculates the boiling terms in each wall cell, leaving the remaining cell values as zero. At each cell the wall temperature was calculated according to the law-of-the-wall explained in section 3.5.2. If ONB was achieved the boiling parameters N, f, q_{evap}, d_{Dep} and Γ_{evap} were calculated according to the model. The evaporation term was relaxed to achieve a stable solution by an explicit relaxation term, given in the case input file *phaseProperties*. The code for the previous parameters can be found in the file *evaporationModel.H* in Appendix C.2 and in *departureDiameters.H* in Appendix C.3.

Condensation The condensation was implemented straight forward according to the theory in the model. To achieve a stable solution the condensation was relaxed, in the same way as the evaporation, to avoid divergence. The code for the condensation term was implemented in the file *condensationModel.H*, see Appendix C.1.

Input Parameters The new boiling parameters were read from the case files. The relaxation factors were given as input from the *fvSolution*-file in the *system*-directory. The other properties of the wall material and the fluid were given as input from the *phaseProperties*-file in the *constant*-directory. The different interpolation schemes for the derivatives and other terms were set according to *fvSchemes* in the *system*-directory. For more details about the interpolation schemes in OpenFOAM see [22]

4.3 Test Cases

4.3.1 Geometry

The geometry was similar to the one used by Kurul *et al.* [21], a vertical cylindrical pipe with diameter 0.0154 m and length 2.85 m. The length was not given and was set so that the flow achieved the desired conditions according to energy balance. To save computational time a wedge with symmetric center line, 0.0077 m radius and an angle of 5° was used instead of a full cylinder.

The mesh refinement was limited to having cell dimensions no larger than the maximum vapour bubble diameter. Since the the bubble diameter was defined as a field variable at each cell, it would not be physically consistent to have a bubble diameter in a cell larger than the cell itself. Hence a to refined mesh would lead to vapour volumes in cells larger than the cells actual volume. Due to this restriction, a uniform cell distribution was used.

Considering the conditions for Ünal's model the mesh was created with 8 cells in radial direction and 3033 in vertical direction. A close up of the mesh can be seen in Figure 4.3.

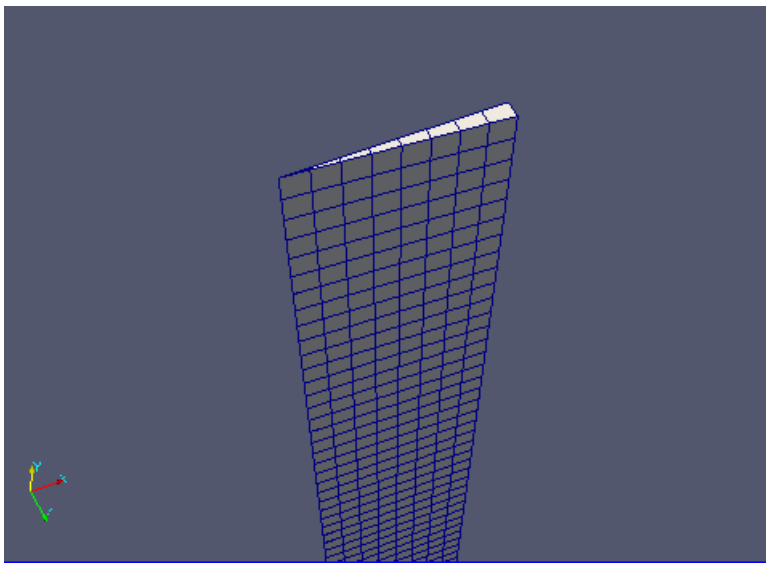

Figure 4.3: Close up of the mesh that was used in the simulations from the software ParaView.

4.3.2 Case

The case was the same as in Kurul *et al.* [21]. It was originally based on the experimental set up in [3]. The wall surface material was assumed to be stainless steal ANSI 316 which has the physical properties given in Table 4.1 The heat flux at the heated wall was set to 570 000 W/m^2.

The liquid flowing into the bottom of the pipe was water with a temperature of $440K$ at $45bar$. The pressure field given to the solver was set to 0 while the real pressure field used in the model was adjusted by a reference pressure of $45bar$. The inlet mass flux was $900kg/m^2/s$ which corresponded to an average inlet velocity of $0.9978m/s$ with the density taken at for water at the inlet temperature, see Table 4.3.

Property	Value
ρ_{surf}	$8030.0 kg/m^3$ %
$C_{p,surf}$	$500.0 J/kg/K$ %
λ_{surf}	$17.08 W/m/K$ %

Table 4.1: Material properties of the heated surface.

To avoid strange values of the wall heat, which was modelled with the turbulent kinetic energy, a single-phase non-heated case was simulated until fully developed conditions were achieved. The inlet values for the boiling case of U, k, ϵ and ν^t were taken from the outlet of the fully developed flow. The single-phase case had the boundary and initial conditions described in Table 4.2.

Field	Initial	Inlet	Outlet	Wall	Centerline
$U[m/s]$	(0 0.9978 0)	(0 0.9978 0)	zeroGradient	(0 0 0)	symmetryPlane
$p[kg/ms^2]$	0	zeroGradient	0	zeroGradient	symmetryPlane
$T[K]$	440	440	zeroGradient	zeroGradient	symmetryPlane
$k[m^2/s^2]$	0.002233961	0.002233961	calculated	wallFunction	symmetryPlane
$\epsilon[m^2/s^3]$	0.0160944670	0.0160944670	calculated	wallFunction	symmetryPlane
$\nu^t[m^2/s]$	$2.7907261 \cdot 10^{-5}$	$2.7907261 \cdot 10^{-5}$	calculated	wallFunction	symmetryPlane

Table 4.2: Initial and boundary conditions for the single-phase case used in the mesh sencitivity test an to obtain input values for the boiling case .

With the above mentioned values the boiling case was set up with appropriate boundary and initial conditions. The case set up can be seen in Table 4.3. The initial and inlet value for κ_i were set 6000 to give a initial and inlet bulk diameter equal to 0.001 m. Since α_V was 0 at the inlet and in the initial stage, the condensation and the interfacial forces would be zero despite the non zero bulk diameter. In addition, $\kappa_i = 75000 \Leftrightarrow d_B = 0.00008$ was tested to determine if this had any effect on the solution.

Field	Initial	Inlet	Outlet	Wall	Centerline
$U_L[m/s]$	(0 0.9978 0)	(0 0.9978 0)	zeroGradient	(0 0 0)	symmetryPlane
$U_V[m/s]$	(0 0 0)	(0 0 0)	pressure outlet	(0 0 0)	symmetryPlane
$p[kg/ms^2]$	0	zeroGradient	0	zeroGradient	symmetryPlane
$T_L[K]$	440	440	zeroGradient	zeroGradient	symmetryPlane
$\kappa_i[m^{-1}]$	6000	6000	zeroGradient	zeroGradient	symmetryPlane
α_V	0	zeroGradient	0	zeroGradient	symmetryPlane
$k_L[m^2/s^2]$	0.002233961	developed	calculated	wallFunction	symmetryPlane
$\epsilon_L[m^2/s^3]$	0.0160944670	developed	calculated	wallFunction	symmetryPlane
$\nu_L^t[m^2/s]$	$2.7907261 \cdot 10^{-5}$	developed	calculated	wallFunction	symmetryPlane

Table 4.3: Initial and boundary conditions for the boiling case, *developed* values refers to values taken from fully developed conditions .

Other constant physical properties of the flow were set according to Table 4.4. The properties were set to conditions for pressure of $45 bar$ and a saturation temperature of $530.5 K$[13], except ρ_L and $C_{p,L}$. The liquid density was taken for the inlet conditions and $C_{p,L}$ was calculated as an average value, consistent with the energy balance.

Property	Value
ρ_L	$902.0\ kg/m^3$
ρ_V	$22.69\ kg/m^3$
$C_{p,L}$	$4576.36\ J/kg/K$
$C_{p,V}$	$3975.5\ J/kg/K$
λ_L	$0.6125\ W/m/K$
λ_V	$0.05077\ W/m/K$
Pr_L	0.85
Pr_L^t	0.85
Pr_V	1.392
μ_L	$103.85 \cdot 10^{-6}\ Pa \cdot s$
μ_V	$17.739 \cdot 10^{-6}\ Pa \cdot s$
T_{sat}	$530.5K$
P_{ref}	$45.0bar$
σ	$0.02438\ N/m$
L	$1675570\ J/kg$

Table 4.4: Values of constant physical properties used in the model.

4.4 Mesh Sensitivity Test

To test the solvers sensitivity to the mesh refinement a single-phase case was set up and simulated on different refinements of the mesh. Since no phase-change was modelled the vertical dimension of the computational domain was shortened to 1.59 m.

Using energy balance, the outlet bulk temperature T_{out}^{bulk} was calculated analytically to 492.771 K from

$$\dot{m}c_p\left(T_{out}^{bulk} - T_{in}^{bulk}\right) = q''_{wall}L\pi D, \tag{4.16}$$

where $\dot{m}$ is the total inlet mass flux, L is the pipe length and D is the pipe diameter. The numerical outlet bulk temperature was calculated by integrating the output temperature over the output cross sectional area. Then the heat flux based on the numerical output temperature $q_{wall}^{num''}$ was calculated and compared to the analytical heat flux. The tested meshes had 6, 8, 12 and 18 radial cells, respectively, with the number of cells in the vertical direction set to achieve close to quadratic cells.

5 Results

In this section the results obtained from the developed solver are presented. The result consist of the solutions, the main case and a case with an alternative κ_i. Quantities are plotted with reference data from Kurul *et al.* [21] and with experimental data from Bartolemei *et al.*. Additional quantities, without reference for evaluation, are also presented for analysis. The presented data in this section is time-averaged over a period with the system in a stable state.

5.1 Temperatures and ONB

The evaporation heat flux is depicted in Figure 5.1. The resulting q''_{evap} from the simulation is plotted along q''_{evap} from Kurul and Podowski [21]. The total heat flux into the system is also plotted, with the dotted line, to illustrate the magnitude of the heat flux going into vapour production.

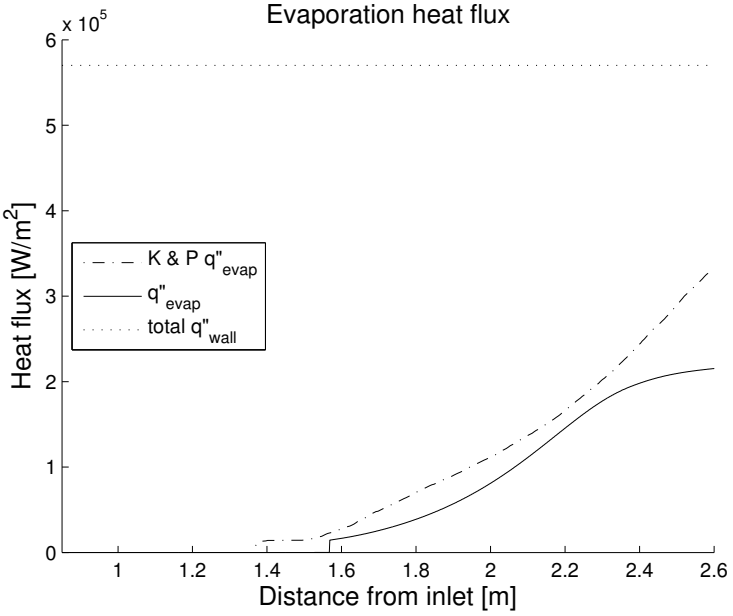

Figure 5.1: Plot of time-averaged q''_{evap} from the main simulation and the reference paper [21]. The total heat flux is given, as the dotted line, to illustrate the magnitude of the evaporation heat fluxes.

In Figure 5.2, the temperature along the center line and the bulk temperature are both depicted with the corresponding data from [21]. The first 0.85 m of the heated channel is not depicted.

Figure 5.3 depicts T_{wall} from the simulation and the corresponding data from [21] along the pipe. To illustrate the behaviour of the law-of-the-wall for the temperature, the temperature in the cells adjacent to the wall is plotted in the same plot. As in the previous plot, the first 0.85 m of the pipe is not included.

In Figure 5.4, the temperature profile (from simulation) at the outlet of the pipe is illustrated along the radial axis from the center line to the wall.

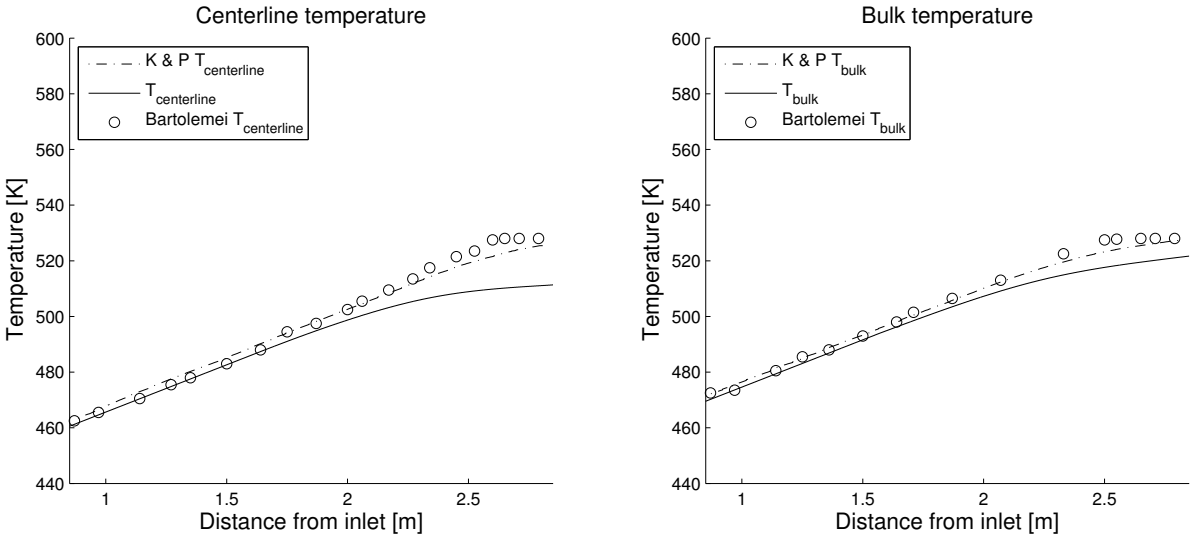

Figure 5.2: Left: Plot of the time-averaged temperature along the center line from the simulation and the from [21]. **Right:** Plot of the time-averaged bulk temperature along the vertical axis from the simulation and the from [21].

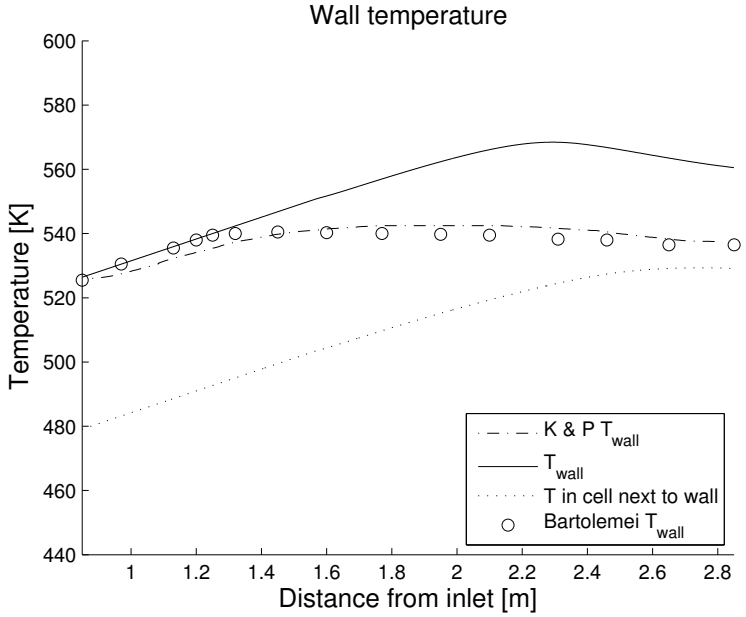

Figure 5.3: Plot of the wall temperature along the pipe from the solution and [21]. The temperature at the cells adjacent to the wall is also depicted to illustrate the behaviour of the law-of-the-wall for the temperature.

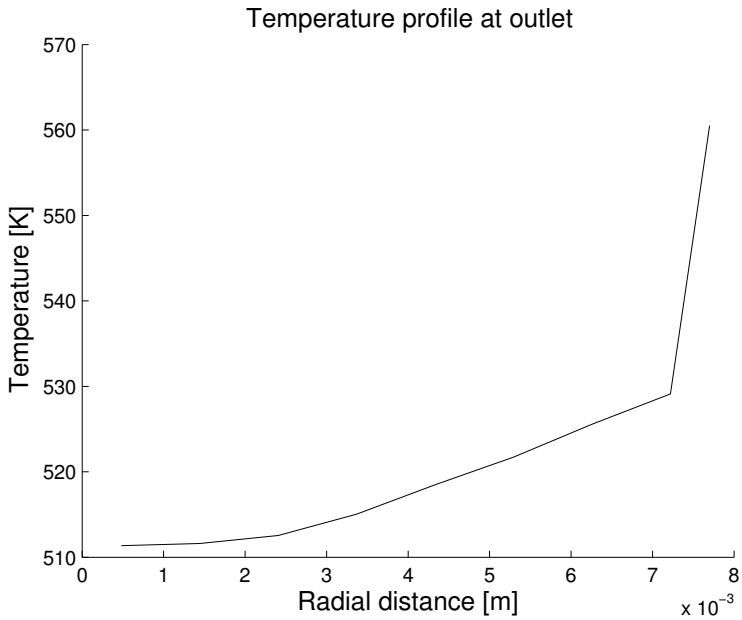

Figure 5.4: Temperature profile at the outlet from solution, illustrated along the radial axis from the center line to the wall.

5.2 Axial Diameters and Void Fraction

The left plot in Figure 5.5 depicts Ünal's detachment diameter and the bulk bubble diameter, in the cells adjacent to the wall, obtained from the κ_i-equation. The diameters are plotted along the wall of the vertical pipe with the initial section, where no boiling occurs, left out. The sudden increase and decrease of the diameters are caused by the initiation of boiling and, hence, calculated diameters arise instead of the initial guesses. The plot to the right illustrates the cross-sectional average void fraction of the vapour phase, along the vertical pipe, and the corresponding reference data [21].

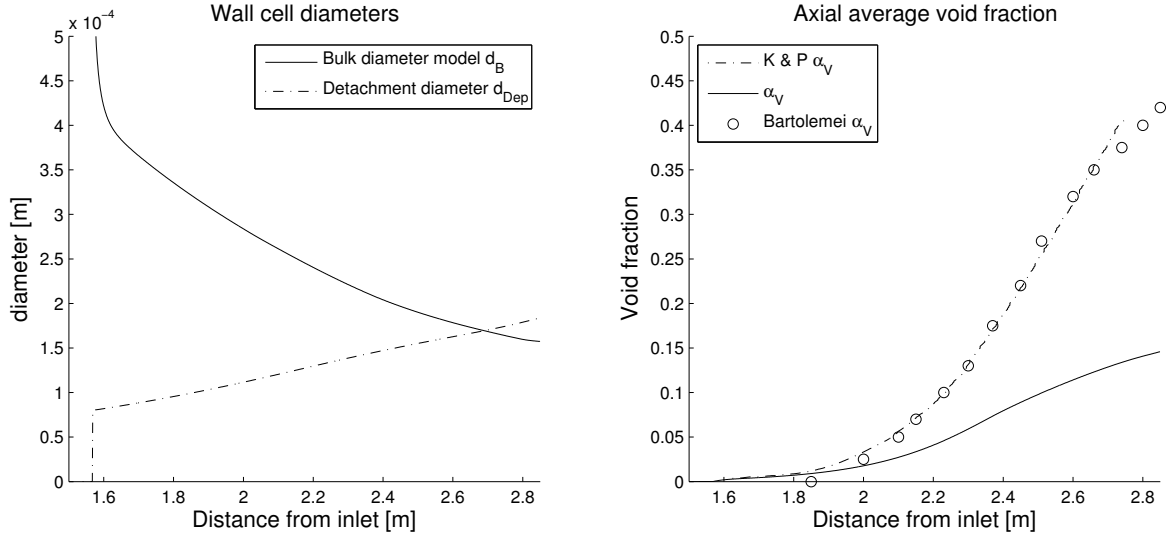

Figure 5.5: Left: The time-averaged detachment diameters, from Ünal's model, and the bulk bubble diameters ,from the wall cells, calculated through the κ_i-equation. **Right:** Time-averaged Cross-sectional average vapour void fractions along the vertical pipe from the results the corresponding data [21]

5.3 Cross-Sectional Profiles of Diameters and Void Fractions

The left plot in Figure 5.6 illustrates the radial bubble diameters at the outlet of the pipe and detachment diameter at the outlet. The right plot depicts the vapour void fraction profile at the outlet. Figure 5.7 shows the radial profile of a_i at the outlet.

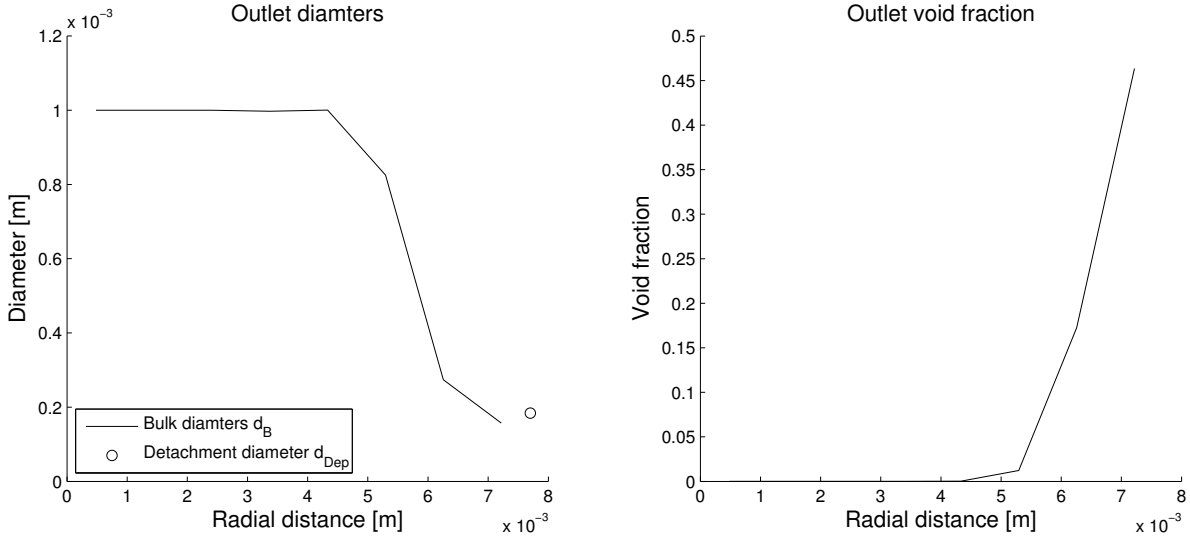

Figure 5.6: Left: Time-averaged radial profile of the cross-sectional bulk diameter at the outlet and the detachment diameter at the outlet, of the pipe. **Right:** Time-averaged radial profile of the vapour void fraction at the outlet of the pipe.

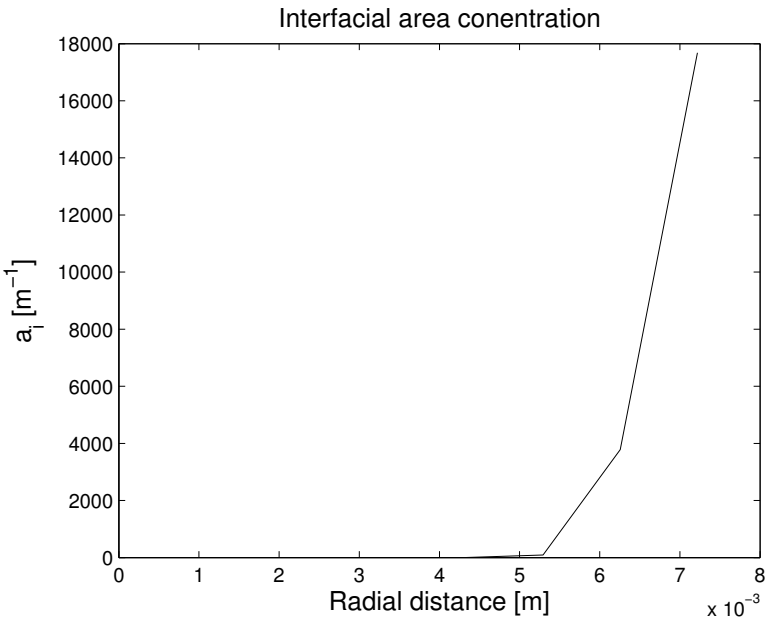

Figure 5.7: Time-averaged radial profile of a_i, from the center line to the wall, at the outlet of the pipe.

5.4 Velocities

The velocity profiles of both phases are depicted in Figure 5.8. The illustrated velocities, in the plots, are the superficial velocities, defined as $j_i = \alpha_i \boldsymbol{u}_i$. The left plot shows the superficial velocity for the vapour phase from the results and from the reference data [21]. The right plot depicts the corresponding data for the liquid phase. The profiles are taken at a distance of 2.45 m from the inlet.

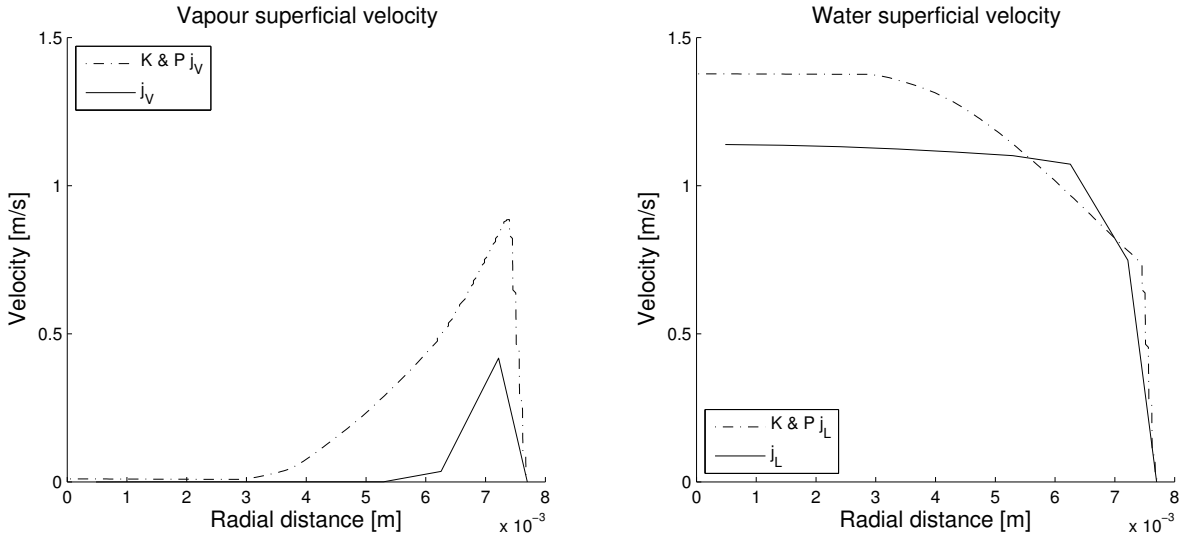

Figure 5.8: Left: Time-averaged superficial vapour velocity profiles, at 2.45 m from the inlet, from the result and the reference paper **Right:** Time-averaged superficial liquid velocity, at 2.45 m from the inlet, from the result and the reference paper

5.5 Results from Alternative Case Setup

The plots in this section are based on the results from the case with $\kappa_i = 75000$. In Figure 5.9, q''_{evap} is depicted with the corresponding values from [21]. The total heat flux going into the system is also illustrated with the dotted line.

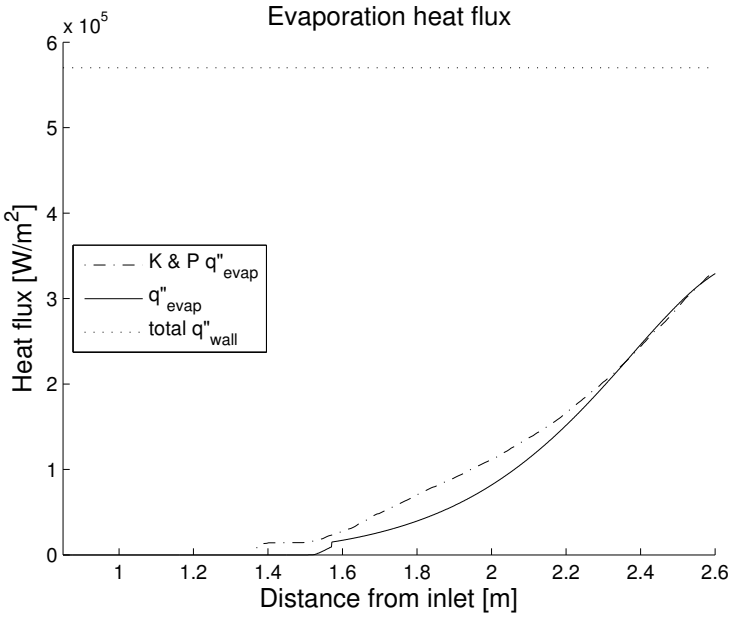

Figure 5.9: Plot of time-averaged q''_{evap} from the resulting simulation, with $\kappa_i = 75000$, and from [21]. The total heat flux that is going into the system is plotted with the dotted line to illustrate the magnitude of the evaporation heat fluxes.

Figure 5.10 depicts the same quantities as Figure 5.3; the temperature in the cells adjacent to the wall, T_{wall} from the simulation and T_{wall} from [21]. In Figure 5.11 the data corresponding to Figure 5.5 is illustrated for the alternative case. The plots in Figure 5.11 corresponds to Figure 5.6 and depicts the cross-sectional profiles of the diameters and α_V at the outlet.

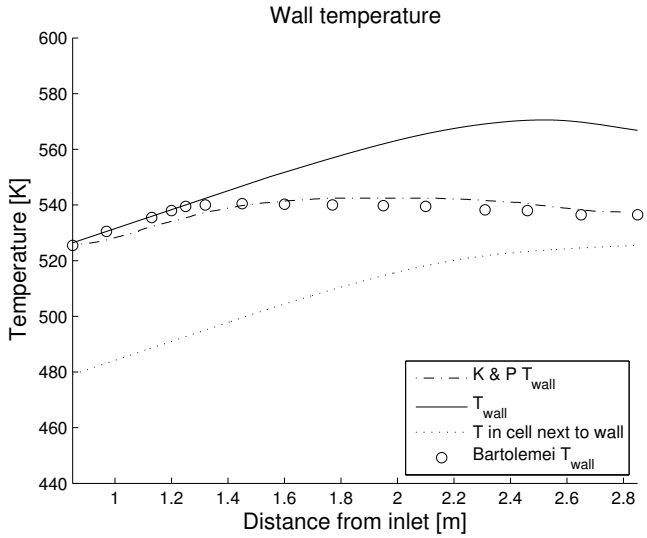

Figure 5.10: Plot of the wall temperature along the pipe from the solution, with $\kappa_i = 75000$, and the reference paper. The temperature at the cells next to the wall is also depicted to illustrate the behaviour of the law-of-the-wall for the temperature.

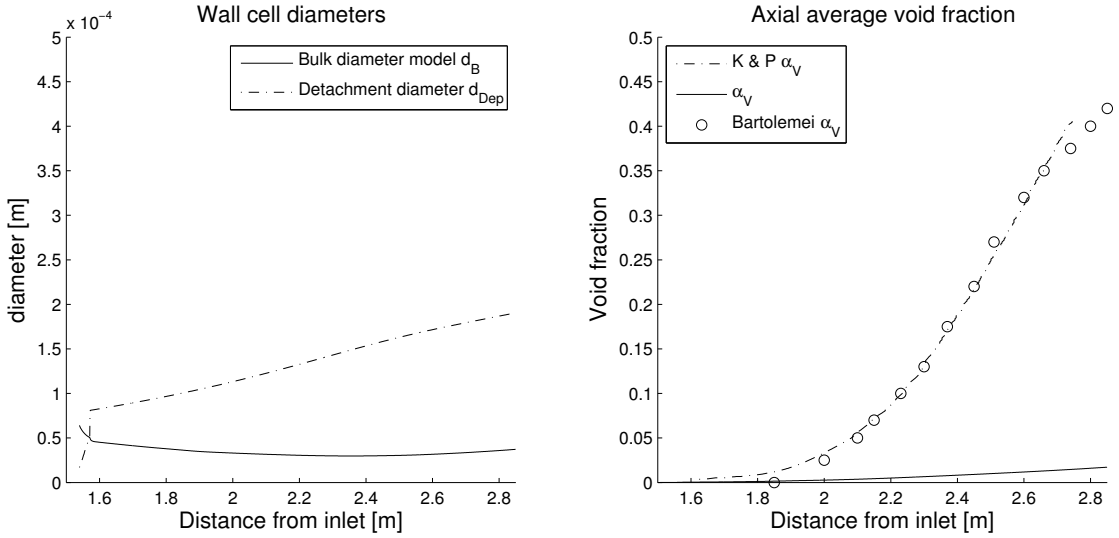

Figure 5.11: Left: Time-averaged detachment diameters, from Ünal's diameter model, and the bulk bubble diameters from the wall cells, calculated through the κ_i-equation with $\kappa_i = 75000$. **Right:** Time-average radial vapour void fraction profile along the vertical pipe from the results, with $\kappa_i = 75000$, and the paper [21].

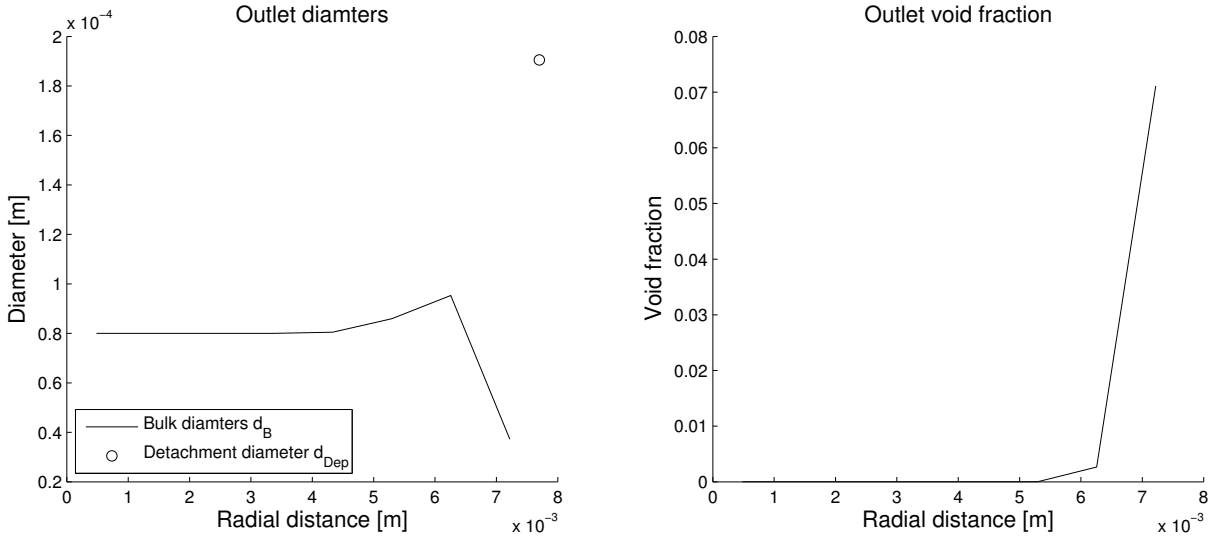

Figure 5.12: Left: Radial profile of the time-averaged bulk diameter at the outlet and the detachment diameter, from Ünal's diameter model, at the outlet of the pipe with $\kappa_i = 75000$. **Right:** Time-averaged radial profile of the void fraction at the outlet of the pipe with $\kappa_i = 75000$.

5.6 Mesh Sensitivity of Single-Phase Solution

The results from the mesh sensitivity test can be seen in Table 5.1. The numerical bulk temperature, at the output, is given for each mesh along with difference in percent between the analytical heat fluxes and the numerically calculated heat flux, as can be seen in the left column.

Radial cells	$T_{out,num}^{bulk}$	Difference
6	$489.084K$	6.986%
8	$490.333K$	4.612%
12	$491.505K$	2.399%
18	$492.287K$	0.917%

Table 5.1: Output from the single-phase mesh sensitivt test with different mesh refinements. The calculated output temperature and difference in the heat flux (calculated from energy balance) are both presented.

In Figure 5.13 the temperature along the vertical center line, for the different mesh refinements, can be seen. The temperature is taken at the cell center of the cells closest to the center line. Different mesh refinements will cause different distances between the cell center and the center line. This will impose a difference in the temperature plots for the different meshes. In Figure 5.14 the radial temperature profile, at a distance of 0.8 m from the inlet, for the different mesh refinements can be seen. The same effect imposing differences i the plots of the center line temperature will be present i the temperature profiles as well.

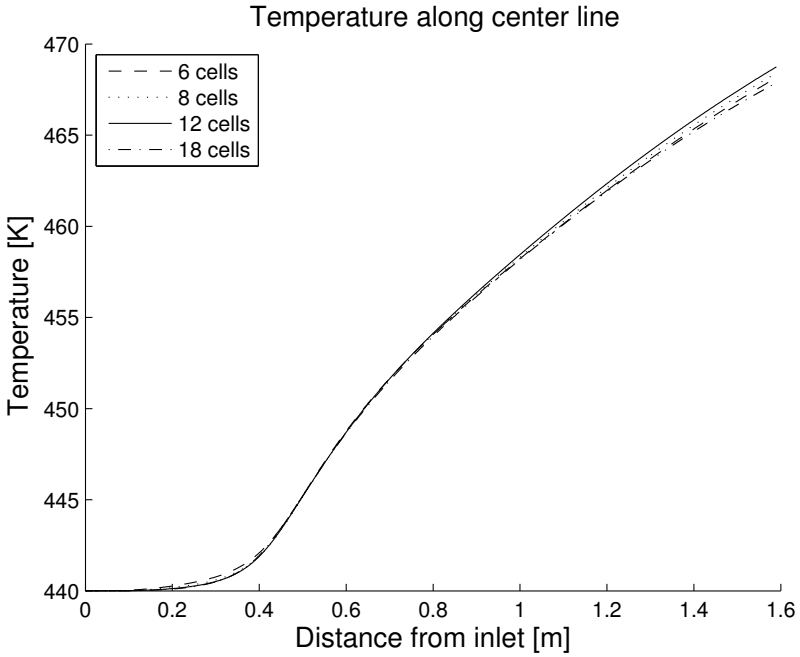

Figure 5.13: Resulting temperatures along the center line from the single-phase mesh sensitivity test. The results from four different mesh refinements are presented.

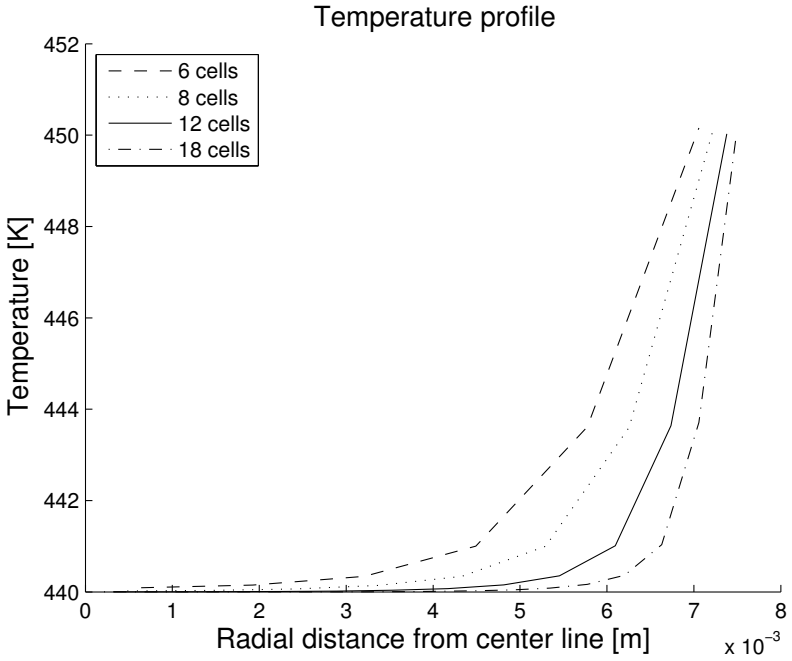

Figure 5.14: Radial temperature profiles at a distance of 0.8 m from the inlet, for meshes with different number of radial cells, from single-phase mesh sensitivity test.

6 Discussion

In this section the results presented in the previous section are analysed. Results deviating from the expected results presented by Kurul and Podowski [21] are discussed and suggestions for future corrections, to improve results, are introduced.

6.1 Solver

The implemented solver is stable and does not crash with reasonable input data. The input parameters, linked to boiling, are coded to be taken from case files. The boiling models have also been implemented as selectable from input. This means that new models easily can be added to the solver and selected from the case files, for example, the wall super heat model.

The code has been thoroughly documented with the intention of explaining what part of the model the code segment implements. Since the the original *twoPhaseEulerFoam*-solver lacked proper documentation, great effort has been spent on elucidating said codes purposes and documenting it for future projects.

6.2 Temperatures and ONB

The evaporation heat flux from the original case in Figure 5.1 seems to be underestimated by the present solver. The overall shape follows the same trend as the reference data until approximately 2.3 m from the inlet, where the increase suddenly declines. Note that this heat flux was subtracted from the total heat flux, to get the heat source for the liquid phase, hence, the temperature increase should slow down when q''_{evap} starts to develop. Comparing Figure 5.1, Figure 5.3 and the data from [21] indicates that the ONB can be found at the same position from the inlet as where q''_{evap} starts to grow.

Looking at Figure 5.3, the ONB can be seen in the data from [21] at approximately 1.4 m from the inlet, where T_{wall} stops increasing and where q''_{evap} starts to increase in Figure 5.1. The wall temperature from the present solver does not seem to react at the same position, despite the fact that q''_{evap} (from *myTwoPhaseEulerFoamBoiling*-solver) develops in the same manner at the first section of boiling. The wall temperature increases until around 2.3 m from the inlet, at which point it starts to decrease. The dotted line, which is the temperature in cells adjacent to the wall, suggests that its increase is continuous as the heat flux starts growing.

The evaporation heat flux from the alternative case in Figure 5.9 shows much better agreement with the reference data. Despite that, looking at T_{wall} from the same case, neither the wall temperature, nor the temperature in the cells adjacent to the wall, seem to react on the development of q''_{evap}. The lack of difference of the temperature in the wall cells could indicate a problem with how the heat equation was formulated or implemented. The high results for T_{wall} could be caused by the law-of-the-wall not working as expected. The dependence of the turbulent kinetic energy could cause a problem since the $k - \epsilon$-model might not be the optimal turbulence model for the present type of problem.

Another factor that could influence the temperature development is the vapour content in the system. As shown later in this chapter, the amount of vapour in the system is underestimated, in comparison with the reference data. This could result in to low enthalpies from the energy equation.

The temperature of the bulk and along the center line shows reasonable agreement with the reference data. Both are a few degrees lower than the results by [21]. This could be caused by the course mesh, that was adopted in the simulations.

The temperature profile at the outlet in Figure 5.4 looks like expected, with the highest temperatures closest to the wall. Supporting the previous observations, T_{wall} seems to be to high at approximately 560 K, close to 20 K higher than in the reference paper.

The overestimated T_{wall} and lack of clear ONB are problematic. The continuously increasing wall temperature causes a to high nucleation site density that could effect the evaporation heat flux and the vapour content.

In previous work, a common approach to calculate T_{wall} is through an iterative procedure [20]. In this procedure, q''_{wall} is partitioned into different parts when boiling starts. The modelled heat fluxes all depend on T_{wall} and should sum up to the same value as q''_{wall}. To calculate T_{wall}, an initial guess for T_{wall} is made. Based on this guess, the partitioned heat fluxes are calculated individually and then summed. The resulting sum is then compared to q''_{wall} to check if the resulting partitioning is consistent. If not, the procedure is repeated, with an adjusted T_{wall}, until a satisfying result is obtained. However, this procedure is not computationally efficient which was part of the purpose by this model, hence, a different approach is preferable.

Another way to proceed with this solver (with the purpose of an efficient solver kept) could be to investigate other law-of-the-wall's for the temperature. Many of them depend on the turbulent kinetic energy, hence, it would most likely need an investigation of other turbulence models. A first candidate could be the SST $k-\omega$ model, which has a better treatment of the wall area than the $k-\epsilon$ model. Since the turbulent kinetic energy is taken at the cell adjacent to the wall and the diameter restriction on the mesh refinement prevents very refined cells, in this area, this model could give better results for T_{wall}.

A faulty energy equation is easily checked by just switching to another verified equation. However, it seems more likely that the low vapour content, mesh refinement and turbulence model are all the cause of the deviant temperatures.

6.3 Bulk Diameter Models and Interfacial Heat Transfer Coefficients

The implemented solver, with the diameter model from Equation 3.66, led to numerical problems. Since the model would not give any positive values for the bulk diameter until the temperature of a cell had reached 516.5 K, the bulk diameter needed a minimum value to avoid division by zero in Equation 3.62 for H_{if} and in Equation 2.44 for a_i. The vapour started to develop before the temperature reached values of 516.5 K in any of the cells. This led to very large values of both a_i and H_{if}, hence a very large Γ_{cond}, causing a instabilities when boiling started. This could not be resolved by relaxation of the condensation factor, due to the high magnitudes Γ_{cond}.

Switching to the expression Equation 3.65 for H_{if} resulted in a solution with a much lower condensation rate. However, this solution showed problems with to high values of α_V. The solver proved sensitive to how the relaxation factors were set and often resulted in divergence of α_V.

Another problem with both these models for H_{if} (in combination with the diameter model by Kurul and Podowski) was that they underestimated q''_{evap}. To solve these issues the final solver was implemented with the H_{if}-model in Equation 3.65 and the diameter calculated with the κ_i-equation.

Because very small vapour contents imply small diameters, a_i was sensitive to fluctuations in α_V or d_B. This was most likely the problem with the bulk diameter model from [21]. Since the vapour started to develop before temperatures got high enough for bulk diameters larger than the minimum default, a_i increased very fast leading to instabilities.

6.4 Diameters and Vapour Void Fraction

In the left plot of Figure 5.5, a difference can be seen between the diameter of the bubbles in the wall cells, from the bulk diameter model and the detachment diameters from Ünal's model. The solver seems to predict the detachment diameters with reasonable values considering the restrictions of Ünal's model. The vapour void fractions in the right plot of Figure 5.5 shows that the calculated void fraction curve is lower than the reference curve. The bulk diameter seem to be over predicted in the early parts of the boiling section and approach reasonable values at the output of the pipe.

Observing Figure 5.6, the data supports the previous observation with a to high bulk diameter at the outlet. However, the detachment diameters show a good overlap with the diameters from the bulk diameter model.

The alternative case shows a completely different behaviour of the bulk diameter, in the left plot of Figure 5.11. The bulk diameter shows good agreement with the detachment diameter in the wall cells when boiling starts. Then it stays at an almost constant level to the outlet with much lower values than the detachment diameter. This could cause too high values of H_{if} and a_i, as seen in Figure 5.7, and hence, to high condensation rates leading to under predicted α_V. In the right plot of Figure 5.11, the previous conclusion is supported by a very low result for α_V.

Looking at the profiles of α_V in Figure 5.12, it reveals that the the vapour content in the bulk is very low. This supports that the condensation rate could be modelled to high. A range of values of κ_i were tested but did not seem to improve the result in any satisfying way.

The condensation rate appears as if dependent on the initial bulk diameter used before boiling starts. This is not optimal and the solver should be changed in a way that this parameter does not influence the result in such large extent. One way to achieve this could be to start calculating the condensation terms, when α_V rises above a predefined small threshold, large enough to not cause problems. The modelled bulk diameter will then have developed to accurate levels to get a consistent condensation rates.

A subject for future investigation could be to introduce the effects of bubble break up due to turbulent eddies, merging of bubbles due to random coalescence and coalescence due to acceleration of bubbles in the wake of preceding bubbles [19].

6.5 Velocities

The superficial velocity profiles in Figure 5.8 follow the same trend as in the reference paper. The vapour superficial velocity in the left plot is underestimated in the whole cross-sectional region. This is most likely an effect of the underestimated α_V.

The liquid superficial velocity in the right plot shows an underestimation in the outer most region and in a larger section around the center line. In between, there is a section where it is overestimated. These results could be caused by a combination of the underestimated α_V and the course mesh. The course mesh will effect the liquid velocities to a larger extent than the vapour velocity, since it was the only phase assumed to be turbulent. The $k-\epsilon$-model is sensitive to course meshes which could explain the deviations from the expected results.

This could be solved by a better mesh and a more suitable turbulence model. However, there are other factors to consider in order to improve the velocities. The interfacial forces, used in the present model, do not consider all physical effects of boiling. Other forces that could be interesting to introduce are the lift force, turbulent dispersion force and wall lubrication force [14].

6.6 Mesh Sensitivity

As can be seen in Table 5.1 the error of the numerically calculated heat flux is declining with an increasing number of cells. This convergence of the heat flux indicates that the solver is still dependant of the mesh when it is coarse. However, the declining of the error indicates that the solver in fact is independent of the mesh size when it is small enough, in the single-phase case.

The temperature along the center line for each mesh shows good alignment between the meshes in Figure 5.13. The slight difference in temperature close to the outlet originates from the method of obtaining the center line temperature.

The center line temperature and wall temperature shows good alignment between the meshes in Figure 5.14. The difference close to the wall is also caused by the difference in cell size. It causes the distance from the cell center of the cells closest to the wall to be smaller for the more refined meshes. However, the areas of the temperature profile with large temperature gradients

differs. This could affect the modelled wall temperature since it depends on the temperature at cell adjacent to the wall. This suggest that a more refined mesh should be used to improve the accuracy of the temperature.

6.7 Mesh Restriction

Despite the good results from the single-phase case the solver imposes restrictions on the mesh. The cell sizes must be large enough to encapsulate the full size bubbles at the wall and in the bulk. A to small cell size would lead to cell volumes smaller than the volumes of the bubbles calculated in the cells. This would result in a volume of vapour, larger than cell volume, being squeezed into the cell. That is, more vapour than would physically fit into the cell is squeezed into it, which is not physically possible.

Since the results indicate that the turbulence causes problems and the turbulence model is sensitive to mesh refinements. This calls for future attention on this issue, to generate a more robust solver. To achieve this, a few terms need to be handled in a different manner. The implementation of Γ_{evap} as a volumetric source term should, preferably, be reformulated into a boundary condition for the α_V-equation. This is not a trivial task, but ultimately needed to improve the mesh independence of the solver. Attention should also be spent on how the bulk diameters in each cell is modelled in a cell with smaller dimensions than the diameter.

Defining the diameter in cells smaller than the bubble itself is also a complicated task. The diameter would need to be distributed over several cells, in some manner, and then a correct way to calculate the volume occupied by vapour would be needed.

7 Conclusion

The stability of the *myTwoPhaseEulerFoamBoiling*-solver and the preparation of the code (for future development) was one of the main goals, hence, this part could be considered completed. The code and this document should give an easy understanding of the code and how to implement future changes.

Since the bulk diameter model seems to cause some problems with the solution, this part of the solver should be subject to future improvements. This could be done by adding source and sink terms to the a_i-equation and finding an appropriate way to handle the small bulk diameter.

The restrictions on the mesh refinement imposed by the diameter models are also problematic. In both the mesh sensitivity test and the center line temperature plots (Figure 5.2) this is evident. The coarse mesh is also a possible source of error, when modelling the turbulence, and, hence, when modelling the wall temperature. To avoid this restriction, a different approach is needed for the mass transfer terms and the diameter models that do not define the diameter per cell.

The evaporation rate, Γ_{evap}, should preferably be defined as a boundary condition for α_V instead of a volumetric source term. This could be a complicated task since it includes formulating an analytical expression for this boundary condition. The expression should include the related the physical quantities that influence the evaporation, like detachment diameter and nucleation site density, which makes it non-trivial. The condensation rate could still be a volumetric source but the modelling of the bulk diameters should be grid-independent.

The independence of the wall cells would also remove the need to work with separate cells in the mesh, making the solver more suitable for parallelization of the solution procedure.

Concerning the long time goals of ABB, two major parts of the solver need to be changed. The first part is to make it independent of the geometry. To achieve this the methods to calculate the bulk values of the velocity and the temperature need to be changed. Also, parts of the solver that depends on geometric properties, like models including Reynold and Nusselt numbers, need to be reworked. The second part consist of making the solver valid for other liquids than water. To do this, all correlations that have been specifically determined for water, like Jens-Lottes correlation, need to be replaced.

Despite many of the properties being over or under predicted, many of them follow the same trends as was expected. This leads to the conclusion that the outcome of this project is useful for future projects at ABB. The theory governing the fundamentals of a boiling model has been introduced and the base for the solver has been developed. This will save time in the future development of a robust solver, that can be used in ABB's applications.

References

[1] J. D. Anderson. Modern compressible flow : with historical perspective, 2003.

[2] ANSYS, Inc. *ANSYS CFX-Solver Theory Guide*, 12.1 edition, 5 November 2009.

[3] G.G. Bartolemei and V.M. Chanturiya. Experimental study of true void fraction when boiling subcooled water in vertical tubes. *Thermal Engineering*, 14(2):123–128, 1967.

[4] Jayatilleke C. The influence of prandtl number and surface roughness on the resistance of the laminar sublayer to momentum and heat transfer. *Prog. Heat Mass Transfer*, 1, 1969.

[5] V.P. Carey. *Liquid-Vapor Phase-Change Phenomena.* Taylor & Francis, fifth edition, 1992. ISBN 1-56032-074-5.

[6] Robert Cole. A photographic study of pool boiling in the region of the critical heat flux, 1960.

[7] L. Davidson. Fluid mechanics, turbulent flow and turbulence modelling, 2014. http://tfd.chalmers.se/~lada/MoF/lecture_notes.html.

[8] Fluent, Inc. *FLUENT 6.3 User's Guide*, 9 September 2006.

[9] OpenFOAM Foundation. Reporting bugs. `http://www.openfoam.org/mantisbt/view.php?id=1291`. Accessed: 2014-08-31.

[10] A Ghione. Development and validation of a two-phase cfd model using openfoam. Master's thesis, KTH, Royal Institute of Technology, 2012.

[11] M. Greco. Development and implementation of a subcooled boiling model. Master's thesis, KTH, Royal Institute of Technology, 2009.

[12] A. Henryk. *Thermal-Hydraulics in Nuclear Systems.* 2010. Qc 20120217.

[13] F.P. Incropera and D.P. DeWitt. *Fundamentals of Heat and Mass Transfer 5th.* Wiley, fifth edition, 2002. ISBN 0-471-38650-2.

[14] M. Ishii and T. Hibiki. *Thermo-Fluid Dynamics of Two-Phase Flow.* SpringerLink : Bücher. Springer, 2010.

[15] M. Jischa and H.B. Rieke. About the prediction of turbulent prandtl and schmidt numbers from modeled transport equations. *International Journal of Heat and Mass Transfer*, 22:1547–1555, 1979.

[16] Z. Naumann L. Schiller. A drag coefficient correlation. *Z. Ver. Deutsch. Ing.*, 77, 2004.

[17] B.E. Launder and D.B. Spalding. The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 3(2):269 – 289, 1974.

[18] S. Toll M. Ekh. Mechanics of solids & mechanics of fluids. part i: Fundamentals, 2013. Division of Material and Computational Mechanics.

[19] S. Kim M. Ishii and J. Kelly. Development of interfacial area transport equation. *Nuclear Engineering and Technology*, 37(6):525–536, 2005.

[20] E. Michta. Modeling of subcooled nucleate boiling with openfoam. Master's thesis, KTH, Royal Institute of Technology, 2011.

[21] Kurul N. and Podowski M.Z. Multidimensional effects in forced convection subcooled boiling. *Proceedings of the Ninth International Heat Transfer Conference*, 2:21–26, 1990.

[22] OpenCFD. *OpenFOAM - The Open Source CFD Toolbox - Programmer's Guide*. OpenCFD Ltd., United Kingdom, 2.3.0 edition, 5 February 2014.

[23] OpenCFD. *OpenFOAM - The Open Source CFD Toolbox - User's Guide*. OpenCFD Ltd., United Kingdom, 2.3.0 edition, 5 February 2014.

[24] OpenFOAMWiki. Contrib/makeaxialmesh. `http://www.openfoamwiki.net/index.php/Contrib_MakeAxialMesh`. Accessed: 2014-07-31.

[25] W.E. Ranz and W.R. Jr Marshall. Evaporation from drops. *Chemical Engineering Progress*, 48:141–146, 1952.

[26] H. Rusche. *Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phae Fractions*. Imperial College, London, 2002. PhD Thesis.

[27] B-U. Bae H-Y. Yoon D-J. Euh C-H. Song and G-C Park. Computational analysis of a subcooled boiling flow with a one-group interfacial area transport. *Journal of NUCLEAR SCIENCE and TECHNOLOGY*, 45(4):341–351, 2008.

[28] Dougall S.R. and Rohsenow M.W. Film boiling on the inside of vertical tubes with upward flow of the fluid at low vapor qualities. *Technical report No. 9079-26*, 1963. Department of Mechanical Engineering, MIT, Massachusetts Institute of Technology.

[29] H. K. Versteeg and W. Malalasekera. *An introduction to computational fluid dynamics*. Pearson Education Limited, second edition, 2007.

[30] Rubesin M. W. Viegas J. R. and Horstman C. C. On the use of wall functions as boundary conditions for two-dimensional separated compressible flows. *Technical Report AIAA-85-0180*, 1985. AIAA 23rd Aerospace Sciences Meeting.

[31] C. Morel W. Yao. Volumetric interfacial area prediction in upward bubbly two-phase flow. *International Journal of Heat and Mass Transfer*, 47:307–328, 2004.

[32] Ünal H. C. Maximum bubble diameter, maximum bubble-growth time and bubble-growth rate during the subcooled boiling of water up to 17 mn/m 2. *International Journal of Heat and Mass Transfer*, 19:643–649, 1976. AIAA 23rd Aerospace Sciences Meeting.

A Create Mesh

The mesh used in this thesis was created with the *blockMesh* tool that comes with the OpenFOAM installation. To create the wedge shaped mesh the utilities *makeAxialMesh* version 2.x and *collapseEdges* were used. *makeAxialMesh* had to be downloaded from [24], extracted and placed in the users utility folder.

The mesh was created in a separate case in containing the usual sub directories *0*, *constant* and *system* with the *polyMesh* folder in *0* and not in *constant*. A file *blockMeshDict* was created in *polyMesh* with the mesh defintion of the mesh as a cuboid in the standard way, see [23]. The *constant* folder was left empty.

The *system* directory contained the files *fvSolution*, *fvSystem*, *controlDict*, *rotationDict*, *meshQualityDict* and *collapseDict*. The *rotationDict* contained the parameters for *makeAxialMesh* to create a axial symmetric mesh, *meshQualityDict* and *collapseDict* was used by *collapseEdges* to adjust edges and points in the wedge shaped mesh. For details about how to write these files there exist examples of *meshQualityDict* and *collapseDict* in the *collapseEdge* folder under utilities in the OpenFOAM 2.3.0 installation. An example case of *makeAxialMesh* is also provided with the download of the utility.

To complete the mesh creation, *blockMesh* was first run in the case directory. Then *makeAxialMesh* was executed and finally *collapseEdges*. To make sure that the mesh was not corrupted during the creation one could run *checkMesh*. The final mesh was then located under the latest time step directory in a folder named *polyMesh*. This complete folder was then copied into the *constant* folder of the case to be simulated.

B myTwoPhaseEulerFoamBoiling.C

```
40 #include "fvCFD.H"
41 #include "twoPhaseSystem.H"
42 #include "PhaseIncompressibleTurbulenceModel.H"
43 #include "pimpleControl.H"
44 #include "IOMRFZoneList.H"
45 #include "fixedFluxPressureFvPatchScalarField.H"
46
47 //  #include "boilingModel.H"
48
49 // #include "IFstream.H"
50 // #include "OFstream.H"
51
52
53 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
54
55 int main(int argc, char *argv[])
56 {
57
58     #include "setRootCase.H"
59
60     #include "createTime.H"
61     #include "createMesh.H"
62     #include "readGravitationalAcceleration.H"
63
64     #include "createFields.H"
65     #include "createMRFZones.H"
66     #include "initContinuityErrs.H"
67     #include "readTimeControls.H"
68     #include "CourantNos.H"
69     #include "setInitialDeltaT.H"
70
```

```
71      pimpleControl pimple(mesh);
72
73      // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
74
75      Info<< "\nStarting time loop\n" << endl;
76
77      bool boilingOn = false;
78
79      while (runTime.run())
80      {
81          #include "readTimeControls.H"
82          #include "CourantNos.H"
83          #include "setDeltaT.H"
84
85          runTime++;
86          Info<< "Time = " << runTime.timeName() << nl << endl;
87
88      // Reset boiling properties and calculate simpler global properties
89      #include "boilingModels.H"
90
91      // Calculate the real pressure field
92      volScalarField pReal = p + pAdd;
93
94      // Calculate calculate the condensation rate
95      #include "condensationModel.H"
96
97      // Calculate boiling properties only defined at the wall
98      #include "evaporationModel.H"
99
100     // Pass boiling parameters to the twoPhaseSystem-class to be
101     // able to access them from the diameterModel
102     N = Nsite;
103         f = freqBub;
104         dDep = diameterDep;
105         Aw = wallCellArea;
106
107         // Pressure-velocity PIMPLE corrector loop
108         while (pimple.loop())
109         {
110
111         // Calls the solve function in twoPhaseSystem.C and calculates void ↩
                fraction(alpha)
112             fluid.solve(GammaCond, GammaEvap, SuAlpha, SpAlpha);//, DDtRho1, ↩
                    DDtRho2);
113
114             // Calculate mixture density
115             rho = fluid.rho();
116
117         // Update bulk diameters
118             fluid.correct();
119
120         // Save the diameters for output when diameterModel is constant
121 //      diameter = phase1.d();
122
123         // Solve the energy equation eq. (3.13)
124             #include "EEqns.H"
125
126         // Solve the momentum equations, eq. (3.4) and eq (3.6)
127         #include "UEqns.H"
128
129             // Pressure corrector loop
130             while (pimple.correct())
131             {
```

```
132         // Solves the pressure equation explained in sec (4.2.4)
133                 #include "pEqn.H"
134             }
135
136             // Calculate the material derivatives of the velocities
137             #include "DDtU.H"
138
139             // Correct for turbulence
140             if (pimple.turbCorr())
141             {
142                 fluid.correctTurbulence();
143             }
144
145         }
146
147         // Saving mass transfer rates for relaxation
148         GammaEvapOld=GammaEvap;
149     GammaCondOld=GammaCond;
150
151     // Calls write function
152         #include "write.H"
153
154         Info<< "ExecutionTime = "
155             << runTime.elapsedCpuTime()
156             << " s\n\n" << endl;
157     }
158
159     Info<< "End\n" << endl;
160
161     return 0;
162
163 }
```

C Boiling Terms

C.1 condensationModel.H

```
11      // Calculating the trubulent thermal conductivity through the effective ↩
            kinematic
12      // viscosity: lambdaT = Cpv2*rho2*(nuEff - nu2)/Pr2T
13      volScalarField lambdaT = Cpv2*rho2*(phase2.turbulence().nuEff()-mu2/rho2)/↩
            Pr2T;
14
15      // Calculate the interfacial heat transfer coefficient according to eq. ↩
            (3.65)
16      hC = rho2*Cpv2*Foam::sqrt(4.0/3.14159*Foam::mag(U1-U2)/diameter*thermCond2/(↩
            rho2*Cpv2)*1.0/(1.0+lambdaT/thermCond2));
17
18      // Calculate temporary condensation rate. Calculated according to eq. (3.61)
19      volScalarField GammaCondTemp = Foam::max(hC*(6.0*alpha1/diameter)*(Tsat-↩
            thermo2.T())/latHeatEvap,hC*(0.0*alpha1/(diameter))*(Tsat-thermo2.T())/↩
            latHeatEvap);
20
21      // Set the time-relaxed condesation rate
22      GammaCond = condRelaxFactor*GammaCondTemp+(1.0-condRelaxFactor)*GammaCondOld↩
            ;
```

C.2 evaporationModel.H

```
15      /********************* Integrate output temperature ********************
16       ***************** Used to get output bulk temperature *****************/
17
18      // Locat label of output boundary
19      label patchOutlet=mesh.boundaryMesh().findPatchID("outlet");
20
21      // Get output patch
22      const fvPatch& thePatchItselfOutlet=mesh.boundary()[patchOutlet];
23
24      // Get surface areas
25      const surfaceScalarField& magSf = mesh.magSf();
26
27      // Initiate output bulk temperature parameters
28      scalar areaCellOut = 0.0;
29      scalar totAreaOut = 0.0;
30      scalar ToutBulk = 0.0;
31
32      // Loop through the cells on the output patch to integrate ToutBulk
33      forAll(thePatchItselfOutlet,iface)
34      {
35      label icell = thePatchItselfOutlet.faceCells()[iface];
36      areaCellOut = magSf.boundaryField()[patchOutlet][iface];
37      totAreaOut += areaCellOut;
38      ToutBulk += areaCellOut*thermo2.T()[icell];
39      }
40
41      // Divide with the total area of output pacth to acheve mean value.
42      ToutBulk = ToutBulk/totAreaOut;
43
44      // Print results to output
45      Info<< " ToutBulk " << ToutBulk << "\n\n" << endl;
46
47
48
49
50
51      /********************* Manipulations of data only * *********************
52       ************************** in the wall cells ****************************/
53
54      // Get label of the boundary "walls"
55      label patchWall=mesh.boundaryMesh().findPatchID("walls");
56
57      // Get the "walls" patch
58      const fvPatch& thePatchItselfWall=mesh.boundary()[patchWall];
59
60      // Get faces of the mesh
61      const faceList & ff = mesh.faces();
62
63      // Get points of the mesh
64      const pointField & pp = mesh.points();
65
66
67
68
69
70      /************** Loop to damp the heat flux in the first cells ********/
71
72  //  // Set factors to damp the heat flux at the first n cells as the nominator
73  //  // and denominator of a fraction
74  //  scalar heatFluxDamp = 100.0;
```

```
75 //  scalar iHFD = 0.0;
76 //  forAll(thePatchItselfWall,iFace)
77 //  {
78 //      // Get label of wall cell
79 //      label iCell = thePatchItselfWall.faceCells()[iFace];// Cell index for ↩
       each face in the "walls"-boundary
80 //
81 //      // Calculate damped heat flux
82 //      wallHeatFlux[iCell] = qTot.value()*(iHFD/heatFluxDamp);
83 //
84 //      // Update damping factor
85 //      if(iHFD<heatFluxDamp)
86 //      {
87 //      iHFD++;
88 //      }
89 //  }
90
91      /*****************************************************************/
92
93
94      // Start loop for calculating boiling properties
95      forAll(thePatchItselfWall,iFace)
96      {
97
98      // Get label of wall cell
99      label iCell = thePatchItselfWall.faceCells()[iFace];
100
101     // Get cell center of wall cell
102     vector iCellCentre = mesh.cellCentres()[iCell];
103
104     // Define point close to center line at the same position as the wall cell ↩
          in y-direction
105     vector centerLinePoint(1.0e-6,iCellCentre.y(),0.0);
106
107     // List labels of bulk cells
108     label centerCell = mesh.findCell(centerLinePoint);
109
110
111
112
113
114     /************Calculate bulk temperature and velocity**************/
115
116     // Loop to integrate bulk temperature and y-velocity component
117     for(int wallCellLabel = centerCell; wallCellLabel < (iCell + 1); ↩
          wallCellLabel++)
118     {
119         vol[iCell] += mesh.V()[wallCellLabel];
120         Tbulk[iCell] += thermo2.T()[wallCellLabel]*mesh.V()[wallCellLabel];
121         Ubulk[iCell] += U2[wallCellLabel].y()*mesh.V()[wallCellLabel];
122     }
123
124     // Divide with the total area of output pacth to acheve mean value.
125     Tbulk[iCell] /= vol[iCell];
126     Ubulk[iCell] /= vol[iCell];
127
128
129     /***********Alternative bulk temperature, taken at yPlus=250*********/
130     /*****************************Not used*****************************/
131 //  scalar xBulk = 250*Foam::sqrt(xDim/2/(Foam::mag(U2[iCell].y())+scalar(1.0e↩
    -12))*mu2[iCell]/rho2[iCell]);
132 //  vector bulkPointT(Foam::max(pipeRadius.value()-xBulk,1.0e-6),iCellCentre.y()↩
    ,0);
```

```
133 //  List labels of bulk cells
134 //  label bulkCellT = mesh.findCell(bulkPointT);
135 //  Tbulk[iCell] = thermo2.T()[bulkCellT];
136 //  Tbulk[iCell] = thermo2.T()[bulkCell];
137     /*****************************************************************/
138
139
140
141
142
143     /********** Calculate cell dimensions and wall cell area **********/
144
145     // Get dimensions of wall cell
146     const cell & cc = mesh.cells()[iCell];
147     labelList pLabels(cc.labels(ff));
148     pointField pLocal(pLabels.size(),vector::zero);
149     forAll(pLabels,pointi)
150     {
151         pLocal[pointi]=pp[pLabels[pointi]];
152     }
153     scalar xDim = Foam::max(pLocal & vector(1,0,0)) - Foam::min(pLocal & vector↩
        (1,0,0));
154     scalar yDim = Foam::max(pLocal & vector(0,1,0)) - Foam::min(pLocal & vector↩
        (0,1,0));
155     scalar zDim = Foam::max(pLocal & vector(0,0,1)) - Foam::min(pLocal & vector↩
        (0,0,1));
156
157     // Calculate area/volume for the wall cell
158     wallCellArea[iCell]= 1/xDim;
159
160     /*****************************************************************/
161
162
163
164
165
166
167     /********** Calculate wall temperature from law-of-the-wall **********/
168     /**** according to the model in senction (3.5.2) of the reference*****/
169
170     // Define von Karman constant
171     scalar karman = 0.4187;
172
173     // Define constant E
174     scalar E = 9.793;
175
176     // Define C_mu
177     scalar Cmu = 0.09;
178
179     // Get distance from cell center to wall
180     yP[iCell] = xDim/2;
181
182     // Reset dimensionless wall temperature
183     Tstar[iCell] = 0;
184
185     // Calculate wall area
186     scalar wallArea = yDim*zDim;
187
188     // Calculate dimensionless wall distance according to eq. (3.35)
189     yStar[iCell] = rho2[iCell]*Foam::pow(Cmu,0.25)*Foam::pow(k2[iCell],0.5)*yP[↩
        iCell]/mu2[iCell];
190
191     // Guess inital yStarT
```

```
192     yStarT[iCell] = 0.5;
193
194     // Loop to calculate yStarT where law-of-the-wall changes definition
195     scalar error = 1000;
196     scalar nrIter = 0;
197     while((error > 1) & (nrIter < 100))
198     {
199         nrIter++;
200         scalar TstarLin = Pr2[iCell]*yStarT[iCell];
201         scalar TstarLog = Pr2T*(Foam::log(E*yStarT[iCell])/karman + 9.24*(Foam::↩
                pow(Pr2[iCell]/Pr2T,0.75) -1.0)*(1+0.28*Foam::exp(-0.007*Pr2[iCell]/↩
                Pr2T)));
202         error = Foam::mag(TstarLin - TstarLog);
203         yStarT[iCell] += 0.25;
204     }
205
206     // If statment to select correct law-od-the-wall definition
207     if (yStar[iCell] > yStarT[iCell])
208     {
209         // Calculate Tstar with the logarithmic definition
210         Tstar[iCell] = Pr2T*(Foam::log(E*yStar[iCell])/karman + 9.24*(Foam::pow(↩
                Pr2[iCell]/Pr2T,0.75) -1.0)*(1.0+0.28*Foam::exp(-0.007*Pr2[iCell]/↩
                Pr2T)));
211
212     } else
213     {
214         // Calculate Tstar with the linear defintion
215         Tstar[iCell] = Pr2[iCell]*yStar[iCell];
216     }
217
218     // Calculate the wall temperature
219     TwallWater[iCell] = Tstar[iCell]*qTot.value()/(rho2[iCell]*Cpv2[iCell]*Foam↩
            ::pow(Cmu,0.25)*Foam::pow(k2[iCell],0.5)) + thermo2.T()[iCell];
220
221     /*****************************************************************/
222
223
224
225
226
227
228     /************Wall temperature according to Dittus-Boelter************/
229
230     ReyNr[iCell] =  rho2[iCell]*0.9978*2*pipeRadius.value()/mu2[iCell];
231
232     TwallWaterDB[iCell]= Tbulk[iCell] + qTot.value()/(thermCond2[iCell]*0.023*↩
            Foam::pow(ReyNr[iCell],0.8)*Foam::pow(Pr2[iCell],0.4)/(2*pipeRadius.↩
            value()));
233
234     /*****************************************************************/
235
236     // Calculate wall super heat iccording to selection from input
237     #include "wallSuperHeat.H"
238
239     // If statement to check if boiling occures in the cell according to eq. ↩
            (3.34)
240     if(TwallWater[iCell]<(Tsup[iCell]+Tsat[iCell]))
241     {
242
243         // Calculate heat flux to phase2, the total heat flux goes to heating ↩
                since
244         // no boiling occures
245         qWall2[iCell] = wallHeatFlux[iCell]*wallCellArea[iCell];
```

```
246
247     } else
248     {
249
250         // Calculate the nucleation site density according to eq. (3.43)
251         Nsite[iCell] = Foam::pow(210*(TwallWater[iCell]-Tsat[iCell]),1.805);
252
253         // Calculate the bubble detachment diameter according to Unals model
254         #include "departureDiameters.H"
255
256         // Calculate the bubble detachment frequency according to eq. (3.44)
257         freqBub[iCell] = Foam::sqrt((4.0/3.0)*((Foam::mag(g[2].y()))*(rho2[iCell↩
                ]-rho1[iCell])))/(diameterDep[iCell]*rho2[iCell]));
258
259         // Set diameters below the limit of what is valid in Unals model to zero
260         if(diameterDep[iCell] < 0.00008)
261         {
262         diameterDep[iCell] = 0.0;
263         }
264
265         // Calculate the heat flux going to evaporation according to eq. (3.42)
266         qEvap[iCell] = Foam::min((3.14159/6.0)*Foam::pow(diameterDep[iCell],3)*↩
                rho1[iCell]*latHeatEvap.value()*Nsite[iCell]*freqBub[iCell],↩
                wallHeatFlux[iCell]);
267
268         // Calculate a temporary evaporation rate according to eq. (3.56)
269         scalar GammaEvapTemp = qEvap[iCell]*wallCellArea[iCell]/(latHeatEvap.↩
                value()+Cpv2[iCell]*(Tsat[iCell]-Tbulk[iCell]));
270
271         // Set time-relaxed evaporation rate
272         GammaEvap[iCell] = evapRelaxFactor*GammaEvapTemp+(1.0-evapRelaxFactor)*↩
                GammaEvapOld[iCell];
273
274         // Calculate heat source with evaporation heat flux subtracted
275         qWall2[iCell] = (wallHeatFlux[iCell]-qEvap[iCell])*wallCellArea[iCell];
276
277         // Check that mesh is not to refined for bubble diameters at the wall
278         if ((diameterDep[iCell]>xDim) || (diameterDep[iCell]>yDim))
279         {
280         Info<< "ERROR : Mesh to refined to fit bubble departure diameter in cell↩
                " << "\n\n" << endl;
281         Info<< "Refine mesh at wall " << "\n\n" << endl;
282         Info<< "diameterDep[iCell] = " << diameterDep[iCell] << "\n\n" << endl;
283         Info<< "xDim = " << xDim << "\n\n" << endl;
284         Info<< "yDim = " << yDim << "\n\n" << endl;
285         Info<< "================================ qEvap[iCell] = " << qEvap[iCell] ↩
                << "\n\n" << endl;
286         Info<< "================================ qWall2[iCell] = " << qWall2[iCell↩
                ] << "\n\n" << endl;
287
288         return 1;
289         }
290
291     }
292
293     }
```

C.3 departureDiameters.H

```
15      /************** Calculating the single-phase forced convection ↩
            **************
16       ************** heat transfer coefficient for the heated surface************↩
             */
17
18      // Calculate faning Reynold numbers according to eq. (3.55)
19      ReynoFan[iCell] = (rho2[iCell]*2.0*pipeRadius.value()*Foam::mag(U2[↩
            centerCell]))/mu2[iCell];
20
21      // Initial guess for fan coefficient
22      Cf[iCell] = 0.062;
23
24      // Iterate to get the fan friction coefficient according to eq. (3.54)
25      for(int i=0; i <10; i++)
26      {
27      Cf[iCell]=1.0/(Foam::log(max(ReynoFan[iCell]*Cf[iCell],1.0))/0.435+5.05);
28      }
29
30      // Calculate Stanton number according to eq. (3.53)
31      Stanton[iCell] = Foam::pow(Cf[iCell],2.0)/(1.0-1.783*Cf[iCell]);
32
33      // Calculate single-phase heat transfer coefficient according to eq. (3.52)
34      h1F[iCell] = Stanton[iCell]*rho2[iCell]*Cpv2[iCell]*Foam::mag(U2[centerCell↩
            ]);
35
36
37      /************** Calculating coefficients is Unals model **************
38      ********************** and departure diameter **********************/
39
40      // Calculate factor B from Unals model, given in eq. (3.49)
41      scalar B = (Tsat[iCell]-Tbulk[iCell])/(2.0*(1.0-rho1[iCell]/rho2[iCell]));
42
43      // Calculate factor C from Unals model, given in eq. (3.51)
44      scalar C = (latHeatEvap.value()*mu2[iCell]*(Foam::pow(Cpv2[iCell]/(0.013*↩
            latHeatEvap.value()*Foam::pow(Pr2[iCell],1.7)),3.0)))
45          /(Foam::sqrt(fluid.sigma().value()/(Foam::mag(g[2].y())*(rho2[iCell]-↩
                rho1[iCell])))));
46
47      // Calculate factor A from Unals model, given in eq. (3.50)
48      scalar A = Foam::pow(Foam::max(wallHeatFlux[iCell]-h1F[iCell]*(Tsat[iCell]-↩
            Tbulk[iCell]),0.0),1.0/3.0)*thermCond2[iCell]*Foam::sqrt((rhoSurf.value↩
            ()*CpSurf.value()*lambdaSurf.value())/(thermCond2[iCell]*rho2[iCell]*↩
            Cpv2[iCell]))
49            /(2.0*Foam::pow(C,1.0/3.0)*latHeatEvap.value()*rho1[iCell]*Foam::sqrt↩
                (3.14159*thermCond2[iCell]/(rho2[iCell]*Cpv2[iCell])));
50
51      // Calculate factor Theta from Unals model, given in eq. (3.48)
52      scalar D = 1.0;
53      if (Ubulk[iCell] > 0.61)
54      {
55      D = Foam::pow(Ubulk[iCell]/0.61,0.47);
56      }
57
58      // Calculate the departur diameter according to eq. (3.47)
59      diameterDep[iCell] = Foam::max(((2.42e-5)*Foam::pow(pReal[iCell],0.709)*A)/(↩
            Foam::sqrt(B*D)),0.0);
```

C.4 boilingModel.H

```
12      // Reset fields
13      Tbulk = Tbulk*0.0;
14      Ubulk = Ubulk*0.0;
15      vol = vol*0.0;
16      qWall2=qWall2*0;
17      GammaEvap=GammaEvap*0;
18      GammaCond=GammaCond*0;
19      SpAlpha=SpAlpha*0;
20      SuAlpha=SuAlpha*0;
21      Nsite = Nsite*0;
22      freqBub=freqBub*0;
23      diameterDep = diameterDep*0;
24      Nu=Nu*0;
25      hC=hC*0;
26      qEvap=qEvap*0;
27      ReynoBub = ReynoBub*0;
28      ReynoFan = ReynoFan*0;
29      ReyNr=ReyNr*0;
30      TwallWater=TwallWater*0;
31      Stanton=Stanton*0;
32
33      // Get boundBox to get mesh size
34      const boundBox meshSize = mesh.bounds();
35
36      // Get pipe radius
37      dimensionedScalar pipeRadius
38      (
39      "pipeRadius",
40      dimensionSet(0,1,0,0,0,0,0),
41      scalar(meshSize.max().x())
42      );
43
44      // Calculate the Prandtl numbers for phase 2
45      Pr2 = mu2*Cpv2/thermCond2;
46
47      // Initiate the friction coefficient
48      volScalarField Cf = ReynoFan;
49
50      // Calculate the thermal diffusivity of phase 2 [m^2/s]
51      thermDiff2 = mu2/Pr2/rho2;
52
53      // Calculate Reynold numbers based on bubble diameter, eq. (3.64)
54      ReynoBub = rho2*diameter*Foam::mag(U2-U1)/mu2;
55
56      // Calculate the local Nusselt numbers according to the Rans-Marshall ↩
            correlation, eq. (3.63)
57      Nu = 2 + 0.6*Foam::sqrt(ReynoBub)*Foam::cbrt(Pr2);
```

D Void fraction equation

```
510         forAll(dgdt_, celli)
511         {
512
513         // Calculate the source terms for the alpha-equation due to the ↩
                evaporation and condensation. The
514         // source term is divided into a implicit and explicit part according to↩
                 the theory in sec (4.2.1).
515         // A minimum of 1.0e-8 is needed for alpha1 to be considered non-zere, ↩
                hence condensation will occur.
516         if(alpha1[celli] > 1.0e-8)
517         {
518             SpAlpha[celli] = -cond[celli]/phase1_.rho()[celli]/alpha1[celli↩
                    ]-2.0*evap[celli]/phase1_.rho()[celli]-2.0*cond[celli]/phase2_.↩
                    rho()[celli];
519
520         SuAlpha[celli] = evap[celli]/phase1_.rho()[celli]+2.0*alpha1[celli]*evap↩
                [celli]/phase2_.rho()[celli]+2.0*alpha1[celli]*cond[celli]/phase1_.↩
                rho()[celli];
521
522         } else
523         {
524         SuAlpha[celli] = evap[celli]/phase1_.rho()[celli];
525         }
526
527
528         // Update the source terms for the alpha1-equation
529             if (dgdt_[celli] > 0.0 && alpha1[celli] > 0.0)
530             {
531                 Sp[celli] += (-dgdt_[celli]*alpha1[celli]+SpAlpha[celli]);
532                 Su[celli] += (dgdt_[celli]*alpha1[celli]+SuAlpha[celli]);
533             }
534             else if (dgdt_[celli] < 0.0 && alpha1[celli] < 1.0)
535             {
536                 Sp[celli] += (dgdt_[celli]*(1.0 - alpha1[celli])+SpAlpha[celli])↩
                        ;
537         Su[celli] += SuAlpha[celli];
538             }
539
540         }
```

E EEqns.H

```
16 {
17     // Implementation of energy equation, eq. (3.13)
18     fvScalarMatrix he2Eqn
19     (
20         fvm::ddt(alpha2, he2) + fvm::div(alphaPhi2, he2)
21
22         // Compressibity correction
23       - fvm::Sp(fvc::ddt(alpha2) + fvc::div(alphaPhi2), he2)
24       + (
25             he2.name() == thermo2.phasePropertyName("e")
26           ? fvc::ddt(alpha2)*p + fvc::div(alphaPhi2, p)
27           : -alpha2*dpdt
28         )/rho2
29
30       - fvm::laplacian(alpha2*(thermCond2/Cpv2/rho2+(phase2.turbulence().nuEff()↩
             -mu2/rho2)/Pr2T), he2)
31      ==
32
33       // Adding the heat flux as a source term at the near wall cells.
34       // The total heat flux is multiplied with the wall area per unit
35       // volume of each near wall cell
36       qWall2/rho2// *wallCellArea/rho2
37
38       // Adding term due to phase change
39       + (GammaCond*Cpv1*thermo1.T()-GammaEvap*he2)/rho2
40     );
41
42     he2Eqn.relax();
43     he2Eqn.solve();
44
45     thermo2.correct();
46 }
```

F UEqns.H

```
10 mrfZones.correctBoundaryVelocity(U1);
11 mrfZones.correctBoundaryVelocity(U2);
12 mrfZones.correctBoundaryVelocity(U);
13
14 fvVectorMatrix U1Eqn(U1, U1.dimensions()*dimVol/dimTime);
15 fvVectorMatrix U2Eqn(U2, U2.dimensions()*dimVol/dimTime);
16
17 // Gets drag coefficient
18 volScalarField dragCoeff(fluid.dragCoeff());
19
20 {
21     // Gets coefficients for forces
22     volScalarField virtualMassCoeff(fluid.virtualMassCoeff());
23     volVectorField liftForce(fluid.liftForce());
24     volVectorField wallLubricationForce(fluid.wallLubricationForce());
25     volVectorField turbulentDispersionForce(fluid.turbulentDispersionForce());
26
27     {
28     // Implementation of momentum equation for phase 1, eq. (3.4)
29         U1Eqn =
30         (
31             fvm::ddt(alpha1, U1)
32           + fvm::div(alphaPhi1, U1)
33           - fvm::Sp(fvc::ddt(alpha1) + fvc::div(alphaPhi1), U1)
34           + phase1.turbulence().divDevReff(U1)
35          ==
36           // Gravity term g transfered to pressure equatio
37           - fvm::Sp(dragCoeff/rho1, U1) // Explicit part included in pEqn
38           - alpha1*alpha2/rho1
39            *(
40                 liftForce
41               + wallLubricationForce
42               + turbulentDispersionForce
43            )
44           - virtualMassCoeff/rho1
45            *(
46                 fvm::ddt(U1)
47               + fvm::div(phi1, U1)
48               - fvm::Sp(fvc::div(phi1), U1)
49               - DDtU2
50            )
51       + GammaEvap*U2/rho1-fvm::Sp(GammaCond/rho1,U1)
52         );
53         mrfZones.addCoriolis(alpha1 + virtualMassCoeff/rho1, U1Eqn);
54         U1Eqn.relax();
55     }
56
57     {
58     // Implementation of momentum equation for phase 2, eq. (3.6)
59         U2Eqn =
60         (
61             fvm::ddt(alpha2, U2)
62           + fvm::div(alphaPhi2, U2)
63           - fvm::Sp(fvc::ddt(alpha2) + fvc::div(alphaPhi2), U2)
64           + phase2.turbulence().divDevReff(U2)
65          ==
66          // Gravity term g transfered to pressure equation
67           - fvm::Sp(dragCoeff/rho2, U2) // Explicit part included in pEqn
68           + alpha1*alpha2/rho2
```

```
69              *(
70                    liftForce
71                  + wallLubricationForce
72                  + turbulentDispersionForce
73               )
74             - virtualMassCoeff/rho2
75              *(
76                    fvm::ddt(U2)
77                  + fvm::div(phi2, U2)
78                  - fvm::Sp(fvc::div(phi2), U2)
79                  - DDtU1
80               )
81           + GammaCond*U1/rho2-fvm::Sp(GammaEvap/rho2,U2)
82           );
83           mrfZones.addCoriolis(alpha2 + virtualMassCoeff/rho2, U2Eqn);
84           U2Eqn.relax();
85       }
86
87   }
```

G pEqn.H

```
198     // Solves the pressure equatin in eq. (4.15) with the added source term "(↩
            GammaEvap-GammaCond)*(1/rho1 -1/rho2)"
199     // Since the solver was assumed incompressible, the only term to take into ↩
            account is pEqnIncomp which
200     // is derived from the continuity equation via div(phic), phic = alpha1*phi1↩
            +alpha2*phi2. Details about
201     // how similar pressure equations are derived can be found in Alberto ↩
            Ghiones thesis from 2012
202     // "Development and validation of a two-phase CFD model using OpenFOAM"
203         solve
204         (
205             pEqnComp1() + pEqnComp2() + pEqnIncomp -(GammaEvap-GammaCond)*(1/↩
                    rho1 -1/rho2),// - Source,
206             mesh.solver(p.select(pimple.finalInnerIter()))
207         );
```