University of South-Eastern Norway

www.usn.no

FMH606 Master's Thesis 2024

Process Technology

Computational modelling of two-phase oil-water flow in pipes using OpenFOAM

Martin Færøvik Hamre

Faculty of Technology, Natural sciences and Maritime Sciences
Campus Porsgrunn

University of South-Eastern Norway

www.usn.no

Course: FMH606 Master's Thesis, 2024
Title: Computational modelling of two-phase oil-water flow in pipes using OpenFOAM
Number of pages: 128
Keywords: CFD, OpenFOAM, multiphase flow modelling, oil-water flow, interface treatment, turbulence models.

Student: Martin Færøvik Hamre

Supervisor: Amaranath Sena Kumara

External partner: N/A

University of South-Eastern Norway

www.usn.no

Summary:

Two-phase oil-water flows in offshore pipelines are a common occurrence in the petroleum industry. As an oil reservoir mature, the presence of water in the retrieved oil increase. Understanding how this increased presence of water affects the hydrodynamic properties of the mixture is of great importance as it affects the operating conditions for the pipeline.

In this report, a review on the status of oil-water modelling is presented. The review covers the three main modelling approaches: empirical, analytical and numerical. The review highlights the different modelling techniques used in the studies and summarizes its findings. Additionally, an OpenFOAM CFD model based on the experimental work conducted by Kumara (2010) is made. The CFD model aims to predict the water volume fraction and mean axial velocity of a horizontal oil-water pipe flow at various operating conditions.

The findings from the review indicates that to increase the numerical accuracy for oil-water flows, the oil-water interface needs additional treatment. The presence of a turbulence damping scheme or boundary conditions are required to emulate the wall-like effect the oil-water interface represents. This wall-like effect was observed in the CFD simulations. The results showed that two-equation turbulence models like RNG k-ε, realizable k-ε and SST k-ω, were unable to accurately predict the complex nature of an oil-water flow. Of the three turbulence models, the SST k-ω model showed the best accuracy and should be used in any further research. The study concludes that the addition of a turbulence damping scheme, or similar interface treatment, is necessary to increase the numerical accuracy.

Preface

This master's thesis is written at and submitted to the University of South-Eastern Norway as the last part of the Master of Science program Process Technology. The fulfillment of this thesis would not have been possible without the support and guidance from several individuals.

I would like to thank my supervisor Dr. Amaranath Sena Kumara for guiding me throughout the thesis. The work in this thesis would not have been possible without his help and guidance. I want to issue a thank you to associate Professor Joachim Lundberg and Mathias Henriksen for helping me troubleshoot obstacles I've run into when using OpenFOAM.

I want to thank my family for believing in me and supporting me throughout my studies. A special thanks to my fellow students Pål and Sivert for making me and my partners life in Porsgrunn enjoyable.

Lastly, I would like to thank my partner Maja for her unwavering support throughout my studies. I want to thank her for putting up with my complaining and frustrations over the past 5 years. She has been invaluable as a discussion partner and has helped me see things from a different perspective. Additionally, I want to thank her for the countless hours she has spent proofreading my reports and generally supporting my studies. You truly are amazing.

Porsgrunn, 15th May 2024

Martin Færøvik Hamre

Contents

List of Figures

List of Tables

Nomenclature

Latin letters

A	Flow cross-sectional area [m^2]
B	Adjustable damping factor in Equation 2.20 [-]
C_μ	Coefficient in Equation 3.22 [-]
$C_{1\varepsilon}$	Coefficient in Equation 3.24 [-]
$C_{2\varepsilon}$	Coefficient in Equation 3.24 [-]
$C_{3\varepsilon}$	Coefficient in Equation 3.24 [-]
d	diameter [m]
D	pipe diameter [m]
	Drag force [N]
E	Eötvös number [-]
f	Friction factor [-]
F	Surface tension term [N/m^3]
g	Gravitational acceleration [m^2/s^2]
G	Production of turbulent kinetic energy [kg/ms^3]
I	Turbulence intensity [%]
k	Turbulent kinetic energy [m^2/s^2]
l	length scale [-]
L	Lift force[N]
M	Momentum transfer [kgm/s]
n	grid size [-]
N	Froude number [-]
p	Pressure [N/m^2]
Q	Volumetric flow rate [m^3/s]
r	Radial position [m]
	droplet radius [m]
R	pipe radius [m]
	Oil droplet radius [m]
	Universal gas constant [J/Kmol]
S	Slip ratio [-]
	Mean strain rate [1/s]

	Mass source term [kg/m^{3}s]
t	Time [s]
T	Absolute temperature [K]
u	Velocity [m/s]
U	Bulk velocity [m/s]
x	Cartesian axis direction [m]
y	Cartesian axis direction [m]
Y	Dissipation of turbulence kinetic energy and specific dissipation rate [kg/ms^3]
z	Cartesian axis direction [m]

Greek letters

α	Volume fraction of phase [-]
β	Closure coefficient in Equation 2.20
	Coefficient in Equation 3.52
ε	Turbulent kinetic energy dissipation rate [m^2/s^3]
η	Water hold-up [-]
θ	inclination angle [°]
λ	Water cut [-]
μ	Dynamic viscosity [m^2/s]
υ	Kinematic viscosity [m^2/s]
ρ	Density [kg/m^3]
σ	Surface tension coefficient [N/m]
	Turbulent Prandtl number [-]
τ	Wall shear stress [N/m]
φ	Angular velocity [1/s]
Γ	Effective diffusivity of turbulent kinetic energy and specific dissipation rate [kg/ms]
ω	Specific dissipation [1/s]
v	Coefficient in Equation 3.29 and 3.30 [-]
	Velocity scale [-]
γ	Compressibility factor [-]
Ω	Mean rate of rotation tensor [1/s]

Abbreviations

2D	Two dimensional
3D	Three dimensional
AR	Aspect ratio
CFD	Computational fluid dynamics
DNS	Direct numerical simulation
FVM	Finite volume method
PIP	Phase inversion point
PIV	Particle image velocimetry
RANS	Reynolds-averaged Navier-Stokes
Re	Reynolds number
ST	Stratified
SW	Stratified wavy
USN	University of South-Eastern Norway
VOF	Volume of Fluid

Subscripts

eff	Effective
f	fluid
Fr	Froude
g	Gas
I	Interface
k	Turbulent kinetic energy
	Represents oil and water in Equations 3.8-3.10
l	Liquid
o	Oil
so	Superficial for oil phase
sw	Superficial for water phase
t	Turbulent
w	Water

1 Introduction

1.1 Background

Two-phase oil-water flow in pipelines is a common occurrence in the petroleum industry. In a reservoir the presence of gas, oil and water is frequently found and produces multiphase mixtures when transported in pipelines. As an oil reservoir matures, the presence of gas and water increases as illustrated in Figure 1.1. Knowing how an increased percentage of water affects the hydrodynamic properties of the mixture is of great importance when designing equipment and operating offshore pipelines.

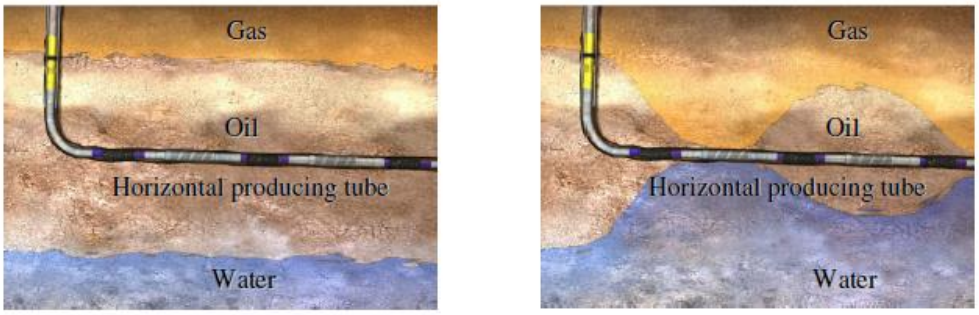

Figure.1.1 – Multiphase flow in transportation pipes due to water and gas coning: (a) Initial condition, (b) After production (Kumara 2010).

Understanding multiphase flows requires extensive experimental studies to produce reliable data. Experimental studies that aim to obtain such data require the implementation of sophisticated, and expensive, measurement techniques that are non-invasive (Kumara 2010). Recent years have seen increasing interest in the development of computational models for predicting multiphase flow fields. A major issue with current computational models is that the phases are treated as a bulk flow. This leads to loss in computational accuracy as the detailed hydrodynamic property of each phase is lost.

This thesis aims to review the current status of numerical modelling of two-phase oil-water flows, as well as creating a computational fluid dynamics (CFD) model based on the experimental study of Kumara (2010). The results from the CFD model are compared to the data from the experimental study.

1.2 Previous work

This thesis is a continuation of the work done by the group project of Vindenes et al. (2021) at the University of South-Eastern Norway (USN) during the fall of 2021. In the group project a well-defined pipe geometry and mesh was created, and basic initial simulations were performed. This thesis uses the computational domain created by Vindenes et al. (2021) and aims to develop the simulation strategy further.

1.3 Objectives

The main scope of this thesis is to perform CFD simulations of a horizontal pipe with an oil-water flow. The simulations are done using the OpenFOAM software. Within the scope there are three main objectives:

1) Perform a literature review of recent advances in the numerical modelling of two-phase oil-water models.
2) Implementation of computational model for oil-water flow using OpenFOAM. The model will attempt to accurately predict the mean axial velocity profile of both water and oil at varying mixture velocities and water cuts.
3) Validate the simulation data based on the experimental work performed by Kumara (2010)

The signed project proposal is found in Appendix A

1.4 Structure of thesis

- Chapter 2: Literature review of the recent advances in numerical modelling of two-phase oil-water flows with the aim of publishing.
- Chapter 3: Presentation of the methodology for CFD modelling of a two-phase oil-water flow.
- Chapter 4: Simulation strategy for the simulations presented in this thesis.
- Chapter 5: Presentation of the results and discussion of the comparison against experimental data.
- Chapter 6: Conclusion and suggestions for future work on oil-water simulations.

2 Literature review

In this section, basic physics and flow structure of an oil-water flow is presented and recent advances in oil-water modelling is reviewed.

2.1 Introduction

Multiphase flow is a well-established phenomenon within the process industry, particularly in the petroleum industry. Two-phase oil-water pipe flow has become of special importance over the years due to the maturing of oil wells. As the oil wells mature, the water cut in the extracted crude oil increases which leads to increasing technical difficulties during pumping and separation. This results in the reduced efficiency of crude oil production in the reservoir.

Understanding the characteristics of an oil-water flow, such as flow patterns and separation characteristics, is very important for efficient and safe production. Due to the complex nature of oil-water flows, an accurate prediction of the flow is challenging. This is due to multiple factors such as operating conditions, pipe inclination, pressure drop, geometrical parameters and physical properties such as interfacial tension, density and viscosity affect the flow (Ahmed and John, 2018; Kamp et al., 2017).

Characterization of oil-water pipe flow consists of identification of flow pattern, flow pattern transitions, pressure drop, phase inversion point, droplet formation, droplet size(s) distribution and viscosity estimations. The most important parameters in the petroleum industry for a multiphase flow include: 1) Pressure drop at varying flow rates; 2) water/oil/gas hold-up or accumulation and 3) thermal characteristics (Urdahl et al., 1997).

Extensive research has been done on two-phase oil-water pipe flow with regard to flow patterns, and several flow regimes have been identified by multiple independent experimental studies (Amundsen, 2011; Angeli and Hewitt, 2000; Barnea and Taitel, 1992; Charles et al., 1961; Edomwonyi-Otu and Angeli, 2015; Elseth, 2001; Kumara et al., 2010b; Lovick and Angeli, 2004; Lum et al., 2006; Panagiota Angeli, 1996; Rodriguez and Oliemans, 2006; Trallero et al., 1997; Yang et al., 2021). Research shows an increasing amount of understanding for mechanisms like flow regime patterns and transitions, flow properties such as pressure drop and mixture viscosity. However, local hydrodynamic flow properties like velocity and turbulence profiles have received less focus. These parameters are extremely important in the development of numerical models and should receive an increased focus (Kumara, 2010). Additionally, there is still a lack of understanding with regards to the phase inversion phenomenon and its effects on the pressure drop and holdup.

Several reviews have been conducted on the intricacies of liquid-liquid flow systems. For instance, Brauner (2003) focused on the flow and pipe configurations of an oil-water flow. The review included experiments, models and formulations for liquid-liquid flow phenomena. Xu (2007) aimed at giving a brief review on the research of oil-water pipe flows in the past decade. His review had three focus areas: 1) flow pattern indication and its transition; 2) phase inversion modelling and 3) pressure drop prediction. His review concluded that the main difficulty with understanding oil-water flows is the existence of the interface. Ismail et al. (2015) reviewed the current state of research on oil-water flows highlighting the need for further research compared to gas-liquid flows. Ismail et al. (2015)

also discussed the complex interfacial chemistry. Ahmed and John (2018) discussed the use of numerical modelling for predicting oil-water flows as well as reviewing recent research. These reviews highlight the vast amount of research done on oil-water flows but also indicate that the following areas require further studies: 1) investigation of the effect of less dense fluid phase on the development of turbulence of the denser fluid phase and vice versa; 2) improves correlation to predict heat transfer coefficients; 3) effect of temperature on the evolution of flow patterns and 4) numerical simulations should be applied in the investigation of oil-water flows.

In recent years, the use of computational and numerical models to solve complex flow problems has increased. Recent technological advances have made computational tools more readily available, making numerical models a powerful tool for analyzing and predicting advanced flow problems. This review, based on technological advancements and previous reviews, will focus on the recent advances in numerical modelling of oil-water flows.

2.2 Physics of two-phase oil-water pipe flow

2.2.1 Oil-water flow terminology

In oil-water modelling there are several basic definitions that are frequently used when discussing oil-water flows. In a pipe with oil-water flow the input volumetric flow rates are Q_o and Q_w, respectively. The input volumetric fractions are given by:

$$\lambda_w = \frac{Q_w}{Q_w+Q_o} \qquad \lambda_o = \frac{Q_o}{Q_w+Q_o} \tag{2.1}$$

λ_w is often referred to as the inlet water cut in oil-water flows. Based on the input flow rates and cross-sectional area of the pipe, the superficial velocity of the oil and water phase is given by:

$$U_{sw} = \frac{Q_w}{A} \qquad U_{so} = \frac{Q_o}{A} \tag{2.2}$$

The relationship between superficial velocities and input phase fractions is given by combing Equation 1.1 and 2.2:

$$\frac{U_{so}}{U_{sw}} = \frac{\lambda_o}{\lambda_w} \tag{2.3}$$

In-situ velocity, which is the actual velocity for each phase in the pipe, is calculated by the input volumetric flow rate divided by the cross-sectional area the phase occupies. Figure 2.1 illustrate the cross-sectional area of the pipe. This means it's different to the superficial velocity which uses the entire cross-sectional area in the calculation. Therefore, the actual velocities are given by:

$$U_w = \frac{Q_w}{A_w} \qquad U_o = \frac{Q_o}{A_o} \tag{2.4}$$

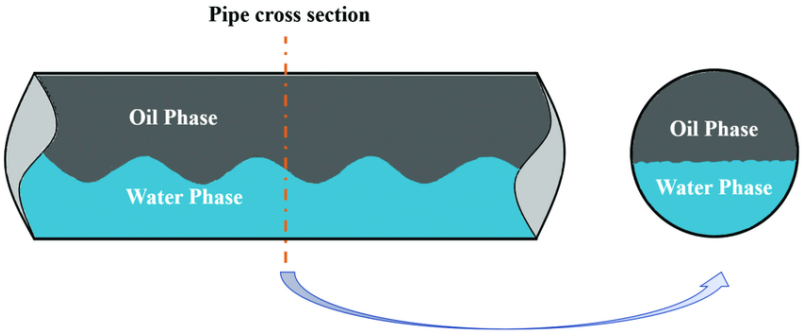

Figure 2.1 - Pipe cross-sectional area for an oil-water flow (Kang et al. 2021).

In an oil-water flow the two phases will most likely not have the same average in-situ velocities. This means that the in-situ volume fraction is different to the input phase fraction and this new volume fraction is given by:

$$\eta_w = \frac{A_w}{A} \qquad \eta_o = \frac{A_o}{A} \tag{2.5}$$

The η_w term, which describes the in-situ volume fractions, is more commonly referred to as the water hold-up in oil-water systems. The ratio between the two in-situ velocities will give the systems slip ratio, S, and it is calculated as follows:

$$S = \frac{U_o}{U_w} = \frac{A_w}{A_o}\frac{U_{so}}{U_{sw}} \tag{2.6}$$

The slip ratio provides the information about which phase travels fastest in the given system. The slip ratio is dependent on several physical properties alongside the flow rates, flow pattern and pipe geometry. A slip ratio above 1 means that the oil travels faster than the water which means S<1 is the opposite scenario.

2.2.2 Flow regimes

Flow regimes, or flow pattern, describes the flow structure, or the distribution of one fluid phase relative to the other. In two-phase oil-water pipe flow the interaction between the two fluids leads to the formation of different flow patterns. Which pattern is formed is dependent on several physical properties such as input water fraction, superficial velocities, liquid densities etc. (Kumara et al., 2010b). The classification of the different flow patterns found in oil-water pipe flow does not have a uniform nomenclature, nor are all the experimental studies performed with the same conditions. Hence, there may exist several flow patterns with different names, but with a similar flow structure. Trallero et al. (1997) classified three sets of patterns with two subsets as seen in Figure 2.2: 1) Segregated flow (stratified flow and stratified flow with mixing interface); 2) Water dominated dispersed flow (Oil in water and water (O/W&W) and Oil in water emulsion (O/W)) and 3) Oil dominated dispersed flow (water in oil and oil in water (W/O & O/W) and water in oil emulsion (W/O)).

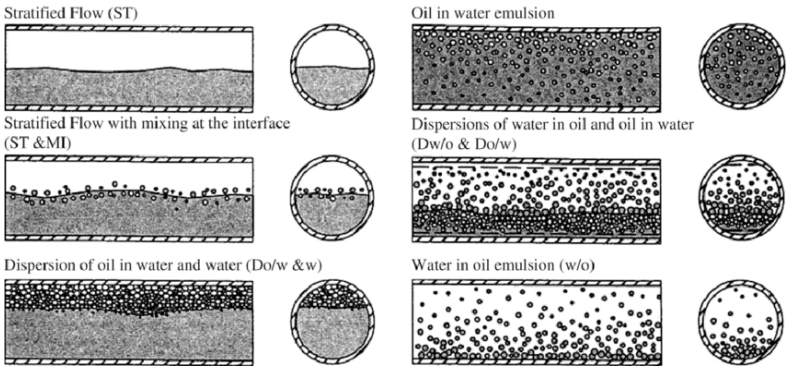

Figure 2.2 - Horizontal oil-water flow patterns (Trallero et al., 1997)

Based on all the previously mentioned experimental studies done on the matter, the flow pattern can be split into two main categories: Segregated/stratified flow and dispersed flow. A short introduction to the two flow regimes follows.

2.2.2.1 Segregated flow

From an experimental study performed by Kumara et al. (2009), three segregated flow patterns for oil-water flows were identified: 1) stratified smooth flow (ST); 2) stratified wavy flow (SW) and 3) stratified flow with mixing interface (ST&MI). These observed patterns were in agreement with the pattern classification proposed by Lum et al. (2006), Rodriguez and Oliemans (2006) and Trallero et al. (1997). The three patterns are shown in Figure 2.3.

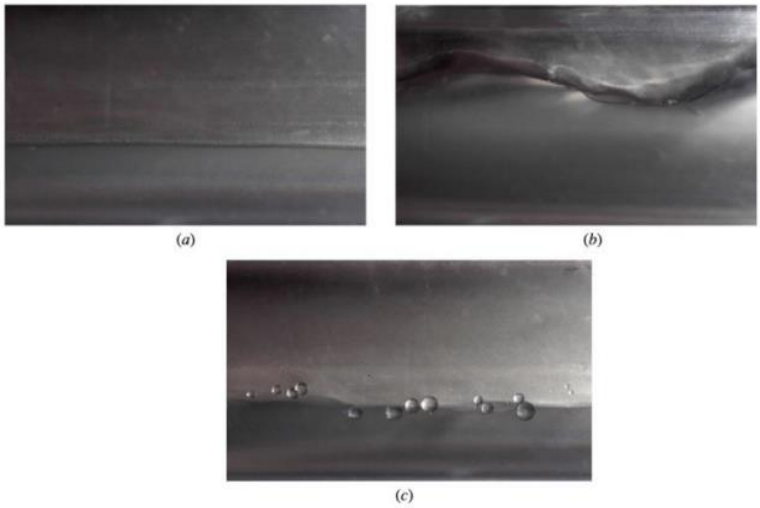

Figure 2.3 - Observed segregated oil-water flow: (a) stratified smooth (ST), (b) stratified wavy (SW), (c) stratified flow with mixing at the interface (ST&MI)(Kumara et al., 2009)

Segregated flow (Figure 2.3 (a)) is observed at low superficial oil and water velocities, where the flow is gravity dominated. With further increases in mixing velocities, the pattern shifts towards a wavy structure (Figure 2.3 (b)). As the velocity continues to increase, water droplets begin to form in the oil phase and oil droplets begin to form in the water phase. These droplets exist along the interfacial waves, as seen in Figure 2.3 (c).

2.2.2.2 Dispersed flow

Every flow pattern that does not fall under the segregated classification represents different dispersions. “Dispersions are always formed when the motion of oil-water flow is sufficiently intense” (Kumara 2010). Which of the two liquids is dominant largely depends on the superficial velocities of the liquids. Thus, the following two aspects can be noted for oil-water flow: 1) The fluid phase with the higher momentum entrains the fluid with the lower momentum and 2) the less dense fluid phase follows the Reynolds transition criterion and attain instability at Re ~ 2000. Meanwhile, the denser fluid phase become unstable at much higher values of Re ~ 18000, indicating the incompatibility of the Reynolds criterion (Ahmed and John, 2018).

Experimental studies conducted by Elseth (2001), Hapanowicz (2010), Ibarra et al. (2015) and Kumara et al. (2009), indicate that the flow turbulence is influenced by the interface and the non-compatibility of Reynolds number criterion. Kumara et al. (2009) observed four different dispersion flow patterns which are presented in Figure 2.4.

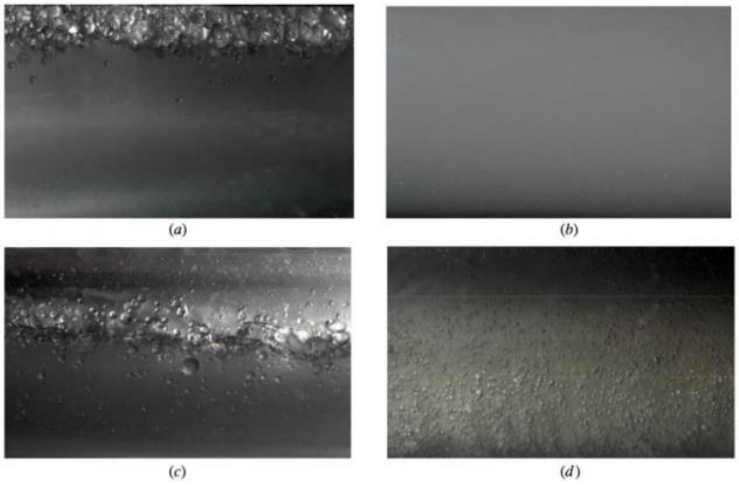

Figure 2.4 - Observed dispersed oil-water flow: (a) dispersion of oil in water (Do/w&w), (b) dispersion of oil in water (Do/w), (c) dispersion of oil in water and water in oil (Do/w&w/o), (d) dispersion of water in oil (Dw/O)(Kumara et al., 2009).

2.2.3 Phase inversion

Phase inversion in oil-water emulsion systems refer to the phenomenon in which, a small change in the operational conditions causes the dispersion of oil droplets in water (Do/w) to change to dispersion of water droplets in oil (Dw/o), or vice versa (Kumara, 2010). This transition phase typically occurs with changes, often abrupt, in heat and mass transfer between the two phases. Since the rheological characteristics of the dispersion and the associated pressure drop change abruptly and significantly at, or near the Phase Inversion Point (PIP), the PIP is a crucial factor to consider when designing oil–water transportation pipelines (Luo et al., 2022; Xu, 2007).

Luo et al. (2022) outlined the importance of how the water cut influences the PIP. The study demonstrated that with a low water cut, crude oils can emulsify all the water to form a stable W/O emulsion. However, when the water cut of a system exceeds a certain critical value, the

crude oils lose the ability to emulsify all water; instead, they are enveloped by a water phase and form unstable O/W emulsions. Luo et al. (2022) found that the critical water cut of a specific system corresponds to an abrupt change in the apparent viscosity of the emulsion. This marks the PIP for the emulsion, transitioning from W/O to O/W. Additionally, two correlations were identified:

1) The apparent viscosity of a stable W/O emulsion decreases with increased shear rate and temperature and increases with a higher water cut.
2) The apparent viscosity of an unstable O/W emulsion decreases with increased shear rate, temperature and water cut.

These correlations are illustrated in Figure 2.5.

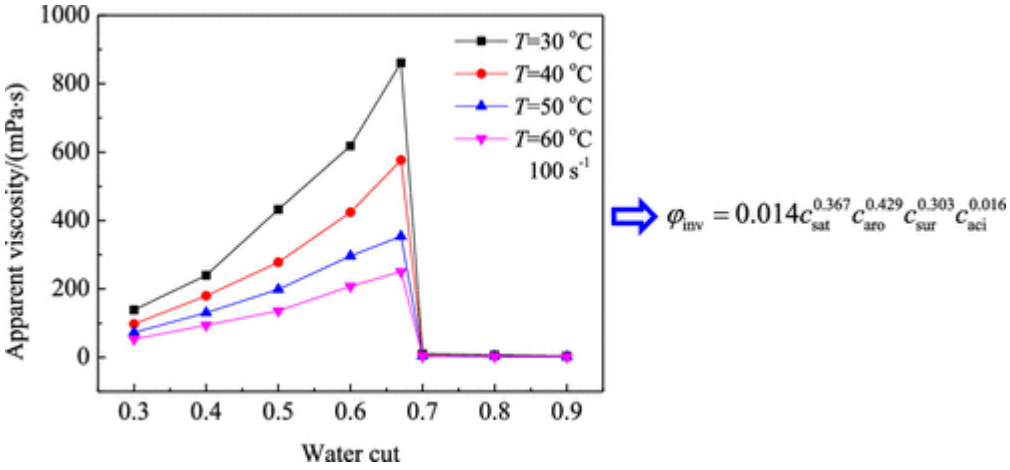

Figure 2.5 - Changes in apparent viscosity (Luo et al. 2022).

2.3 Modelling of two-phase oil-water flow

Correctly predicting two-phase oil-water flow is a highly complex process and currently there exists no exact analytical solution. In the past, oil-water flows have mainly been studied through experiments and have been very empirically oriented. With the technological progress in the past few decades, the use of numerical techniques to solve the complex two-phase flow has increased. This chapter will outline the three prediction methods for two-phase oil-water flows which are: empirical models, analytical models and numerical models.

2.3.1 Empirical models

Empirical models for two-phase flow often draw from the analogous ones for gas-liquid flows and are then adopted for liquid-liquid flows. These models rely on a rich variety of correlations.

Govier and Omer (1962) compared a vast number of existing empirical correlations. The investigated correlations suggested accurate prediction of pressure drops in oil-water flow. However, it was reported that the number of restrictions in the models made them difficult to apply and adapt to certain conditions. The correlations were limited to specific flow patterns and assumed a static pipe diameter and static gas density.

Beggs and Brill (1973) were the first to predict multiphase flow behavior at all inclination angles in a pipe using an empirical model. They conducted extensive testing at various pipe inclinations ranging from 0° to 90°. The test fluids used were air and water. A correlation for the pressure-gradient was developed based on their experiments:

$$\frac{dp}{dL} = \frac{\left(\frac{f\rho_n \upsilon_m^2}{2d} + \rho_s g sin\theta\right)}{1 - E_k} \tag{2.7}$$

Here, E_k is given by Equation 2.8:

$$E_k = \frac{\upsilon_m \upsilon_{Sg} \rho_n}{p}\left(\frac{dp}{dZ}\right) \tag{2.8}$$

And

$$\rho_s = \rho_{L(\theta)} + \rho_g[1 - H_{L(\theta)}] \tag{2.9}$$

The pressure-gradient correlation is inaccurate when applied to an oil-water flow. However, Beggs and Brill (1973) correlated the transition boundaries for the different flow patterns with no-slip liquid holdup and mixture Froude number, which is shown as:

$$N_{Fr} = \frac{\upsilon_m^2}{gd} \tag{2.10}$$

Given a horizontal pipe, the following inequalities could be used to determine which flow patterns exist in an oil-water flow:

Segregated:

$\lambda_L < 0.01$ and $N_{Fr} < L_1$ or $\lambda_1 \geq 0.01$ and $N_{Fr} < L_2$ (2.11)

Transition:

$\lambda_L \geq 0.01$ and $L_2 < N_{Fr} \leq L_3$ (2.12)

Intermittent:

$0.01 \leq \lambda_L < 0.4$ and $L_3 < N_{Fr} \leq L_1$ or $\lambda_L \geq 0.4$ and $L_3 < N_{Fr} \leq L_4$ (2.13)

Distributed:

$\lambda_L < 0.4$ and $N_{Fr} \geq L_1$ or $\lambda_L \geq 0.4$ and $N_{Fr} > L_4$ (2.14)

where λ_L is the non-slip holdup factor, L_1, L_2, L_3, L_4 are the correlation variables and N_{Fr} is the Froude number.

Al-Wahaibi (2012) developed a correlation for the pressure gradient in horizontal oil-water pipe flows. The correlation is based on the work of Angeli and Hewitt (1999). The correlation for the developed pressure gradient is as follows:

$$\frac{dp}{dx} = 2.4\left(\frac{(f_{cor}\rho_m U_m^2)}{2D}\right)^{0.8} \tag{2.15}$$

Where 2.4 is a dimensional coefficient fitting parameter, ρ_m is the mixture density, U_m is the mixture velocity and D is the pipe diameter. The model was validated against 11 separate experimental data sets which can be seen in Figure 2.6.

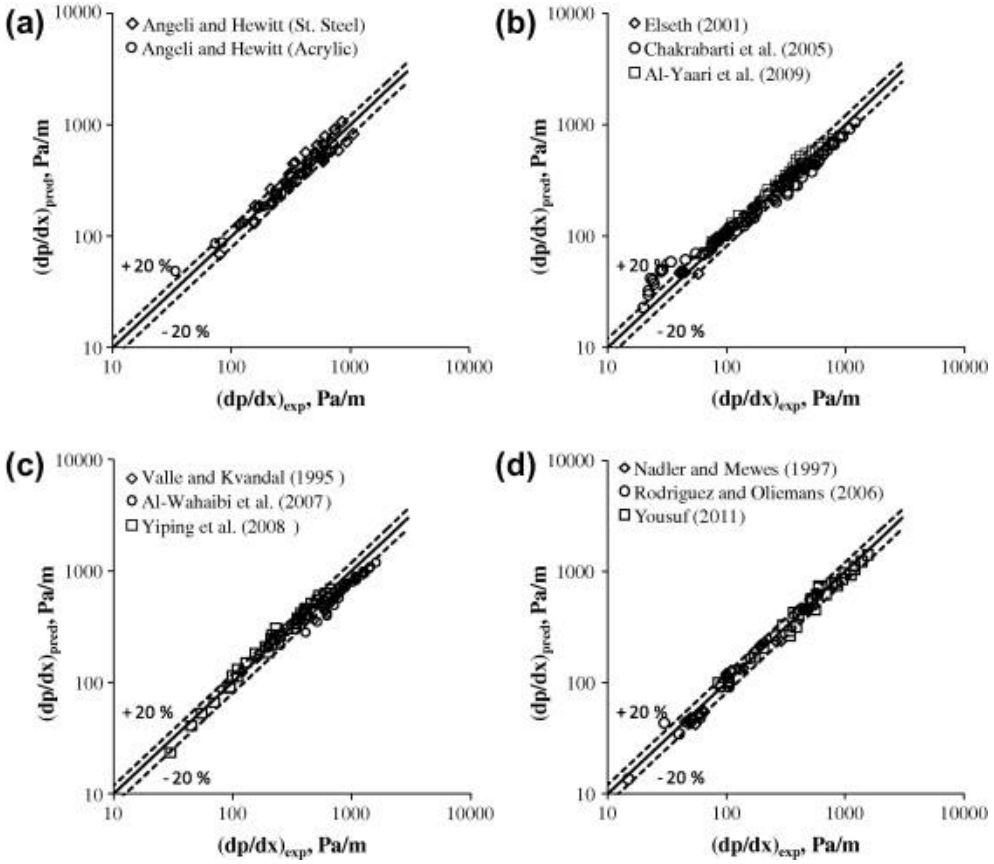

Figure 2.6 - Evaluation of the proposed correlation against experimental data (Al-Wahaibi, 2012).

The developed correlation was in good agreement with the experimental data. The average absolute error ranged from 2.65% at the lowest, to 20.78% at the highest. The empirical model showed a higher accuracy than the two-fluid model used by Angeli and Hewitt (1999).

Abubakar et al. (2016) continued the work of Al-Wahaibi (2012) and developed a new friction factor correlation for oil-water flow which included drag-reducing polymers (DRP). The study aimed to predict the pressure gradients of an oil-water flow after the addition of the drag-reducing polymers. The friction factor correlation was found by plotting the calculated DRP mixture friction factor against their corresponding mixture Reynolds number as seen in figure 2.7.

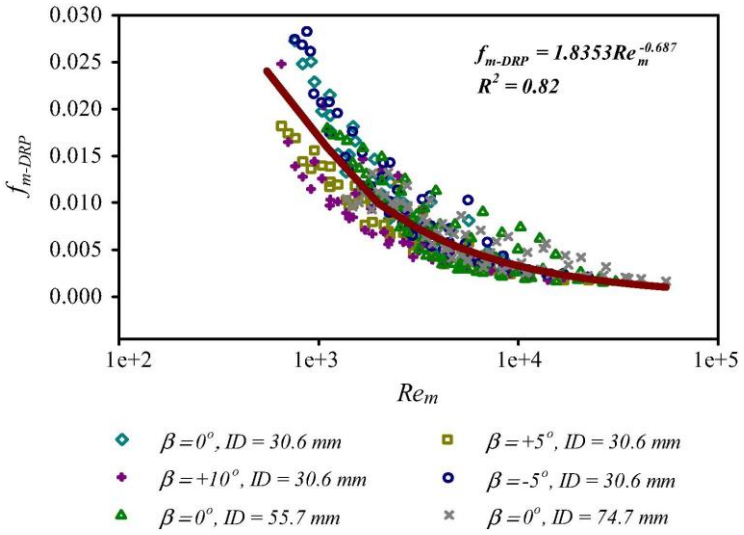

Figure 2.7 – Fanning friction factor variation with the mixture Reynolds number for different pipe inclinations and different pipe diameters (Abubakar et al., 2016).

The correlation was then tested against experimental data. Figure 2.8 shows the results of applying the developed correlation against experimental data. The correlation offered satisfactory predictions and Abubakar et al. (2016) argued it showed better performance than the models it was compared to.

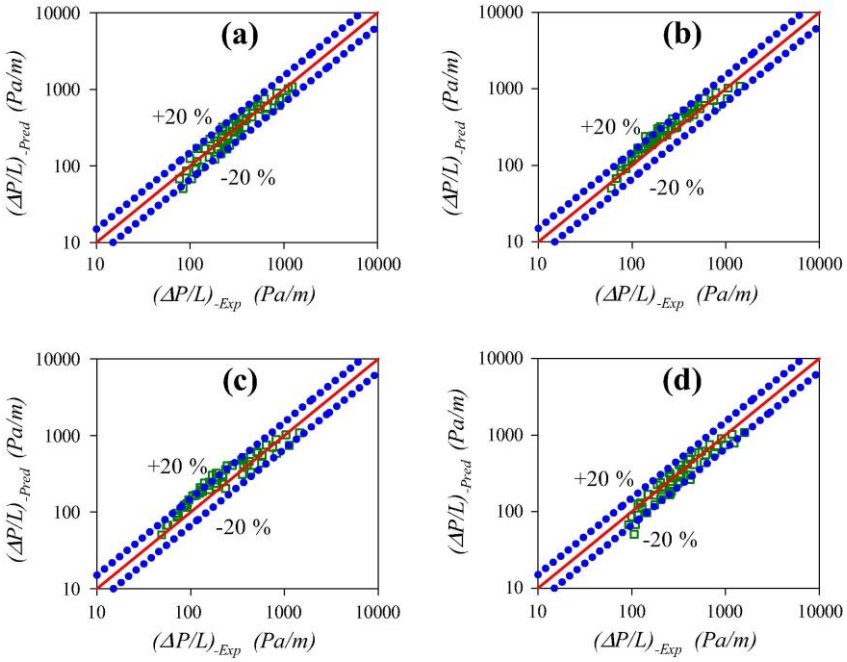

Figure 2.8 - Predicted pressure gradients using the developed correlation against experimental pressure gradients (Abubakar et al., 2016).

2.3.2 Analytical models

Gada and Sharma (2012) developed a predictive analytical solution for fully developed stratified oil-water flows in an inclined plane-channel. The model was validated through a numerical study applying a level set method (LSM)-based Navier-Stokes solver. The performance of the analytical solution was shown to be excellent when compared to the numerical study. Additionally, a physical model was created to reproduce the results from the analytical and numerical models. The model is depicted in Figure 2.9.

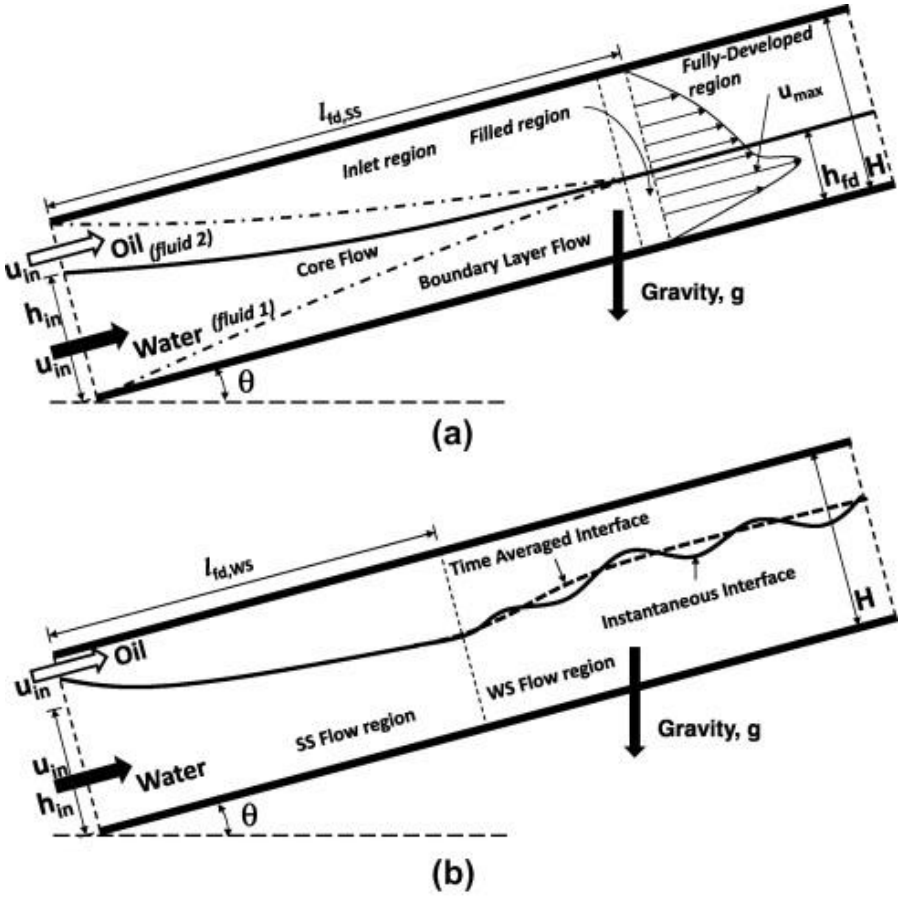

Figure 2.9 Physical model for developing flow in an inclined plane channel (a) Stratified Smooth and (b) Wavy Stratified flow. Channel wall, interface and boundary layers are represented by thick-solid, solid and dash-dot lines, respectively. For WS flow, time-averaged interface is represented by dashed line (Gada and Sharma, 2012).

The created model follows a set of assumptions:

- Fully developed stratified flow
- Density matched viscosity
- Incompressible Poiseuille flow
- No-slip at wall/interface.
- Continuity of velocity and shear-stress at the interface

Based on the given assumptions and performing the following substitutions: $\eta=1$ (property ratio), $H_{fd}=0,5$ (interfacial height), $Q_1^*= Q_2^*= 0,5$, the analytical model proposed by Gada and Sharma (2012) degenerates to the single fluid flow solution shown in Equation 2.16:

$$U_1 = U_2 = 6Y(1-Y) \qquad Y_{Umax} = 0{,}5 \qquad \frac{\Delta \Pi}{\Delta x} = -12$$
$$PO = 12(1-2Y) \qquad P_{O,B} = -P_{O,T} = 12 \qquad U_{max} = 1.5 \tag{2.16}$$

Here, U_1 and U_2 is the velocity profiles, Y_U is the coordinate for the maximum U-velocity, Π is the non-dimensional pressure, PO, $P_{O,B}$, $P_{O,T}$ is the Poiseuille number for the pipe, bottom wall, and top wall, respectively.

Hibiki and Rassame (2019) performed an analytical study aimed at creating a predictive model for the oil fraction in an oil-water flow in a horizontal pipe, applicable to all flow patterns. The proposed model by Hibiki and Rassame (2019) was validated through comparison with existing experimental data obtained under varying test conditions. A total of thirteen experimental datasets were used in the validation process to ensure the model was tested under varying conditions.

Four assumptions were made to derive the oil fraction analytically (Hibiki and Rassame, 2019):

a) Stratified flow with two homogeneous mixture phases in the upper and lower parts.
b) Equal velocity head between the upper and lower homogeneous mixture phases (equal velocity head model).
c) Homogeneous mixtures of liquid 1 and liquid 2 for both parts being considered as single-phase fluids with mixture densities.
d) Thermal equilibrium condition.

The proposed model for predicting the oil fraction is a function of flow quality, density ratio, and includes two entrainment constants, e_1 and e_2 (droplet entrainments for oil phase and water phase). The oil fraction is defined as:

$$\alpha_{oil} = \frac{A_{oil}}{At} = \frac{(A_{1c}+A_{2d})}{A} = \frac{W_2(1-e_2)}{\rho_2 A_{v_{m_2}}} = \frac{W_2 e_2}{\rho_2 A_{v_{m_1}}} \tag{2.17}$$

Where W_2, e_2, v_{m1}, v_{m2} and ρ_2 are the oil mass flow rate, entrainment factor upper part, mixture velocity upper part, mixture velocity lower part and density of oil respectively. The proposed analytical equation for predicting the oil fraction is as follows:

$$\alpha_{oil} = \left\{ \left[1 - e_2 + \left(\left(\frac{1-x}{x} \right) \left(\frac{\rho_2}{\rho_1} \right) \right) e_1 \right] \left[(1-e_2) + e_2 \left(\frac{\rho_{m2}}{\rho_{m1}} \right)^{-\frac{1}{2}} \right]^{-1} + \left[e_2 + \left(\frac{1-x}{x} \right) \left(\frac{\rho_2}{\rho_1} \right) (1 - e_1) \right] \left[e_2 + \left(\frac{\rho_{m2}}{\rho_{m1}} \right)^{\frac{1}{2}} (1-e_2) \right]^{-1} \right\}^{-1} \tag{2.18}$$

Where ρ_{m2} and ρ_{m1} is the density of the upper and lower part of the pipe respectively.

The results from the model validation are presented in Figure 2.10. The validation process demonstrated that the model predictions are in good agreement with the experimental data with an accuracy of $\pm 15\%$.

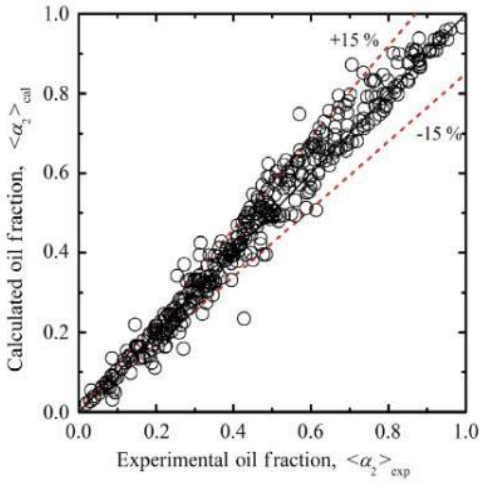

Figure 2.10 - Comparison of total calculated oil fraction and experimental oil fraction for horizontal oil-water flows used in the model validation (Hibiki and Rassame, 2019).

The analytical model proposed by Hibiki and Rassame (2019) also demonstrated that it can be applied to all six oil-water flow regimes identified by Trallero et al. (1997) in Figure 2.2.

2.3.3 CFD and Numerical models

Walvekar et al. (2009) utilized the CFD software ANSYS FLUENT to simulate a 3D horizontal pipe flow with an oil-water system. They simulated a turbulent dispersed flow using a Eulerian-Eulerian multiphase model along with the standard k-ε turbulence model. Due to the nature of the turbulence model selected, the authors observed good results at high mixture velocities and discrepancies at low mixture velocities. This discrepancy comes from the k-ε turbulence model only being valid for fully turbulent flows, and therefore loses accuracy at lower Reynolds-number flows.

Similarly, Burlutskii (2018) also utilized ANSYS FLUENT to simulate a dispersed oil-water system with the k-ε turbulence model. Burlutskii (2018) used the Euler-Lagrange scheme to resolve the interaction between the two phases in a vertical pipe. As with Walvekar et al. (2009), the results exhibited good agreement at high mixture velocities as shown in Figure 2.11.

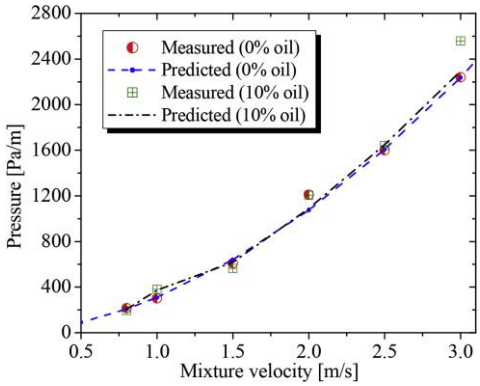

Figure 2.11- Measured and predicted pressure drop as a function of mixture flow velocity (Burlutskii, 2018).

Additionally, Burlutskii (2018) discovered that for highly turbulent liquid-liquid dispersed pipe flows, the shear-lift force holds significantly higher importance for adequately flow representation than detailed modelling of the break-up/coalescence phenomena.

When performing oil-water simulations, selecting the correct turbulence model for the system is crucial for an accurate result. Shi et al. (2017) performed a CFD study on a horizontal oil-water flow system with matched density and medium viscosity ratio (=18.8), involving several different flow regimes (core annular flow, oil plugs/bubbles in water and dispersed flow). In these flow regimes, surface tension and wall contact angle play a pivotal roles in the calculations. Shi et al. (2017) showed that the volume of fluid (VOF) model, in conjunction with the SST k-ω turbulence model and turbulence damping activated, can predict the flow structures of core annular flow and oil plugs/bubbles in water. The turbulence damping scheme added the following additional source terms to the ω-equation, which reduces the destruction term:

$$S_\omega = A_i \Delta n \beta \rho_i \omega_\omega^2 \tag{2.19}$$

$$\omega_\omega = B \frac{6\mu_i}{\beta \rho_i \Delta n^2} \tag{2.20}$$

$$A_i = 2\alpha_i |\nabla \alpha_i| \tag{2.21}$$

Where A_i represents an interface area density that activates the correction term in the vicinity of the interface, α_i is the volume fraction of phase *i*, Δn is the grid size in the interface region, β is a closure coefficient and B is an adjustable damping factor. The turbulence damping function increased the accuracy of the results by 5%. However, the authors also highlighted the shortcomings of using the VOF model for dispersed flows as the interface length scales tend to become smaller than the computational grid sizes. This was emphasized further by Chen et al. (2023), who performed CFD simulations for a dispersed oil-water flow. For dispersed flows, a Eulerian approach is best suited as this method treats each phase as a continuous fluid-a two-fluid model-with separate volume fractions for each phase. Chen et al. (2023) used the OpenFOAM solver multiphaseEulerFoam, which applies the Eulerian method coupled with population balance models and the mixture k-ε turbulence model. MultiphaseEulerFoam also implements a blending factor that captures the retardation of droplet rising and coalescing in the dense packed layer (DPL) of the flow. The study's results showed that there are two main challenges for dispersed flow simulations:

1) Modelling the turbulent dispersion force is highly system dependent.
2) Phase inversion and water release rates in the densely packed layer require further research to improve existing drag and coalescence models.

These conclusions are consistent with other research, as Pouraria et al. (2021) reported similar troubles for dispersed flows.

When dealing with dispersed flows a two fluid model must be used to achieve the desired results with realistic accuracies. For segregated or stratified flows, a different modelling approach can be employed. For a system with a stratified flow the VOF approach can be applied as demonstrated by the studies of Gao et al. (2003), Kang et al. (2021), Al-Yaari and Abu-Sharkh (2015), Pouraria et al. (2021), Sunday et al. (2023) and Kumara (2010).

Kang et al. (2021) performed a 2D numerical study where they compared three different turbulence models against the experimental work of Kumara (2010). In a bipolar coordinate

system, the k-ε, k-ω and SST k-ω turbulence models were implemented, and the solution strategy depicted in Figure 2.12 was utilized.

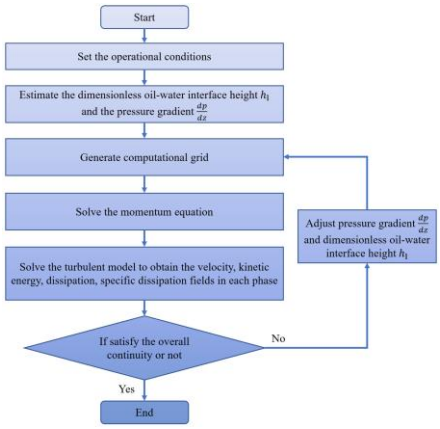

Figure 2.12 - Numerical solution strategy (Kang et al., 2021).

Kang et al. (2021) reached the same conclusion as in similar studies performed by Archibong-Eso et al. (2019) and Shi et al. (2017), where the SST k-ω turbulence model showed the highest accuracy for an oil-water system. Although the SST k-ω turbulence model provided the best results compared to the two other models, it exhibited inaccuracies when compared to the experimental data regarding the pressure gradient, as seen in Figure 2.13. The model showcased low accuracy at lower superficial velocities, but when the superficial velocities of the two phases were relatively close, the accuracy improved, and the deviation decreased below 15%.

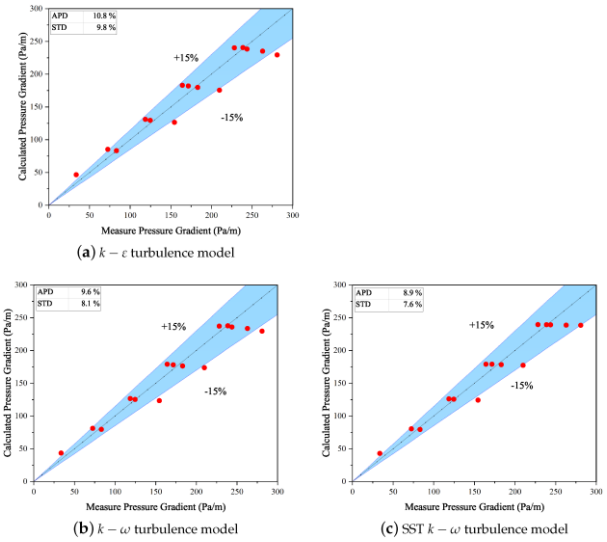

Figure 2.13 - Pressure gradient comparison for simulated results and experimental results (Kang et al., 2021)

For the calculated mean axial velocity, all three turbulence models failed to predict the correct profile. As seen in Figure 2.14, the models were unable to predict the asymmetric profile around the oil-water interface and overpredicted the velocity compared to the experimental data.

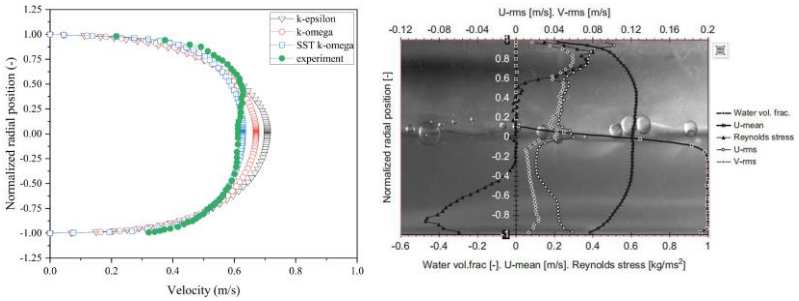

Figure 2.14 - Mean axial velocity for Um=0,61 m/s and a water cut of 0.5 (Kang et al., 2021, Kumara 2010).

The inaccuracy around the oil-water interface becomes very clear when the water cut is lowered to 0.25 as seen in Figure 2.15.

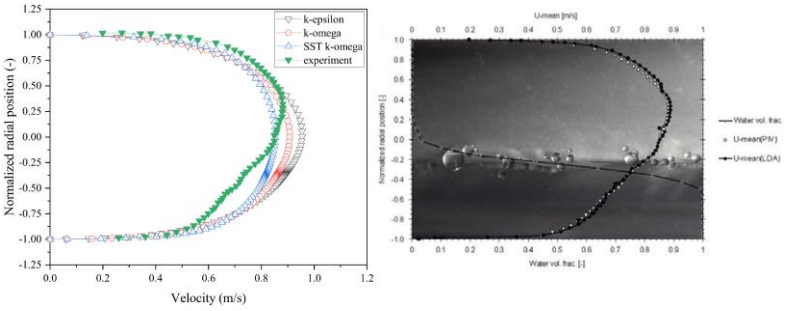

Figure 2.15 - Mean axial velocity for Um=0,68 m/s and a water cut of 0.25 (Kang et al., 2021, Kumara 2010).

The discrepancy between the velocity profiles could be attributed to several factors:

1) The numerical model is based on a one-equation model for both phases, meaning one set of momentum equations is applied across the entire system. This is likely the reason that the velocity curve is symmetrical and insensitive to the changes at the interface.
2) No turbulence damping or interface treatment has been implemented in the numerical model to handle the complex physics around the oil-water interface.
3) The shape of the oil-water interface is not considered.

Kumara (2010) encountered a similar issue with the accuracy of mean axial velocity profile in his CFD simulations, which were compared with his experimental study. A VOF based solver in ANSYS FLUENT was used and a Piecewise-linear interface calculation (PLIC) scheme was applied for the reconstruction of the interface. This approach did not manage to capture the asymmetrical velocity profile.

These observations underscore that a numerical model using a single momentum equation, and a VOF based solver for a multiphase flow is inadequate without the incorporation of additional turbulence and interface treatment. To increase numerical accuracy, special considerations, such as turbulence damping and flow physics, at the oil-water interface are necessary.

These considerations were emphasized in the studies by Edomwonyi-Otu and Angeli (2015) and Santos et al. (2019), which had a larger focus on the structure of the oil-water interface. Reports from these studies indicate that in a two-phase oil-water system the interface takes a concave shape. Furthermore, Liu et al. (2022) argue that in systems with a low density difference between oil and water, the surface tension becomes increasingly important. The

surface tension between the liquids causes the wetting fluid to climb over the pipe wall, leading to a curved interface as shown in Figure 2.16. This illustrates that special considerations must be made for the interface shape, and that a planar interface assumption will result in poor accuracy.

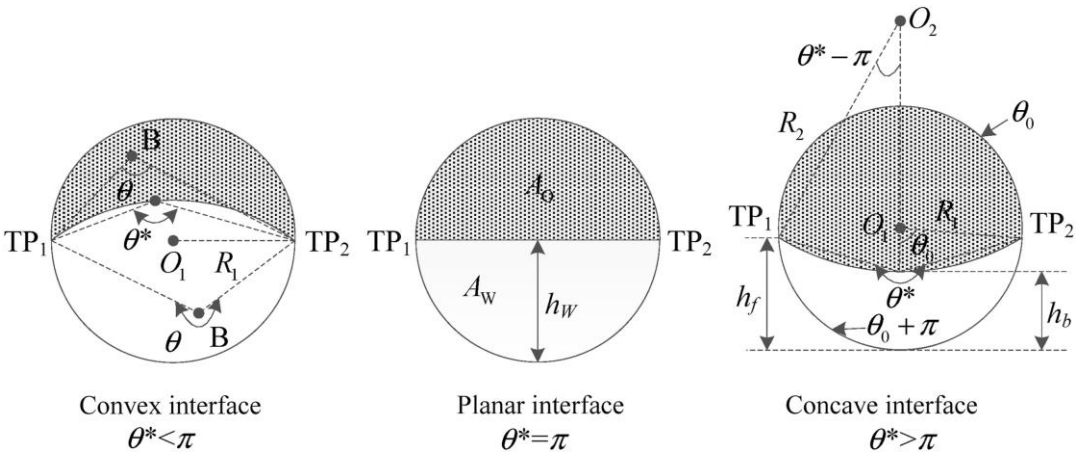

Figure 2.16 - Interface geometry (Liu et al., 2022).

The interface configuration in a stratified oil-water system can be characterized by the Eötvös number, E_O, given by:

$$E_O = \frac{\Delta\rho g R^2}{2\sigma} \tag{2.19}$$

Where $\Delta\rho$ is the density difference between the two liquids, R is the pipe radius, g is the gravitational force and σ is the interfacial tension. The Eötvös number essentially characterizes the shape of the interface between two fluids. As the Eötvös number increases, which indicates either a decrease in surface tension or an increase in density difference, the interface become more planar. Conversely, as the Eötvös number decreases, the more closely the interface approaches a curved surface. For systems with a high Eötvös number, a more standard approach could be employed, whilst for low Eötvös numbers more complex interface capturing method must be used. Liu et al. (2022) demonstrated, through numerical simulations, the existence of a concave interface when E_O<10. The results, as shown in Figure 2.17, show a better agreement with experimental results when assuming a curved interface.

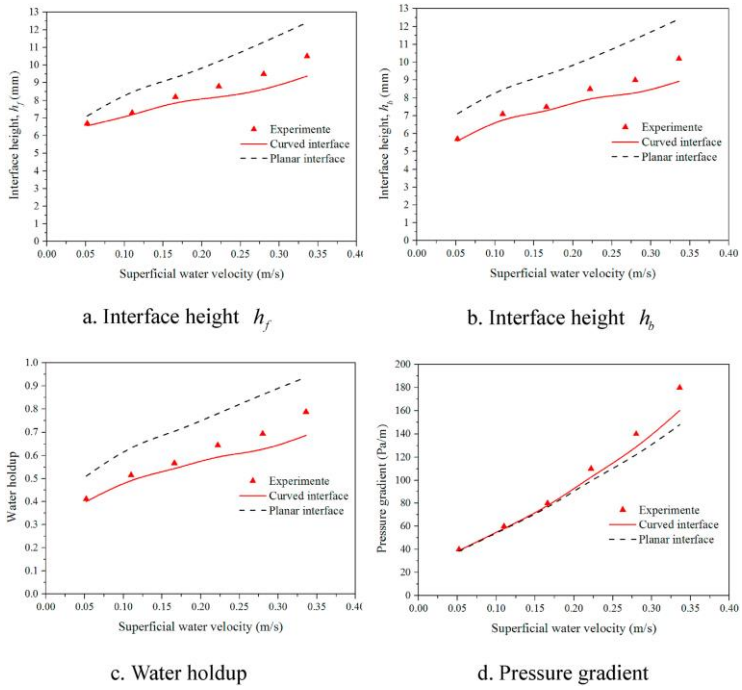

Figure 2.17 – Comparison between calculated and experimental data for interface heights, water holdup and pressure gradient with curved interface and planar interface (Liu et al., 2022).

Gao et al. (2003) carried out a numerical study using the VOF approach, assuming a curved interface. The model implemented a customized turbulence calculation, wherein the RNG k-ε turbulence model in the fully turbulent regions and a low Reynolds number k-ε model at the near-wall regions. Additionally, the Continuum Surface Force (CSF) model by Brackbill et al. (1992) and turbulence damping by Lam and Bremhorst (1982), were applied at the wall. The results were compared to the experimental work of Elseth (2001) and showed promising results. The results are shown in Figure 2.18 and the produced velocity profiles show a good agreement with the experimental data. The model was able to predict an asymmetrical axial velocity profile and show close agreement with the experimental data for 50% and 75% water cuts. Some discrepancies are seen for 25% water cut. The study shows the effectiveness of turbulence damping and using the LRN k-ε model instead of wall functions.

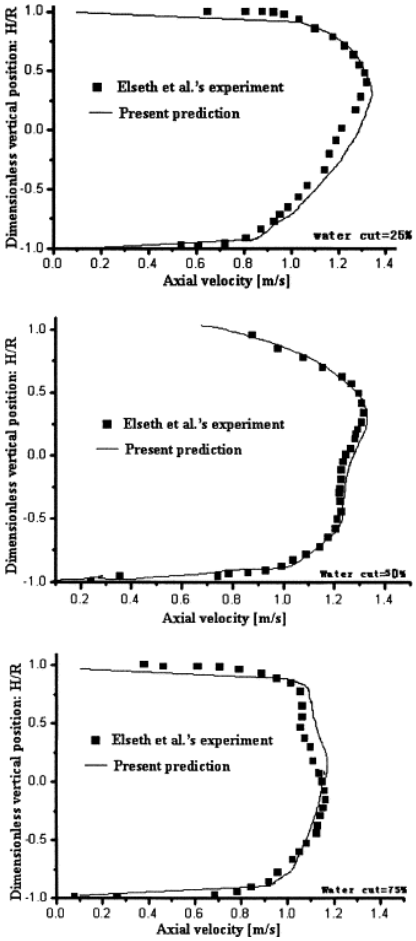

Figure 2.18 - Comparison of predicted and experimental mean axial velocity (Gao et al., 2003).

Liu et al. (2022) employed the same turbulence scheme at the wall as Gao et al. (2003), while applying the standard k-ε model for the turbulent region. Additional interface treatment was applied to the numerical model of an oil-water pipe flow. Liu et al. (2022) established the following conditions for their numerical calculations: $k_{I,O} = k_{I,W} = 0$ and $(\mu_t)_{I,O} = (\mu_t)_{I,W} = 0$, resulting in the following coupling between the two phases at the interface: $u_{I,O} = u_{I,W}$. These conditions were based on the finding of Newton and Behnia (2000), who suggested that the presence of free surfaces reduces turbulence similarly to the presence of a wall. Duan et al. (2015, 2014) proposed, as a preliminary approximation, that the existence of a free surface could be treated as a moving wall. These conditions produced satisfactory results with regards to the mean axial velocity, as seen in Figure 2.19. The simulated profile are compared to the experimental work of Ibarra et al. (2018).

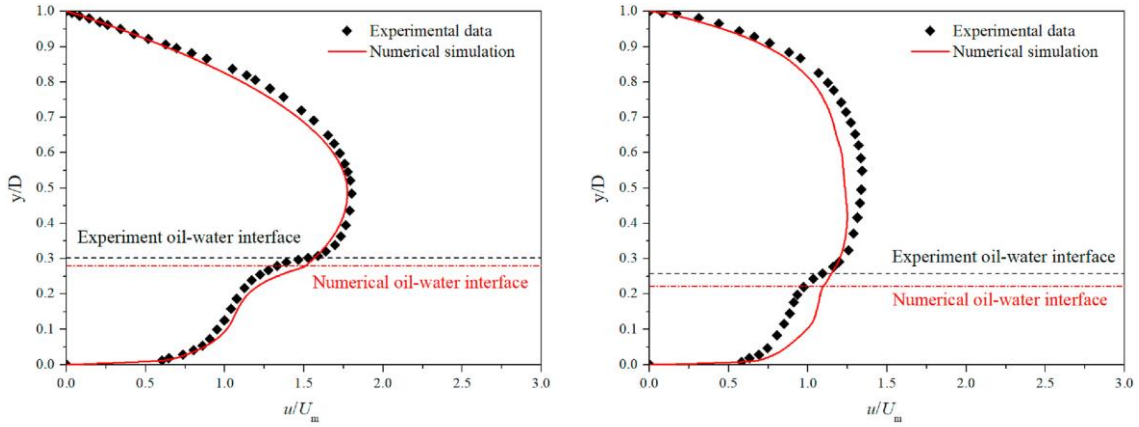

Figure 2.19 - Comparison of numerically and experimental data for the mean axial velocity profiles (Liu et al., 2022).

It is argued that the oil-water interface behaves like a moving wall, causing less flow resistance for the oil phase compared to near the pipe wall. Meanwhile, for the water phase the interface behaves like a faster moving wall, dragging the water flow forward. While the results are interesting, there isn't sufficient data and variation in the flow conditions to fully support the assumptions made. Nonetheless, the findings from Gao et al. (2003) and Liu et al. (2022) provide an intriguing way on how the treatment of the two-phase oil-water interface could be approached.

2.4 Conclusion

This review has attempted to present the recent advances in modelling of oil-water flows. Additionally, it introduces flow terminologies and the physics of a two-phase flow. The studies discussed in this review demonstrate that the complexity of an oil-water flow is still a challenge in numerical modelling.

Although the studies presented did not reach the desired level of accuracy, several important discoveries should be noted. When not applying any special conditions the SST k-ω turbulence model showed clear advantages over the other turbulence models for stratified flows. For dispersed flows where most of the flow region is turbulent, other factors such as mesh size, drag forces, lift etc. influence the results as much as the choice of turbulence model. Another notable discovery is the importance of how the oil-water interface is treated. The studies indicate that special treatment must be applied to the oil-water interface and the turbulence model to achieve more accurate results. While the studies presented show promising results, it remains unclear what specific conditions must be implemented and the how these conditions are applied to each specific system.

3 CFD methodology for oil-water flow

Accurately predicting an oil-water flow is a highly complex process and currently no exact analytical solution exists. Historically, studies on oil-water flows have predominantly relied on physical experiments. Conducting these experiments can become very costly due to the equipment required for non-invasive measurements. CFD studies offer a cost-effective alternative for studying fluid flows as computational power is more easily accessible.

To fully access the capabilities of CFD, a thorough understanding of the technique is required. Each study is unique and demands an approach that is suited for the simulated system. This chapter aims to introduce the CFD methodology, how to approach the modelling aspect and outline the governing equations for an oil-water flow.

3.1 CFD structure

All CFD software structures its interfaces differently, but they are all fundamentally structured on the same principles (Versteeg and Malalasekera, 2007). In Figure 3.1 the three main principles for a CFD software are shown.

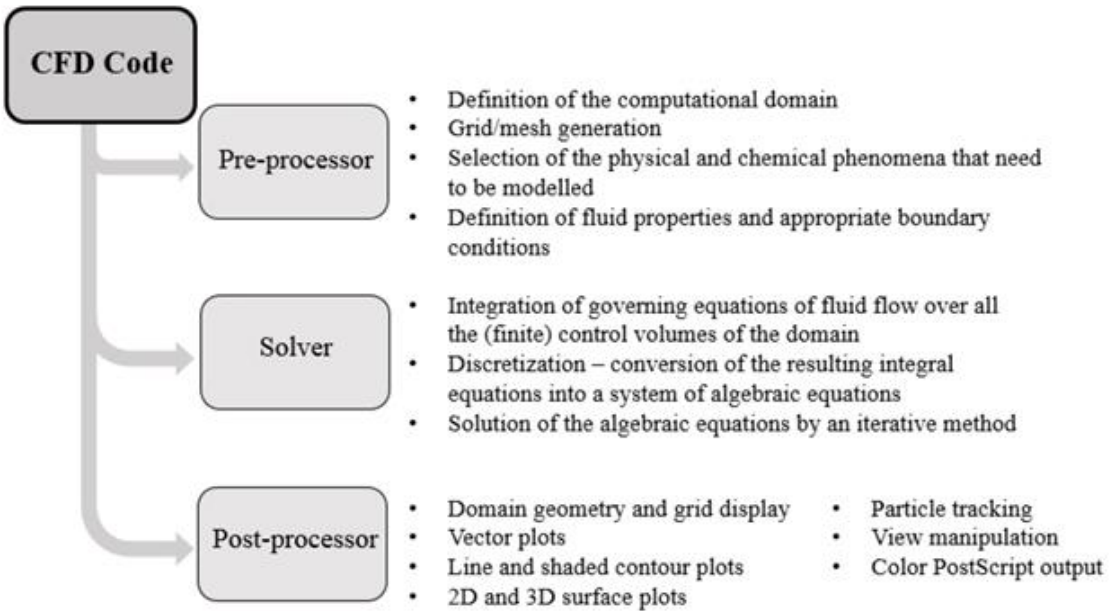

Figure 3.1 – Structure elements for a CFD software (Versteeg and Malalasekera, 2007; Tawekal 2015).

1. Pre-processor:
   The computational domain is defined with a geometry. The generated geometry is then divided into smaller, non-overlapping sub-domains. This is often called a grid, or mesh, of discrete cells (Versteeg and Malalasekera, 2007). A well-structured grid is integral for achieving an accurate solution as all the calculations happen in these cells. Once the computational domain is defined, the physical and chemical properties for

the system must be specified. Based on these properties sensible boundary conditions are chosen.

2. Solver:
   There are three numerical solution techniques when it comes to CFD simulations: 1) the finite difference method (FDM); 2) the finite element method (FEM) and 3) the finite volume method (FVM). Which methods that is applied are different for each system, but the FVM is the most used as this approach is suitable for any type of grid (Versteeg and Malalasekera, 2007). The FVM method is used for this thesis. The FVM method divides the computational domain into a finite number of continuous control volumes (CV) and the conservation equations are applied in the control volumes (Versteeg and Malalasekera, 2007).
3. Post-processor:
   This is where the simulation results are processed and presented. Data visualization tools such as 2D and 3D surface plots and particle tracking are applied at this stage to collect data.

3.2 Mesh generation

The success of any CFD simulation relies on the quality of the mesh (grid). The quality of the mesh affects the convergence, numerical solution and stability of a simulation (Lande, 2021). This means that the mesh generation stage should receive adequate attention because as shown by V. Hernandez-Perez et al. (2010): a high quality mesh will influence the results of the simulation and affect the convergence rate for the chosen solver.

3.2.1 Mesh structure

As illustrated in Figure 3.2, the structure of a mesh can generally be divided into two groups: structured and unstructured grids (Lande, 2021).

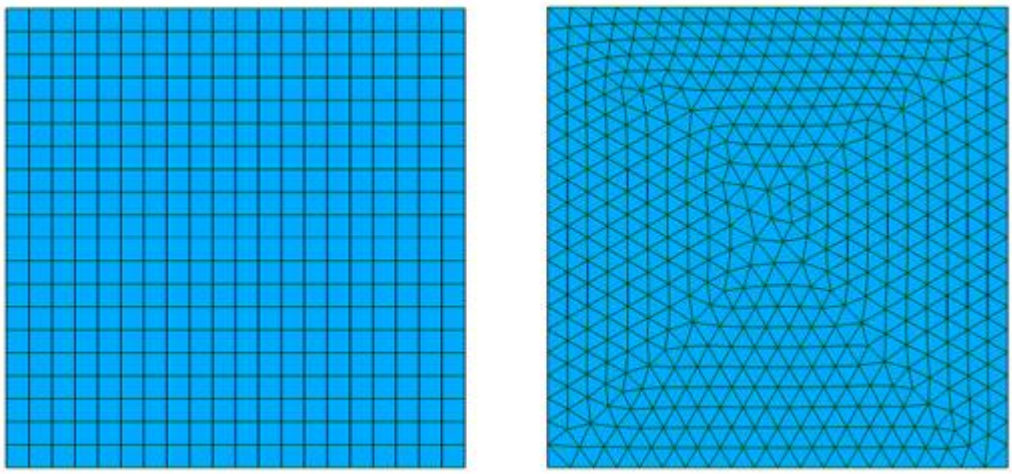

Figure 3.2 - (a) structured mesh and (b) unstructured mesh (Lande, 2021).

The choice of mesh structure depends on the complexity of the geometry. For complex geometries, using a single-block structure may make it difficult to create a high-quality structured grid. Often, this approach leads to high skewness and non-orthogonality. An

advantage of the structured mesh is that the solution strategy for the solver is straightforward due to the consistent number of neighboring cells.

An unstructured mesh is as the name implies, unstructured. This means that the elements do not follow a regular pattern and every cell is considered to be a block. Generating a high-quality unstructured mesh is difficult for several reasons:

- The solution of the numerical equations is complicated as each cell in the mesh has an inconsistent number of neighboring cells.
- Solution time is increased as more demanding algorithms are required.
- Results in the near-wall areas are inaccurate as the boundary layers are not properly resolved.

A hybrid mesh combines the elements of both structured and unstructured meshes. Figure 3.3 illustrates an example of a hybrid mesh, where cell shapes in the near-wall boundary layers allow for improved resolution. The remaining mesh elements can take the shapes shown in Figure 3.4.

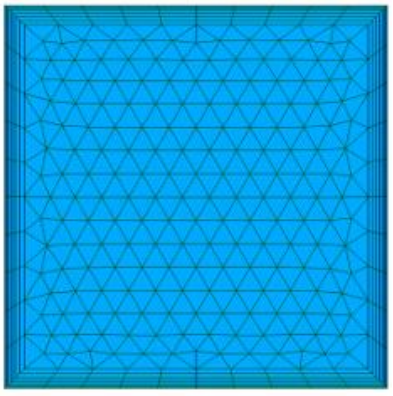

Figure 3.3 - Hybrid mesh (Lande, 2021).

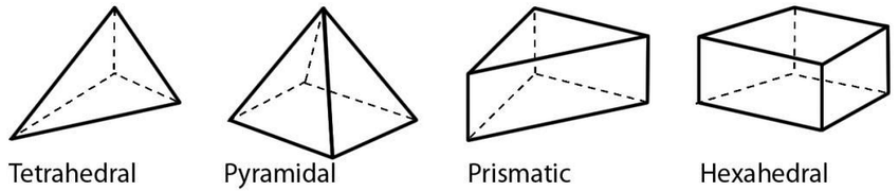

Figure 3.4 – Examples of cell shapes (Setaih et al., 2010).

3.2.2 Mesh quality

The quality of the generated mesh is assessed by observing and calculating the following parameters:

1. Smoothness:
   - Smoothness, also known as expansion rate, growth factor or uniformity, is the transition between cell sizes in the grid. Figure 3.5 illustrates two transition steps where a smooth transition is preferred.

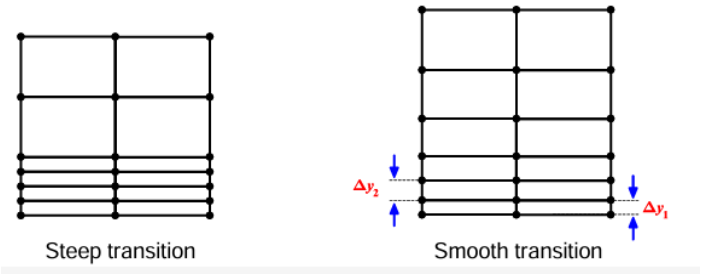

Figure 3.5 – Smoothness (What is a good Mesh?, 2014).

2. Skewness:
   - Skewness defines the ideality of a cell or a face. It describes the angle between the lines of a cell where 90° is the optimal angle. Angles smaller than 45° and larger than 135° are considered highly skewed. It's important to keep the skewness at an acceptable level because initially the solver assumes that the cells are equilateral/equiangular as illustrated in Figure 3.6 (Asyikin, 2012; Tawekal, 2015).

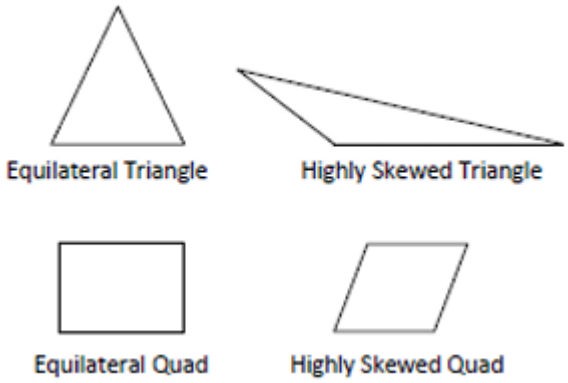

Figure 3.6 - Cell skewness(Asyikin, 2012).

3. Aspect ratio:
   - Mesh aspect ratio is the ratio between the longest side, Δx, and the shortest side Δy. See Figure 3.7 for an illustration.

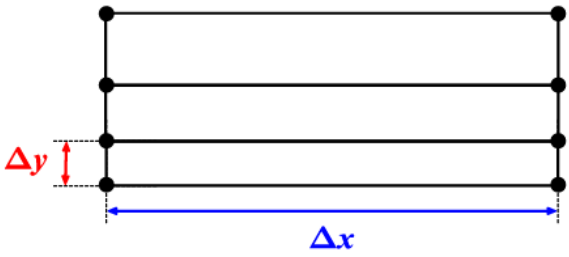

Figure 3.7 – Aspect ratio (Lande, 2021).

   - An aspect ratio (AR) as close to 1 is preferred as it means the cell is equilateral. High aspect rations can increase the numerical diffusion and cause instabilities in the solver (Greenshields, 2023). An AR between 1-20 is recommended in the most important areas of the geometry like the near-wall boundary layers (Mesh Quality, 2022).

4. Orthogonality:
   - Mesh orthogonality is the angular deviation of the vector S (located at the face center f) from the vector d connecting the two cell centers P and N (Figure 3.8) (What is a good Mesh?, 2014).
   - This factor mainly affects the diffusive terms and if the mesh has a high orthogonality, non-orthogonal corrects must be used (Greenshields, 2023).

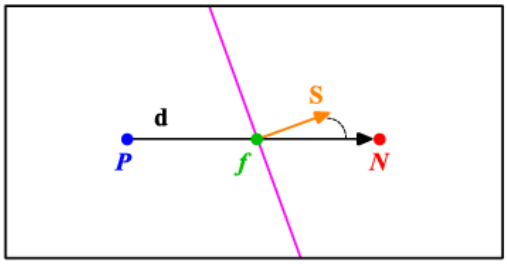

Figure 3.8 - Mesh orthogonality (What is a good Mesh?, 2014).

3.3 Modelling approaches

Most CFD software offer two main approaches for modelling multiphase flows: the Euler-Lagrange approach and the Euler-Euler approach (Lian et al., 2022). In CFD software such as OpenFOAM and ANSYS FLUENT, both approaches are available. Figure 3.9 show the dominant modelling methods for multiphase flows in OpenFOAM, along with suggested solvers.

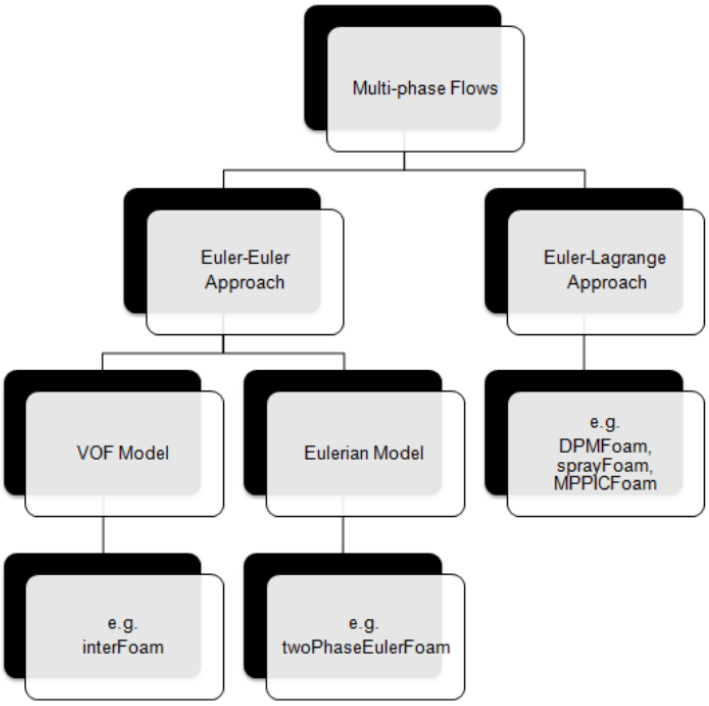

Figure 3.9 - The two dominant methods of solving multi-phase flows in OpenFOAM (Multi-phase flow simulations in OpenFOAM.).

3.3.1 The Euler–Lagrange approach

The Euler-Lagrange approach treats the fluid phase as a continuum by solving the Navier–Stokes equations. This approach is often used when the flow is a dispersion. The dispersed phase is solved by tracking smaller particles like droplets or bubbles, via the computational flow field. Momentum, mass and energy exchange can occur between the dispersed phase and the fluid phase. This approach follows the assumption that the dispersed phase occupies a small volume fraction. Droplet/particle trajectory calculations are done individually at specified intervals during the fluid phase calculation (ANSYS FLUENT 12.0 Theory Guide, 2009).

3.3.2 Euler-Euler approach

In the Euler-Euler approach, the different phases are treated mathematically as interpenetrating continua. Since the volume of a phase cannot be occupied by the other phases, phasic volume fraction is introduced. These volume fractions follow two assumptions: 1) They are continuous functions of space and time and 2) their sum is equal to one. The derivation of the conservation equations for each phase is made to yield a set of equations, with a similar structure for all phases (Adaze et al., 2019). Three different Euler-Euler multiphase models are available in most CFD software: 1) The VOF model; 2) mixture model and 3) Eulerian model.

3.3.2.1 The VOF model

Within the Euler-Euler approach, perhaps the most used model is the Volume of Fluid (VOF) model. This model was developed by Hirt and Nichols (1979) and it received a lot attention within the multiphase community. The VOF model is used for mixtures of two or more immiscible fluids where tracking the interface between the different phases is of much importance. This is done by a surface-tracking function applied to a fixed Eulerian mesh (ANSYS FLUENT 12.0 Theory Guide 2009, Hirt and Nichols, 1979). This function will calculate which fluid composition the cell is occupied by. Alongside the tracking function, a single set of momentum equations is shared by the fluids. The use of the VOF model for two-phase oil-water flows is well documented, please see the works of Alias et al. (2015), Chen et al. (2023), Pouraria et al. (2016), Shuard et al. (2016), Song et al. (2021), Sunday et al. (2023) and Kumara (2010).

3.3.2.2 The mixture model

In the mixture model, the phases are treated as interpenetrating continua. The model solves the momentum equation for the mixture and prescribes relative velocities to describe the dispersed phases (Multi-phase flow simulations in OpenFOAM). This model can be used as a substitute for the Eulerian model as it's less computationally expensive. For this scenario the momentum, continuity and energy equations for the mixture, the volume fraction equations for the secondary phases and the algebraic expressions for the relative velocities are solved simultaneously. This model differs from the VOF model in two ways: 1) the mixture model allows the phases to be interpenetrating. Therefore, the volume fractions α_1 and α_2 in a control volume can be any value between 0 and 1, depending on the space occupied by the phases and 2) the mixture model allows the phases to move at different velocities, using the concept of slip velocities (ANSYS FLUENT 12.0 Theory Guide, 2009).

3.3.2.3 Eulerian Model

The phases are treated as interpenetrating continua and a set of momentum and continuity equations for each phase are solved. Coupling is achieved through the pressure and interphase exchange coefficients. The way this coupling is handled depends on the types of phases involved. This approach is often used if the system contains more than two phases (ANSYS FLUENT 12.0 Theory Guide 2009).

3.4 Governing equations

The flow methodologies described in this section are based on the Reynolds-averaged Navier-Stokes (RANS) equations as most cases require the need to solve turbulent flow. The amount of computational resources needed to solve the instantaneous Navier-Stokes (N-S) equations is unrealistic in most cases and the RANS approach is adopted (Adaze et al., 2019; White, 1991). More details about RANS can be found in (Versteeg and Malalasekera, 2007).

Another approach, known as Direct numerical Simulation (DNS) (Hu et al., 2001; Tryggvason et al., 2001), is based on the finite volume method. This approach completely resolves the turbulence fluctuations, which is advantageous, but it demands significantly more computational power compared to RANS (Sun and Xiao, 2015; Zhu et al., 2007). DNS is slower and more expensive than the RANS approach and is therefore not widely used for turbulent flows. DNS was not considered in this thesis for this reason.

When modelling an oil-water flow, several flow characteristics and assumptions are made to develop a solvable model:

- The density difference and immiscibility of oil and water will cause the liquids to form an interface between them.
- Mass transfers between the phases are ignored.
- The liquids are Newtonian and incompressible.
- Isothermal flow

3.4.1 Stratified flow solvers

When simulating stratified flows, the VOF method is commonly used due to its interface tracking capabilities. In OpenFOAM, there are several solvers for multiphase flow that utilizes the VOF method like interFoam and multiphaseInterFoam.

The total mass of the oil-water phases is constant throughout the cross-sectional area in the axial direction, meaning the conservation equation is simplified to:

$$\frac{\partial \rho}{\partial t} = \nabla.(\rho U) = 0 \tag{3.1}$$

where ρ and U are the density and velocity of the two-phase fluid, respectively.

Given the assumptions previously mentioned in chapter 3.4, the Navier-Stokes equations for a fully developed flow in the axial direction is as follows:

$$\frac{\partial \rho U}{\partial t} + \nabla.(\rho UU) = \nabla P + \nabla.\left[\mu_{eff}(\nabla U + \nabla U^T)\right] + (\rho g sin\theta) + F_s \tag{3.2}$$

$$\frac{\partial \alpha}{\partial t} + \nabla.(\alpha U) + \nabla.(1-\alpha)\alpha U_r = 0 \tag{3.3}$$

where U is the axial velocity, ∇P is the pressure gradient, μ_{eff} is the effective dynamic viscosity ($\mu+\mu_t$), g is the gravitational acceleration, θ is the inclination angle, F_s is the axial direction body force and α is the fluid volume fraction.

Density and dynamic viscosity for the oil-water flow is calculated in each cell volume as follows:

$$\rho = \alpha_w \rho_w + (1 - \alpha_w)\rho_o \tag{3.4}$$

$$\mu = \alpha_w \mu_w + (1 - \alpha_w)\mu_o \tag{3.5}$$

$$\alpha_w + \alpha_o = 1 \tag{3.6}$$

where subscript 'o' and 'w' represent oil and water respectively. The volume fraction has an indicator function within the solver where α is represented as:

$$\alpha \begin{cases} 1 & water \\ 0 < \alpha < 1 & two - phase\ flow \\ 0 & oil \end{cases} \tag{3.7}$$

3.4.2 Dispersed flow solvers

When simulating dispersed flow, the solver framework needs to be able to handle the complex fluid interactions that a dispersed flow regime introduces. This is handled by implementing the Eulerian-Eulerian approach which treats each phase as a continuous fluid with separate volume fractions for each phase (Greenshields, 2023). Below are the governing equations for multiphaseEulerFoam from OpenFOAM which is based on the Eulerian-Eulerian approach.

$$\frac{\partial}{\partial t}(\epsilon_k\ \rho_k) + \nabla \cdot (\epsilon_k\ \rho_k U_k) = 0 \tag{3.8}$$

$$\frac{\partial}{\partial t}(\epsilon_k\ \rho_k U_k) + \nabla \cdot (\epsilon_k\ \rho_k U_k U_k) = -\epsilon_k\ \nabla p + \nabla \cdot (\epsilon_k\ \tau_k) + \epsilon_k\ \rho_k g + M_k \tag{3.9}$$

Where subscript 'k' represents the water and oil phases, respectively. τ is the stress tensor and is calculated as follows:

$$\tau_k = \mu_{eff}\left[\nabla U_k + (\nabla U_k)^T - \frac{2}{3}(\nabla \cdot U_k)\right] I \tag{3.10}$$

As stated in Chen et al. (2023), the momentum transfer per unit of volume, M_k, is affected by several forces. Drag, lift, virtual mass, turbulent dispersion, and wall lubrication forces will affect the momentum transfer. The momentum transfer equation is expressed as:

$$\mathrm{M_o} = -\mathrm{M_w} = \mathrm{M_o^D} + \mathrm{M_o^L} + \mathrm{M_o^{VM}} + \mathrm{M_o^{TD}} + \mathrm{M_o^{WL}} \tag{3.11}$$

Where:

$$M_o^D = \frac{3}{4}\epsilon_o\ \rho_w \frac{C_D}{d_o}|U_w - U_o|(U_w - U_o) \tag{3.12}$$

$$M_o^L = -\epsilon_o\ \rho_w C_L (U_w - U_o) \times (\nabla \times U_w) \tag{3.13}$$

$$M_o^{TD} = -\rho_w k_w C_{TD} \nabla \epsilon_o \tag{3.14}$$

$$M_o^{VM} = \epsilon_o\ \rho_w C_{VM}\left(\frac{DU_w}{Dt} - \frac{DU_o}{Dt}\right) \tag{3.15}$$

$$M_o^{WL} = -\epsilon_o\, \rho_w \max\left(C_{w1} + C_{w2}\frac{R_o}{y}, 0\right)\frac{|U_r-(U_r\cdot n_w)n_w|^2}{R_o} n_w \tag{3.16}$$

Here, R_o and d_o are the oil droplet radius and diameter, n_w is the outward facing unit vector on the wall and y is the distance from the wall. More details on the individual expressions in the momentum equation can be found in the works of Cheng et al. (2018), Deen et al. (2001) and Lopez de Bertodano et al. (1994). C_D represents the drag coefficient and the model used needs to be considered for each unique case. In the case of dispersed oil-water flow, the Ishii and Zuber (1979) model can be applied as shown in Chen et al. (2023):

$$C_D = \frac{2}{3}\left(\frac{g(\rho_w-\rho_o)d_o^2}{\sigma}\right)^{\frac{1}{2}}\left\{\frac{1+17.67|f(\epsilon_o)|^{\frac{6}{7}}}{18.67 f(\epsilon_o)}\right\}^2 \tag{3.17}$$

$$f(\epsilon_o) = \sqrt{1-\epsilon_o}\left(\frac{\mu_w}{\mu_m}\right) \tag{3.18}$$

$$\frac{\mu_m}{\mu_w} = (1-\epsilon_o)^{\frac{2.5(\mu_o+0.4\mu_w)}{\mu_o+\mu_w}} \tag{3.19}$$

It is important to note that the presented equations are not valid when the system experiences phase inversion or separating flows. The phase inversion creates a shift where the dispersed phase can become the continuous phase and vice versa (phase inversion is discussed in chapter 2.2.3). To resolve this, the OpenFOAM framework provides additional interface treatment. Three variations are available: hyperbolic, linear and no blending. The treatment allows for the dispersed phase and continuous phase to be determined locally in each cell. The available blending treatment provides a feasible mechanism to correct the momentum exchange as shown in Chen et al. (2023).

3.5 Turbulence models

Most pipe flow scenarios are turbulent and thus, must be included in the numerical model. This is achieved by applying the RANS equations, supplemented with a turbulence model. Within the RANS framework, the Reynolds stress tensor is modelled to account for the effects of turbulence and increased viscosity. For the turbulence, the k-ε and k-ω models are commonly used. These models are referred to as two-equation models, where the first equation solves for the turbulent kinetic energy, k, and the second equation for the turbulent dissipation, ε, or the specific turbulence dissipation rate, ω. The selected turbulence model is integrated into the RANS framework and computed accordingly.

3.5.1 k-ε model

The k-ε model is a highly efficient turbulence model which can be effectively used for many different two-phase flow situations. The model is popular due to its robustness, computational economy, and reasonable accuracy for a wide range of flows such as stratified flows, churn flows, annular flows, sedimentation and bubbly flows as reported by Han (2005). The k-ε model was initially proposed by Launder and Spalding (1974), and is only valid for high Reynolds numbers, e.g. fully turbulent flows. The equations for the standard k-ε turbulence model are given below.

The k-ε model uses k and ε to define the velocity scale υ and length scale l to represent the large-scale turbulence as follows:

$$\upsilon = k^{\frac{1}{2}} \tag{3.20}$$

$$l = \frac{k^{\frac{3}{2}}}{\varepsilon} \tag{3.21}$$

The eddy viscosity, μ_t, is specified as:

$$\mu_t = C\ \rho\ v\ l = \rho\ C_\mu \left(\frac{k^2}{\varepsilon}\right) \tag{3.22}$$

The transport equations for k and ε are:

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial}{\partial x_i}(\rho k \boldsymbol{U_i}) = \frac{\partial}{\partial x_j}\left(\left(\mu + \frac{\mu_t}{\sigma_k}\right)\left(\frac{\partial k}{\partial x_j}\right)\right) + G_k + G_b - \rho\varepsilon - Y_M \tag{3.23}$$

$$\frac{\partial(\rho\varepsilon)}{\partial t} + \frac{\partial}{\partial x_i}(\rho\varepsilon \boldsymbol{U_i}) = \frac{\partial}{\partial x_j}\left(\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right)\left(\frac{\partial \varepsilon}{\partial x_j}\right)\right) + C_{1\varepsilon}\left(\frac{\varepsilon}{k}\right)(G_k + C_{3\varepsilon}G_b) - C_{2\varepsilon}\rho\left(\frac{\varepsilon^2}{k}\right) \tag{3.24}$$

Where:

- G_k is the generation of turbulent kinetic energy due to the mean velocity gradients.
- G_b is the generation of turbulent kinetic energy due to buoyancy.
- Y_M is the fluctuating dilatation in compressible turbulence in the dissipation rate.
- $C_{1\varepsilon}$, $C_{2\varepsilon}$ and $C_{3\varepsilon}$ are model constants.
- σ_k and σ_ε are related to the turbulent Prandtl numbers.

For an oil-water flow, G_b=0 since the system has a constant temperature and Y_M is neglected due to the flow being incompressible.

These equations contain a total of five adjustable constants which through comprehensive data fitting for turbulent flow has been shown to have the following values (Versteeg and Malalasekera, 2007):

$C_\mu = 0.09$	$\sigma_k = 1.00$	$\sigma_\varepsilon = 1.30$	$C_{1\varepsilon} = 1.44$	$C_{2\varepsilon} = 1.92$

To calculate the Reynolds stresses the Boussinesq relationship is used accordingly (Versteeg and Malalasekera, 2007):

$$\rho\overline{u'_\iota u'_\jmath} = \mu_t\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - \frac{2}{3}\rho k\delta_{ij} = 2\mu_t S_{ij} - \frac{2}{3}\rho k\delta_{ij} \tag{3.25}$$

Several sub-models have been developed from the k-ε model to increase its accuracy for a wider range of situations. One of these models are the RNG-based k-ε model. This model is derived from the instantaneous Navier-Stokes equations using a renormalization group (RNG) method. The model was proposed by Yakhot and Orszag (1986) and made four main

changes to the standard k-ε model: 1) Additional term in the ε equation for improved accuracy for rapidly strained flows; 2) Larger concentrations for swirl effects; 3) Analytical solution for the turbulent Prandtl numbers and 4) Analytical solution for the effective viscosity which accounts for low Reynolds number effects. The governing equations of RNG k-ε are:

$$\frac{\partial(\rho k)}{\partial t}+\frac{\partial}{\partial x_i}(\rho k\boldsymbol{U_i})=\frac{\partial}{\partial x_j}\left(\alpha_k\mu_{eff}(\frac{\partial k}{\partial x_j})\right)+G_k-\rho\varepsilon \tag{3.26}$$

and

$$\frac{\partial(\rho\varepsilon)}{\partial t}+\frac{\partial}{\partial x_i}(\rho\varepsilon\boldsymbol{U_i})=\frac{\partial}{\partial x_j}\left(\alpha_\varepsilon\mu_{eff}(\frac{\partial\varepsilon}{\partial x_j})\right)+C_{1\varepsilon}\left(\frac{\varepsilon}{k}\right)(G_k)-C_{2\varepsilon}^{*}\rho\left(\frac{\varepsilon^2}{k}\right) \tag{3.27}$$

where,

$$G_k=-\rho\overline{u'_\iota u'_J}\frac{\partial U_j}{\partial x_i} \tag{3.28}$$

$C_{2\varepsilon}^{*}$ is calculated by:

$$C_{2\varepsilon}^{*}=C_{2\varepsilon}+\frac{C_\mu\rho v^3\left(1-\frac{v}{v_o}\right)}{1+\beta v^3} \tag{3.29}$$

where,

$$v=\frac{Sk}{\varepsilon} \tag{3.30}$$

For better handling of how the effective turbulent transport varies with the effective Reynolds number, the turbulent viscosity is calculated by the following equation:

$$d\left(\frac{\rho^2 k}{\sqrt{\varepsilon\mu}}\right)=1.72\frac{\hat{v}}{\left(\sqrt{\hat{v}^3-1+C_v}\right)}d\hat{V} \tag{3.31}$$

where,

$$\hat{v}=\frac{\mu_{eff}}{\mu} \tag{3.32}$$

Equation 3.31 allows for better handling of low-Reynolds number and near-wall flows (ANSYS FLUENT 12.0 Theory Guide, 2009).

α_k and α_ε are the inverse effective Prandtl numbers and have the following relationship:

$$\left|\frac{\alpha-1.3929}{\alpha_0-1.39290}\right|^{0.6321}\left|\frac{\alpha+2.3929}{\alpha+2.3929}\right|^{0.3679}=\frac{\mu}{\mu_{eff}} \tag{3.33}$$

The commonly used model constants for the RNG k-ε model are given below (Versteeg and Malalasekera, 2007):

$C_\mu=0.0845$ $\sigma_k=0.7194$ $\sigma_\varepsilon=0.7194$ $C_{1\varepsilon}=1.42$ $C_{2\varepsilon}=1.68$ $v_0=4.38$ $\beta=0.012$

Another k-ε model derivative is the realizable k-ε model developed by Shih et al. (1995). The model introduces a new dissipation rate equation and eddy viscosity formulation. For instances with large mean strain rates (Sk/ε > 3.7), Speziale (1990) showed that the normal stresses can become negative and Schwarz' inequality for shear stresses can be violated. The standard k-ε sets C_μ=0.09 to prevent this, while Shih et al. (1995) proposed the following formulation for C_μ when calculating the eddy viscosity:

$$C_\mu = \frac{1}{A_0 + A_s \frac{kU^*}{\varepsilon}} \tag{3.34}$$

where,

$$U^* = \sqrt{(S_{ij}S_{ij} + \widetilde{\Omega}_{ij}\,\widetilde{\Omega}_{ij}} \tag{3.35}$$

and,

$$\widetilde{\Omega}_{ij} = \Omega_{ij} - 2\varepsilon_{ijk}\omega_k$$

$$\Omega_{ij} = \widetilde{\Omega}_{ij} - \varepsilon_{ijk}\omega_k$$

where $\widetilde{\Omega}_{ij}$ is the mean rate of rotation tensor viewed in a rotating reference frame with the angular velocity ω_k.

where,

$A_0 = 4.04$ and $A_s = \sqrt{6}cos\varphi$

$$\varphi = \frac{1}{3} arc\cos(\sqrt{6}\,W)\ , \qquad W = \frac{S_{ij}S_{jk}S_{ki}}{\widetilde{S^3}}, \qquad \tilde{S} = \sqrt{S_{ij}S_{ij}}\ , \qquad S_{ij} = \frac{1}{2}\left(\frac{\partial U_j}{\partial x_i} + \frac{\partial U_i}{\partial x_j}\right)$$

The equations above shows that the C_μ variable is a function of the mean strain and rotation rates, the angular velocity of the system rotation, and the turbulence fields (ANSYS FLUENT 12.0 Theory Guide, 2009).

The transport equations for the realizable k-ε model are as follows:

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial}{\partial x_i}(\rho k \boldsymbol{U_i}) = \frac{\partial}{\partial x_j}\left(\left(\mu + \frac{\mu_t}{\sigma_k}\right)\left(\frac{\partial k}{\partial x_j}\right)\right) + G_k + G_b - \rho\varepsilon - Y_M \tag{3.36}$$

$$\frac{\partial(\rho\varepsilon)}{\partial t} + \frac{\partial}{\partial x_i}(\rho\varepsilon \boldsymbol{U_i}) = \frac{\partial}{\partial x_j}\left(\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right)\left(\frac{\partial \varepsilon}{\partial x_j}\right)\right) + \rho C_1 S_\varepsilon - \rho C_2\left(\frac{\varepsilon^2}{k+\sqrt{\nu\varepsilon}}\right) + C_{1\varepsilon}\left(\frac{\varepsilon}{k}\right)C_{3\varepsilon}G_b \tag{3.37}$$

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon} \tag{3.38}$$

Where $C_1 = \max\left[0.43, \frac{v}{v+5}\right], v = S\left(\frac{k}{\varepsilon}\right), S = \sqrt{2S_{ij}S_{ij}}. G_k\ and\ G_b$, are the same as with the standard k-ε model.

The realizable k–ε turbulence model is mostly used for high Reynolds-number water-air flows (Passoni et al., 2023). As there is limited information about its use for oil-water flows, little can be said for its accuracy in predicting such a flow. However, initial results from studies by Adaze et al. (2019), Han (2005) and Shih et al. (1995) suggests that the realizable k–ε turbulence model could provide increased accuracy in systems with complex secondary

and separated flows. It remains uncertain whether this would apply to an oil-water flow and should be investigated.

Additionally, there are Low-Reynolds-Number (LRN) versions of the k-ε model. These are shown to be effective in certain cases (Rahman and Siikonen, 2005). For oil-water flows, the model is often used in combination with the standard k-ε model. In these cases, the LRN model deals with the near-wall area and the standard k-ε model is applied for the turbulent flow. An example of this application is documented by Sunday et al. (2023).

3.5.2 k-ω model

Another well-known two-equation turbulence model is the k-ω model proposed by Wilcox (2006). The model replaces the dissipation rate equation from the k–ε model with the eddy frequency (ω), the specific dissipation rate. The model incorporates modifications for low-Reynolds-number effects, compressibility and shear flow spreading (ANSYS FLUENT 12.0 Theory Guide, 2009). The k-ω model is advantageous compared to the k–ε model as it does not require any wall functions for velocity distribution in the near-wall area. This leads to better performance for flows with adverse pressure gradients compared to the k–ε model (Blakeslee, 2021). The transport equations for the k-ω model are:

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial}{\partial x_i}(\rho k \boldsymbol{U_i}) = \frac{\partial}{\partial x_j}\left(\Gamma_k \left(\frac{\partial k}{\partial x_j}\right)\right) + G_k - Y_k \tag{3.39}$$

$$\frac{\partial(\rho \omega)}{\partial t} + \frac{\partial}{\partial x_i}(\rho \omega \boldsymbol{U_i}) = \frac{\partial}{\partial x_j}\left(\Gamma_\omega \left(\frac{\partial \omega}{\partial x_j}\right)\right) + G_\omega - Y_\omega \tag{3.40}$$

Where:

- G_k is the generation of turbulent kinetic energy due to mean velocity gradients and is defined in the same way as in the k–ε model (Equation 3.28).
- G_ω is the generation of ω.
- Γ_k and Γ_ω is the effective diffusivity of k and ω, respectively.
- Y_k and Y_ω is the respective dissipation due to turbulence for k and ω.

The effective diffusivities are given by:

$$\Gamma_k = \mu + \frac{\mu_t}{\sigma_k} \tag{3.41}$$

$$\Gamma_\omega = \mu + \frac{\mu_t}{\sigma_\omega} \tag{3.42}$$

Where σ_k and σ_ω is the turbulent Prandtl numbers. μ_t which represents the turbulent viscosity is computed as follows:

$$\mu_t = \alpha^* \frac{\rho k}{\omega} \tag{3.43}$$

α^* is a Low-Reynolds-Number correction coefficient that damps the turbulent viscosity. It is given by:

$$\alpha^* = \alpha^*_\infty \left(\frac{\alpha^*_0 + \frac{Re_t}{R_k}}{1 + \frac{Re_t}{R_k}}\right) \tag{3.44}$$

Where $Re_t = \frac{\rho k}{\mu \omega}$, $R_k = 6$, $\alpha_0^* = \frac{\beta_i}{3}$ and $\beta_i = 0.072$. It's important to note that in the k-ω model for high-Reynolds-number $\alpha^* = \alpha_\infty^* = 1$.

G_ω is given by:

$$G_\omega = \alpha \left(\frac{\omega}{k}\right) G_k \tag{3.45}$$

In Equation 3.45, α is given by:

$$\alpha = \frac{\alpha_\infty}{\alpha^*} \left(\frac{\alpha_0 + \frac{Re_t}{R_\omega}}{1 + \frac{Re_t}{R_\omega}} \right) \tag{3.46}$$

α behaves the same way as α^* where its equal to 1 in the high-Reynolds numbers.

The dissipation of k is given as follows:

$$Y_k = \rho \beta^* f_{\beta^*} k\omega \tag{3.47}$$

Where,

$$f_{\beta^*} = \begin{cases} 1 & x_k \leq 0 \\ \frac{1+680x_k^2}{1+400x_k^2} & x_k > 0 \end{cases} \tag{3.48}$$

where

$$X_k = \frac{1}{\omega^3} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j} \tag{3.49}$$

And

$$\beta^* = \beta_i [1 + \varsigma^* F(M_t)] \tag{3.50}$$

$$\beta_i^* = \beta_\infty \frac{\left(\left(\frac{4}{15}\right) + \left(\frac{Re_t}{R_\beta}\right)^4 \right)}{1 + \left(\frac{Re_t}{Re_\beta}\right)^4} \tag{3.51}$$

The dissipation of ω is given by:

$$Y_\omega = \rho \beta f_\beta \omega^2 \tag{3.52}$$

where

$$f_\beta = \frac{1+70x_\omega}{1+80x_\omega} \tag{3.53}$$

$$X_\omega = \left| \frac{\Omega_{ij}\Omega_{ij}S_{ki}}{(\beta_\infty^* \omega)\text{^}3} \right| \tag{3.54}$$

$$\Omega_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i} \right) \tag{3.55}$$

The strain rate tensor, $S_{ki} = \frac{1}{2}\left(\frac{\partial U_j}{\partial x_i} + \frac{\partial U_i}{\partial x_j}\right)$ and β_i^* is defined in Equation 3.51, whilst β is defined as:

$$\beta = \beta_i \left[1 - \frac{\beta_i^*}{\beta_i} \varsigma^* F(M_t)\right] \quad (3.56)$$

$F(M_t)$ is the compressibility function and is given by:

$$F(M_t) = \begin{cases} 0 & M_t \leq M_{t0} \\ M_t^2 - M_{t0}^2 & M_t > M_{t0} \end{cases} \quad (3.57)$$

Here $M_t^2 = \frac{2k}{a^2}$, $M_{t0} = 0.25$ and $a = \sqrt{\gamma RT}$. Here γ is the compressibility factor, R is the universal gas constant and T the absolute temperature. It's important to note that for ω, as with k, that in the high-Reynolds number form of the k- ω model, $\beta_i^* = \beta_\infty^*$ and for the incompressible form $\beta^* = \beta_i^*$.

Due to the k- ω model's strong sensitivity to the freestream boundary condition for external flow applications (Wilcox 2006), a modified version was suggested by Menter (1994). Menter (1994) proposed a new model called Shear Stress Transport k-ω where this sensitivity is overcome. There's been several iterations of this model over the years, but most CFD software use the version developed by Menter et al. (2003). The SST model effectively blends the robust and accurate formulation of the k-ω model in the near-wall region with the free-stream independence of the k-ε model in the far field. This is done by converting the k-ε model into a k-ω formulation. The SST model is similar to the k-ω model but includes the following refinements (ANSYS FLUENT 12.0 Theory Guide, 2009):

- The SST model incorporates a damped cross-diffusion derivative term in the ω equation.
- The model has a blending function which is designed to be one in the near-wall region which activates the k-ω model, and zero away from the surface, which activates the k-ε model.
- Turbulent viscosity is modified with regards to the transport of the turbulent shear stress.
- The model constants are different.

Further details about the SST k-ω model and its equations can be found in ANSYS FLUENT 12.0 Theory Guide (2009). The refinements make the model more accurate and reliable than the standard version. Additionally, the SST k-ω model makes it easier to estimate the onset and degree of flow separation under adverse pressure gradients by including transport effects into the eddy-viscosity approximation (De la Cruz-Ávila et al., 2022).

3.5.3 Near wall treatment

Near wall treatment considerations are important when dealing with turbulent flows, as the presence of walls significantly influences turbulent behavior. The mean velocity field is affected through the no-slip condition that must be satisfied at the wall. The turbulence is also changed by the presence of a wall in non-trivial ways. Very close to the wall, viscous damping reduces tangential velocity fluctuations, while kinematic blocking reduces the normal fluctuations. Towards the outer part of the near wall region, turbulence is rapidly augmented due to the production of turbulence kinetic energy due to the large gradients in the mean velocity (ANSYS FLUENT 12.0 Theory Guide, 2009).

There are two approaches to modelling the near-wall region. One approach involves integrating the turbulence model to resolve the viscosity-affected region using a fine mesh all the way to the wall, including the viscous sublayer (preferably $y^+=1$). This approach fits turbulence models that can solve the flow in the near-wall area like the SST k-ω model. The second approach is using wall functions for the near-wall area. Wall functions are empirical equations utilized to capture the physics of the flow in the near-wall region. The functions bridge the inner region between the wall and the fully developed turbulence region, providing near-wall boundary conditions for the momentum and turbulence transport equations, rather than to specify those conditions directly at the wall itself (Versteeg and Malalasekera, 2007). When used correctly, wall functions yield a relatively accurate result, while significantly reducing computational time (Davidson, 2022).

4 Simulation strategy

The simulation work in this thesis is done using the open source CFD software OpenFOAM. OpenFOAM (Open Field Operation and Manipulation) is a Linux based C++ toolbox for customized numerical solvers. The software is open source which allows the user to make any changes to the source material. This makes OpenFOAM a very flexible numerical tool and can be customized for almost any system. With the *paraView* utility the simulated data is visualized and enables the creation of 2D/3D plots as well as extraction of data points.

4.1 OpenFOAM simulation structure

The case structure for a simulation in OpenFOAM is always the same regardless of the system. The setup consists of 3 folders: 0, constant and system. Each folder contains a different part of the simulation setup and must be configured correctly for each system. Figure 4.1 illustrates an example of how a case structure could look like. Each part of the simulation structure is presented and explained in detail in chapter 4.3.

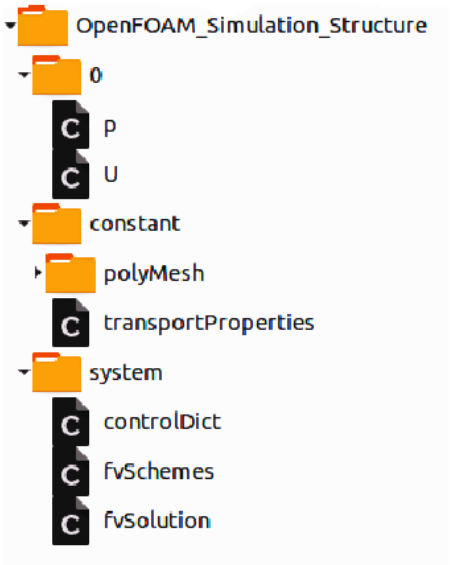

Figure 4.1 - OpenFOAM case structure example (Medina et al., 2015).

4.2 Pre-processing

4.2.1 Pipe geometry.

The geometry for this CFD study is based on the experimental work of Kumara (2010) and was created by the group project of Vindenes et al. (2021). The computational domain is set up to emulate the test section of the test rig. The test rig is shown in Figure 4.2 and the test

section is shown in Figure 4.3. The test section is a circular, stainless-steel pipe with a length of 15 meters and diameter of 56 mm.

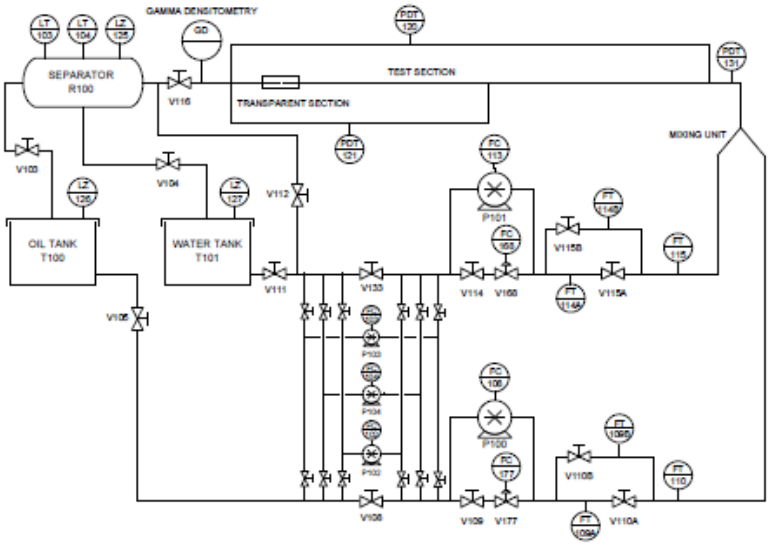

Figure 4.2 – Simplified flow sheet of the test rig (Kumara et al., 2009).

In the test section a single-beam gamma densitometry was used to ascertain the phase fraction measurements over the cross-sectional area of the pipe. Particle image velocimetry (PIV) was used to measure the velocity and differential pressure transmitters for the pressure drop over the test section. Towards the end of the test section, a short transparent pipe section was fitted for visual observations. Figures 4.2 and 4.3 are images taken from this pipe section.

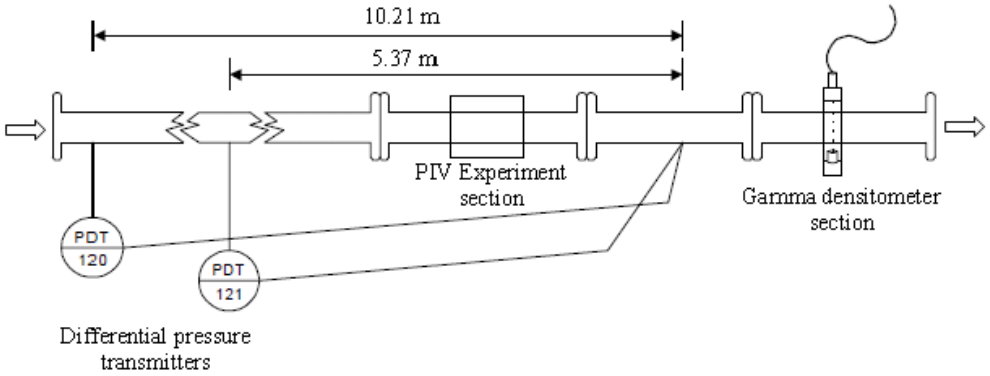

Figure 4.3 - Test section for the experimental study by (Kumara et al., 2009).

4.2.2 Mesh generation.

The mesh used for this study is the one created by the group project of Vindenes et al. (2021). The mesh is created using SALOME 9.7 which is an open-source 3D CAD software. This software provides a parametric approach when designing and meshing geometries. This allows for a visual approach when generating the mesh and gives the user more information when controlling the mesh quality.

When generating a mesh for a circular pipe there is two common structures that are used: the OH structure (butterfly grid) and tetrahedral structure (Lande, 2021). The OH structure effectively splits the pipe into five blocks. This allows for very fine boundary layers with good resolution at the pipe wall but leaves the center of the pipe with a coarser mesh. In total the group project made three mesh iterations in which two were OH meshes. The OH meshes can be seen in Figure 4.4.

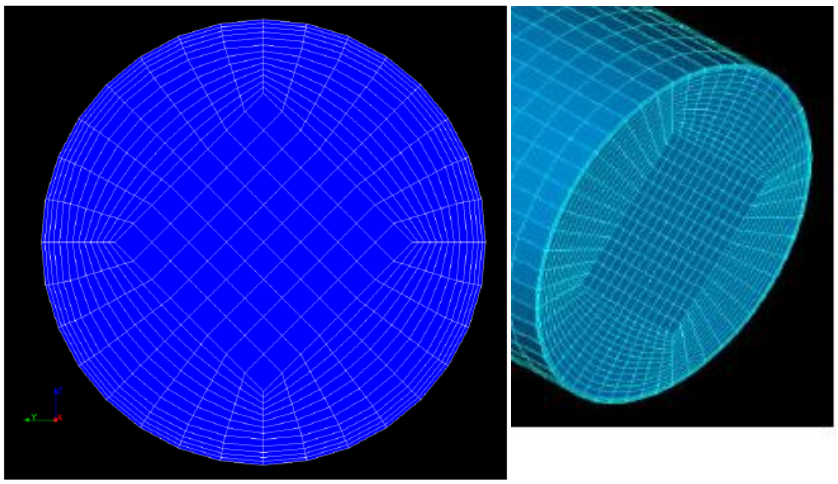

Figure 4.4 - OH mesh 1 & 2 (Vindenes et al. 2021).

From Figure 4.4 one can see that there is bending of the boundary layers at the corners of the block structure. This reduces the quality of the mesh as the skewness increases, which can cause instabilities for the solver. The second mesh used the same structure but with a finer mesh. Some of the boundary layer bending was resolved but led to a doubling of the total number of cells and required more computational power. The third iteration was created using a MEFISTO tetrahedral grid (CAD Exchanger SDK: Computational meshers). This mesh consists of 10 viscous layers and a linearly increasing spacing along the pipe. This hybrid mesh had satisfactory resolution for the boundary layers while providing a finer mesh in the central areas. The created mesh was imported from SALOME to OpenFOAM. More details about the mesh are found in APPENDIX B. The final mesh iteration is shown in Figure 4.5.

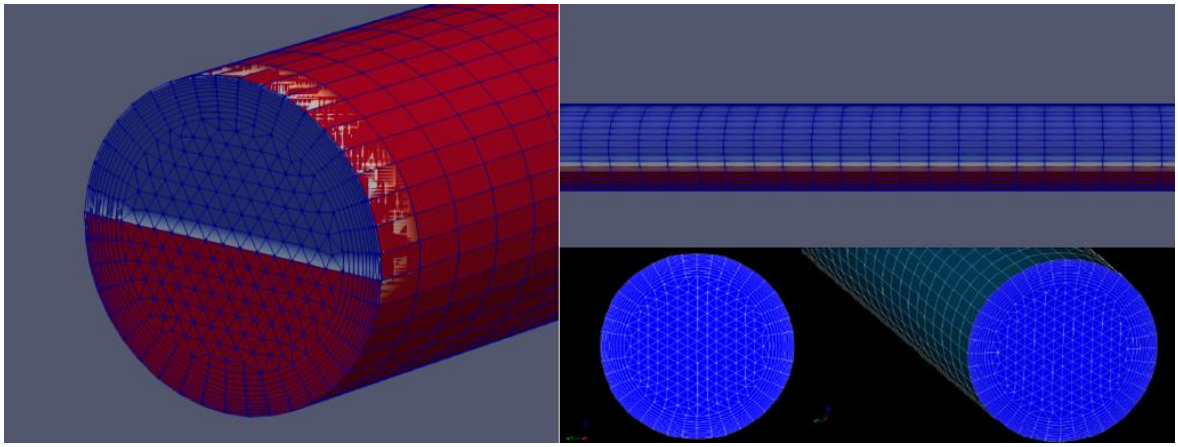

Figure 4.5 - Mesh iteration 3 (a) Cross sectional, (b) axial view and (c) isometric view (Vindenes et al. 2021).

4.3 Case structure

In this section, each folder in the simulation setup is presented and explained. All the base files are taken from the multiphase dam break tutorial in OpenFOAM and are customized for this study. The different simulation files can be found in Appendix C-E.

4.3.1 0 folder

The boundary conditions for the simulations are found in the '0' folder in OpenFOAM. For stratified oil-water simulations the inlet section in the geometry is divided into two equal sections as seen in Figure 4.6. Oil is introduced in the top section of the pipe and water at the bottom part of the pipe. This leads to a stratified flow just behind the inlet section due to the density differences of the fluids. Figure 4.7 shows how the inlet section is divided in OpenFOAM. Here, blue and red represents the oil and water inlet, respectively.

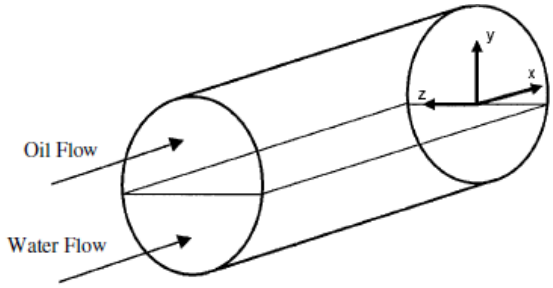

Figure 4.6 - Schematic representation of stratified oil-water flow (Kumara 2010).

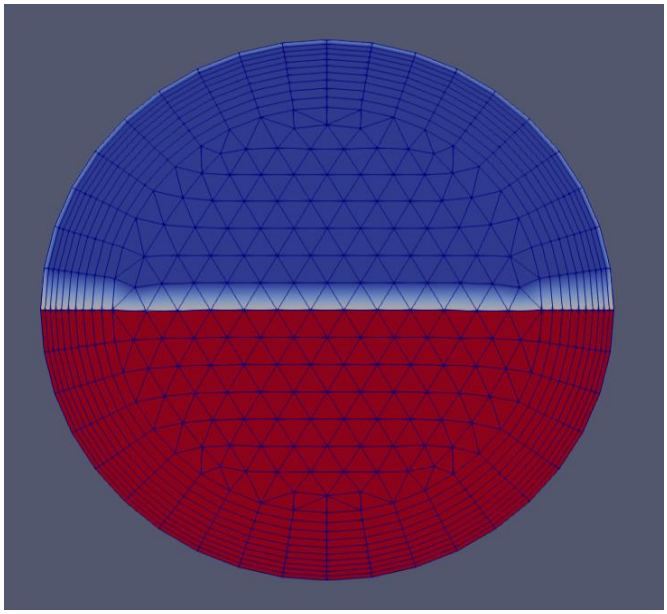

Figure 4.7 - Inlet section in the OpenFOAM environment.

The '0' folder includes 6-7 different files, depending on what solver and turbulence model are used, that are used to define the boundary conditions for the simulation. The boundary conditions for a simulation applying the SST k-ω turbulence model and the interFoam solver with a mixture velocity of 0.50 m/s and 0.25 water cut, are presented in table 4.1.

The water cut is decided by manipulating the inlet velocities. Since the area of the inlets are equal, Equation 2.1 and 2.4 can be used to determine the inlet velocity for oil and water. Most outlet values are defined as zeroGradients which means that the specific value normal to the flow outlet is zero. For the velocity (U), the no slip condition is applied at the walls, while wall functions are utilized for the turbulence parameters such as omega, k, epsilon and nut. For the hydrostatic pressure (p_rgh), a fixed flux pressure is applied. This adjusts the pressure gradient so that the boundary flux matches the velocity boundary condition for solvers that include body forces such as gravity and surface tension (Greenshields, 2023). Lastly, the k and omega starting values are calculated using Equations 4.1-4.3.

$$k = \frac{3}{2}(I|U|)^2 \tag{4.1}$$

$$\omega = \frac{k^{0.5}}{C_\mu^{0.25} L} \tag{4.2}$$

$$I = 0.16 * Re^{-\left(\frac{1}{8}\right)} \tag{4.3}$$

where I is the turbulence intensity, U is the inlet velocity and Re is the Reynolds number.

Table 4.1 - Boundary conditions for a simulation using the SST k-ω turbulence model and interFoam solver.

	U [m/s]	Alpha.water [-]	P_rgh [kg/ms^2]	Nut [m^2s^{-1}]	Omega [s^{-1}]	K [m^2s^{-2}]
Inlet_water	fixedValue Value: 0.25	fixedValue Value: 1	fixedFluxPressure Value: 100000		fixedValue Value: 1.3472	fixedValue Value: 0.0017
Inlet_oil	fixedValue Value: 0.75	fixedValue Value: 0	fixedFluxPressure Value: 100000		fixedValue Value: 0.5136	fixedValue Value: 0.00022
Outlet	zeroGradient	zeroGradient	fixedFluxPressure Value: 100000		zeroGradient	zeroGradient
Wall	noSlip	zeroGradient	fixedFluxPressure Value: 100000	nutkWallFunction Value: 0 “.*” - calculated	omegaWallFunction Value: 1	kqRWallFunction Value: 0.003
defaultFaces	Empty		Empty	Empty	Empty	Empty

4.3.2 Constant folder

In the ‘constant’ folder, the physical properties of the fluid are specified. This folder contains three files: ‘g’, ‘momentumTransport’ and ‘transportProperties’. The ‘g’ file defines the gravitational force that affects the system. The ‘momentumTransport’ file specifies whether the flow is laminar or turbulent, and which turbulence model is used is specified in this folder, if turbulent flow is chosen. The ‘transportProperties’ file specifies the values of fluid properties like kinematic viscosity, density and surface tension. Whether the liquid is Newtonian or non-Newtonian, is also defined here. The generated mesh is also located in this folder in the form of a “blockMeshDict” or a “polyMesh” folder. Since the generated mesh was imported from SALOME, a polyMesh folder was created due to the size of the geometry.

4.3.3 System folder

The ‘system’ folder consists of five files which control how the simulation is performed. In these files the time step, discretization schemes, solution criteria etc. are set.

4.3.3.1 Controldict

At the beginning of every OpenFOAM simulation the solver sets up a database that control the input and output of the simulation. In the controldict dictionary several essential control parameters can be adjusted, like time step, courant number and run time. For simulations using the interFoam solver, the courant number should never exceed 0.5 (Greenshields, 2023). The courant number is defined as:

$$Co = \frac{\Delta t|U|}{\Delta x} \tag{4.4}$$

Where Δx is cell width in the velocity direction, U is the velocity and Δt is the time step. This means in order for the solver to maintain a max courant number of 0.5, the time step must be adjustable. This essentially means if the velocity increases, the time step must decrease. The values used for the simulations in this thesis is:

- Δt = 0.001
- maxCo = 0.5
- maxDeltaT = 0.1

These values were chosen to ensure a smooth simulation as well as maintaining an accurate solution.

4.3.3.2 decomposeParDict

Oil-water simulations are complex and requires a lot of calculations. To cut down on how long each simulation took, the computational domain was decomposed (split) into several sub-domains. The number of sub-domains is user specified but cannot exceed the number of available CPUs on the computer. The computer that the simulations were performed on had a 6 core CPU, so 6 sub-domains with an equal number of cells were created.

4.3.3.3 fvSchemes

Solving the equations required for the simulation involves discretizing them (Versteeg and Malalasekera, 2007). OpenFOAM has a large library for this and can be customized according to the requirements of the user. In the 'fvscheme' file all the discretization schemes for the different equations are chosen. The different schemes are as follows (Greenshields, 2023):

- ddtSchemes: This sets the time scheme. For this thesis the Euler scheme is used which is a transient, first order bounded implicit scheme. This scheme fits well with the small-time steps that are used when bounding the courant number to 0.5.
- gradSchemes: The gradient term scheme. Here gauss linear is used which indicates a standard finite volume discretization with Gaussian integration and linear interpolation.
- divSchemes: Divergence scheme, i.e. terms of the form $\nabla \cdot$. These are the most important schemes in CFD simulations. These types of schemes always use the Gauss scheme and the difference in selection comes from the interpolation scheme. Depending on which field is discretized the interpolation schemes are different. Most advective terms are usually in the form div(phi, …) where phi denotes the volumetric flux of velocity on the cell faces. For example, in this thesis the velocity flux field is denoted div(rho*phi, U) and uses the Gauss linearUpwind grad(U) scheme. LinearUpwind is an unbounded second order, upwind-biased scheme that is well suited for velocity. Additionally, for the two-phase flow in this thesis, the div(phi, alpha) scheme is very important as this calculates and reconstructs the cells that contain the oil-water interface.
- laplacianSchemes: Defines the Laplacian scheme which is applied to equation terms with the Laplacian operator ∇^2. This is set to the default option which is Gauss linear corrected. 'Corrected' indicates an unbounded, conservative and second order approach to the surface normal gradient.
- interpolationSchemes: This sub-dictionary contains terms that are interpolations of values, typically from cell centers to face centers. Mostly used for the interpolation of velocity to face centers in the calculations of the flux (phi). This is by default set to linear.
- snGradSchemes: Stands for surface normal gradient schemes. This scheme evaluates the gradient normal to the face center shared by two cells. This scheme is set to 'corrected'.

4.3.3.4 fvSolution

This sub-directory decides which equation solver is used, tolerances and solution algorithms. In the simulations the PIMPLE algorithm is used for the pressure-velocity coupling. The pimple algorithm combines the PISO and SIMPLE algorithms (Greenshields, 2023). Figure 4.8 illustrates the solution strategy for the PIMPLE algorithm. In the simulations 'nCorrectors' is set to 2, which indicate how many times the pressure equation is solved in the outer loop.

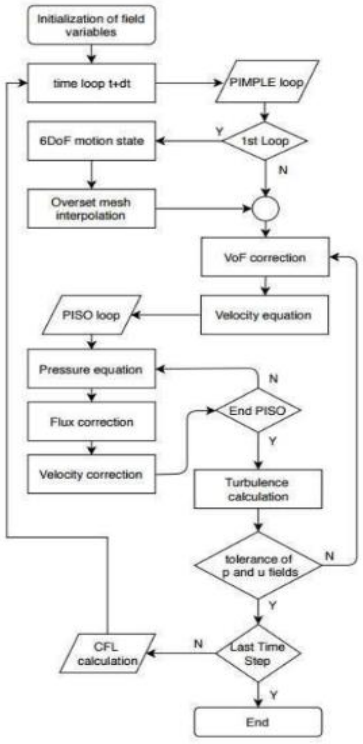

Figure 4.8 - PIMPLE algorithm in OpenFOAM (Niotis et al., 2019).

For the volume fraction several inputs must be made. ‘nAlphaCorr’ is set to 1 and indicate how many times the algorithm loops the volume fraction calculation. ‘nAlphaSubCycles’ is the number of times the volume fraction is calculated per ‘nAlphaCorr’. This number is set to 1 in the simulations. ‘cAlpha’ is the compression term at the interface for the transport equation. Here a value of 2 is chosen meaning an enhanced treatment for the oil-water interface. The remaining inputs in ‘fvSolution’ involve pressure correction and turbulence model parameter corrections. Some of the ‘fvSolution’ settings are presented in table 4.2, and the complete file is found in Appendix E.

Table 4.2 – Solution criteria used for the simulations.

	P_corr	P_rgh	P_rghFinal	(U|k|omega|epsilon)	(U|k|omega|epsilon) Final
Solver	PCG	GAMG	GAMG	smoothSolver	smoothSolver
Preconditioner	GAMG				
Smoother	GaussSeidel	GaussSeidel		symGaussSeidel	symGaussSeidel
Tolerance	$1*10^{-5}$	$5*10^{-9}$	$5*10^{-9}$	$1*10^{-6}$	$1*10^{-8}$
relTol	0	0	0	0	0

GAMG (Geometric-agglomerated Algebraic Multigrid) is used for its ability to generate a quick solution. It does this by generating a quick solution on a mesh with small number of cells; mapping this solution onto a finer mesh; using it as an initial guess to obtain an accurate solution on the fine mesh (Greenshields, 2023). By applying a smoothing scheme such as Gauss-Seidel this process results in a quicker calculation time.

4.3.3.5 setFieldsDict

The 'setFieldsDict' sub-directory is used to pre-fill the pipe with water before starting the simulation. This technique is advantageous as it reduces the simulation time.

4.4 Interface treatment

As reviewed in chapter 2, the need for additional interface treatment for a two-phase oil-water flow is essential for increasing the simulation accuracy. Two interface treatment approaches were attempted in this thesis: 1) Customizing the k-ε turbulence model to apply wall-like conditions to the oil-water interface and 2) turbulence damping at the oil-water interface.

The application of the wall-like conditions to the oil-water interface as outlined in Liu et al. (2022) was attempted. The source code for the k-ε turbulence model was taken from the OpenFOAM framework. The code was then customized so that the proposed conditions only applied in the cells that contained the oil-water interface. Several attempts were made to make this work but ultimately failed due to lack of coding knowledge in C++. The customized code "worked" but the logic implemented to find the interface cells was unsuccessful, so the proposed conditions were never applied to the simulation. The customized code can be found in Appendix F.

For the application of turbulence damping at the oil-water interface, the scheme developed by Fan and Anglart (2019, 2020) was used. This turbulence damping scheme was applied to the entire flow field, so it had to be customized so that it was only applied at the cells containing the oil-water interface. As with the turbulence model, this was a complicated task. This customization also failed due to the same reasons the customized turbulence model did. This led to the simulations being performed with only the interface treatment provided by OpenFOAM. The source-code for the turbulence damping scheme is found in Appendix G.

4.5 Processing

The chosen solvers for the two-phase oil-water simulations were interFoam and multiphaseInterFoam. These solvers are based on the VOF model which is discussed in detail in chapter 3. Although subtle, the difference between the two solvers is that the multiphaseInterFoam solver includes additional surface-tension and contact-angle effects for each phase. Initial simulations were performed to determine which solver to use going forwards. The differences between the two solvers are discussed in chapter 5.2.

The simulations performed in this study are done at three different water cuts for two different mixture velocities. A mixture velocity of 0.50 m/s was used for water cuts 0.25, 0.50, 0.75, and 0.25 and 0.50 for a mixture velocity of 0.68 m/s.

4.6 Post-processing

The post-processing work is done with the paraView utility. ParaView is launched by writing *paraFoam* in the terminal. Simulation data is analyzed using the *plot over line* tool which allows for plotting data at any point, and any direction. The mean axial velocity and water volume fraction is found by plotting a straight line through the center of the pipe in the Y-direction. The data is then converted to an Excel sheet and plotted against experimental data.

5 Results & discussion

This section presents and discusses the results of the CFD study. An analysis of the differences between interFoam and multiphaseInterFoam are presented first. A comparison of the performance of different turbulence models against experimental data follows.

5.1 General comments

- The experimental data used for the comparisons are collected using an online graph analyzing tool called *Web Plot Digitizer*. Hence, the accuracy for the gathered experimental data will not be 100% accurate. The figures used to capture the experimental data is found in Appendix H.
- The simulation data are gathered at 14.99m downstream of the inlet. This point was chosen to ensure that the flow was fully developed.
- The total time in which each case was simulated were different. Each case was simulated to the point where a steady state was reached. How long this took differed for each case as different turbulence models were used and the initial conditions changed for each case. The average solution time for each case is presented in table 5.1.

Table 5.1 – Average simulation times for the different turbulence models at different water cuts and U_m=0.50 m/s.

Water cut	SST k-ω [s]	RNG k-ε [s]	Realizable k-ε [s]
$\lambda_w = 0.25$	60	60	430
$\lambda_w = 0.50$	40	40	60
$\lambda_w = 0.75$	70	70	520

5.2 Solver comparison

A part of the simulation study was to assess which solver was the most suitable to use for the oil-water flow. Two solvers were assessed: interFoam and multiphaseInterFoam. A comparative analysis was done for three different water cuts employing the RNG k-ε and SST k-ω turbulence models at a mixture velocity of 0.50 m/s.

5.2.1 RNG k-ε

Figures 5.1-5.3 show the results from the two solvers obtained using the RNG k-ε turbulence model for the three water cuts. For λ_w=0.25 and 0.50, the two solves produce near identical results as observed in Figures 5.1 and 5.2. In Figure 5.3 it's seen that when the water cut is increased to 0.75, the interFoam solver predicts a slightly lower velocity in the water phase. The two solvers consistently predict more or less the same position for the oil-water interface

for the three water cuts. It's evident that when using the RNG k-ε turbulence model, the additional contact angle and surface tension calculations in multiphaseInterFoam do not have any significance on the results for this geometry.

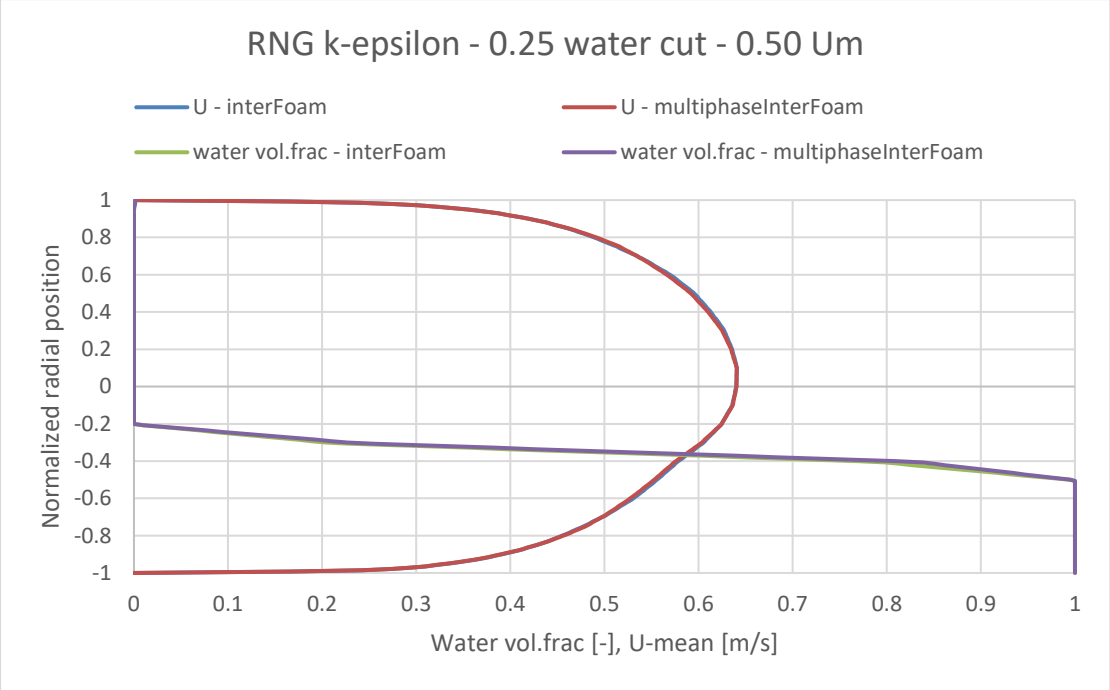

Figure 5.1 – Comparison of interFoam and multiphaseInterFoam for the RNG k- ε turbulence model with λ_w=0.25.

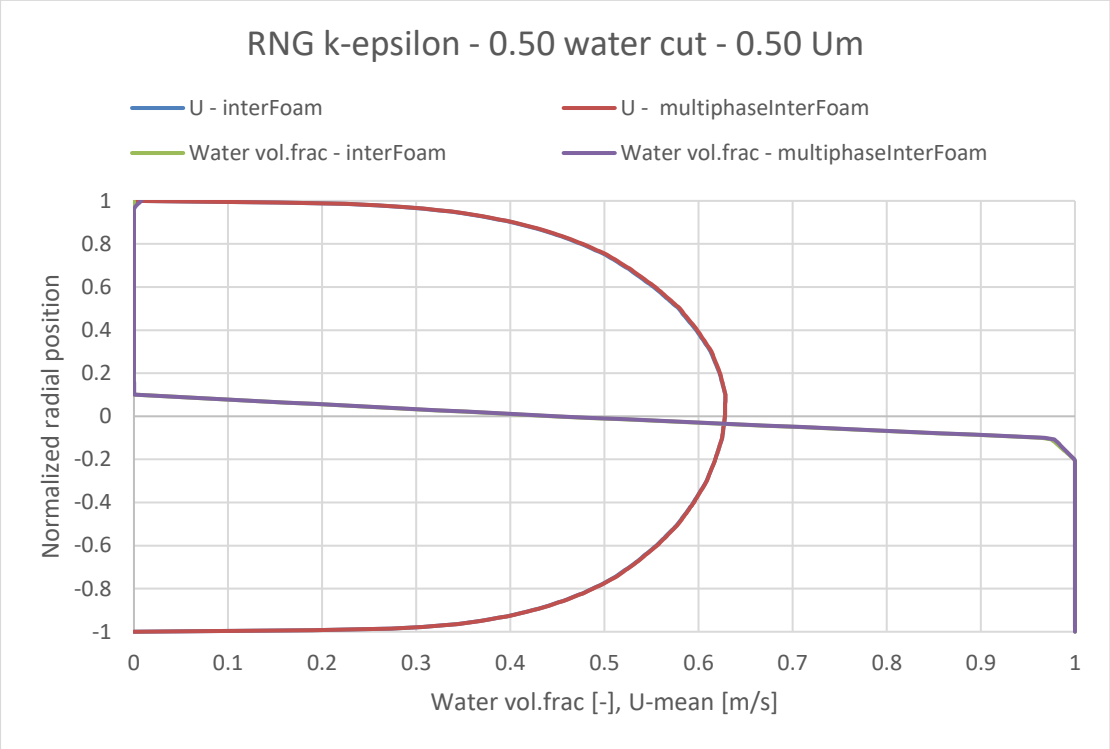

Figure 5.2 - Comparison of interFoam and multiphaseInterFoam for the RNG k- ε turbulence model with λw=0.50.

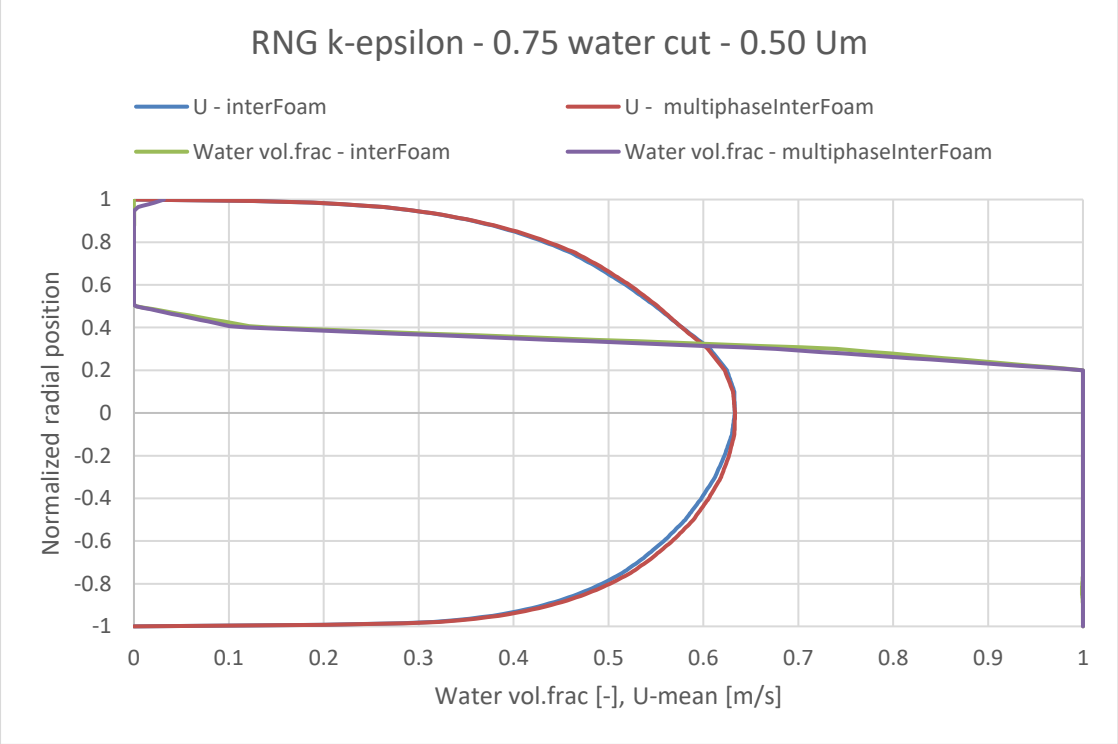

Figure 5.3 - Comparison of interFoam and multiphaseInterFoam for the RNG k- ε turbulence model with λw=0.75

5.2.2 SST k-ω

Figures 5.4-5.6 show the results from using the two solvers with the SST k-ω turbulence model for the three water cuts. The results present a very similar trend to those obtained with the RNG k-ε turbulence model, but with some differences. For λ_w=0.25, a slight deviation is seen between the solvers. Figure 5.4 show that interFoam predicts a slightly higher velocity in the oil phase, and as it transitions from the oil phase to the water phase, it predicts a marginally lower velocity compared to multiphaseInterFoam. The position of the oil-water interface for the two solvers shows very little difference, bar a slight deviation at radial position -0.4 in Figure 5.4.

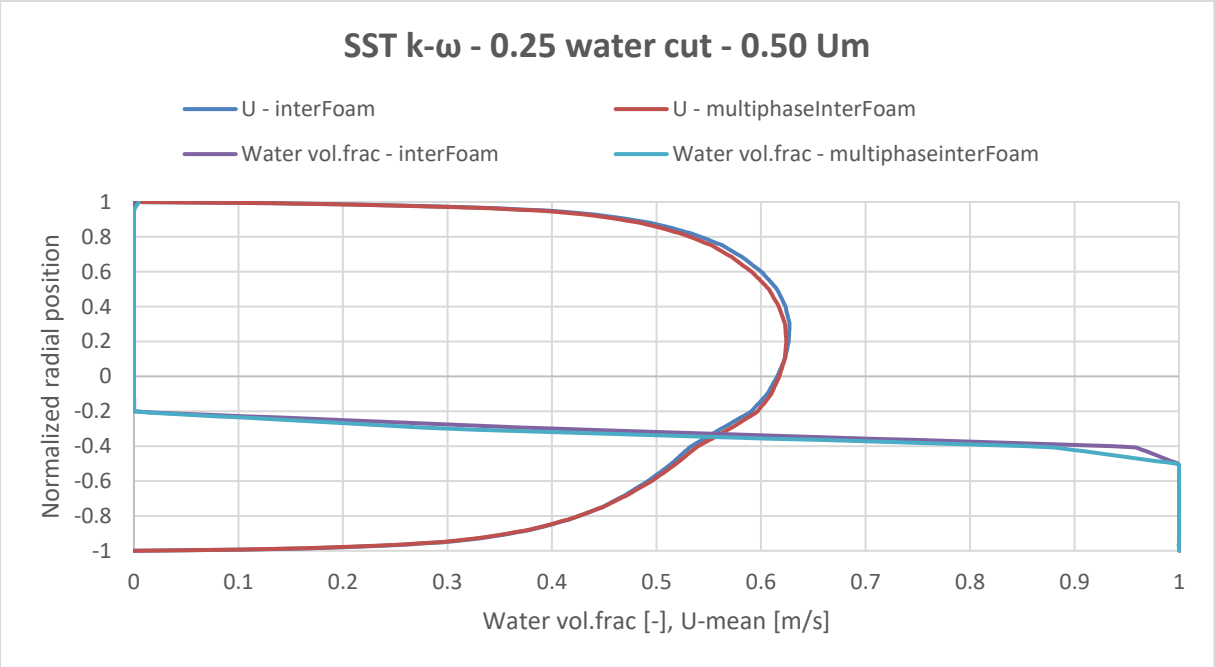

Figure 5.4 - Comparison of interFoam and multiphaseInterFoam for the SST k- ω turbulence model with λw=0.25.

Figures 5.5 and 5.6 show that for λ_w=0.50, interFoam slightly underpredicts the oil phase velocity, while for λ_w=0.75, it overpredicts the oil phase velocity compared to multiphaseInterFoam. Additionally, as observed in the other simulations, the position of the oil-water interface is near identical between the two solvers.

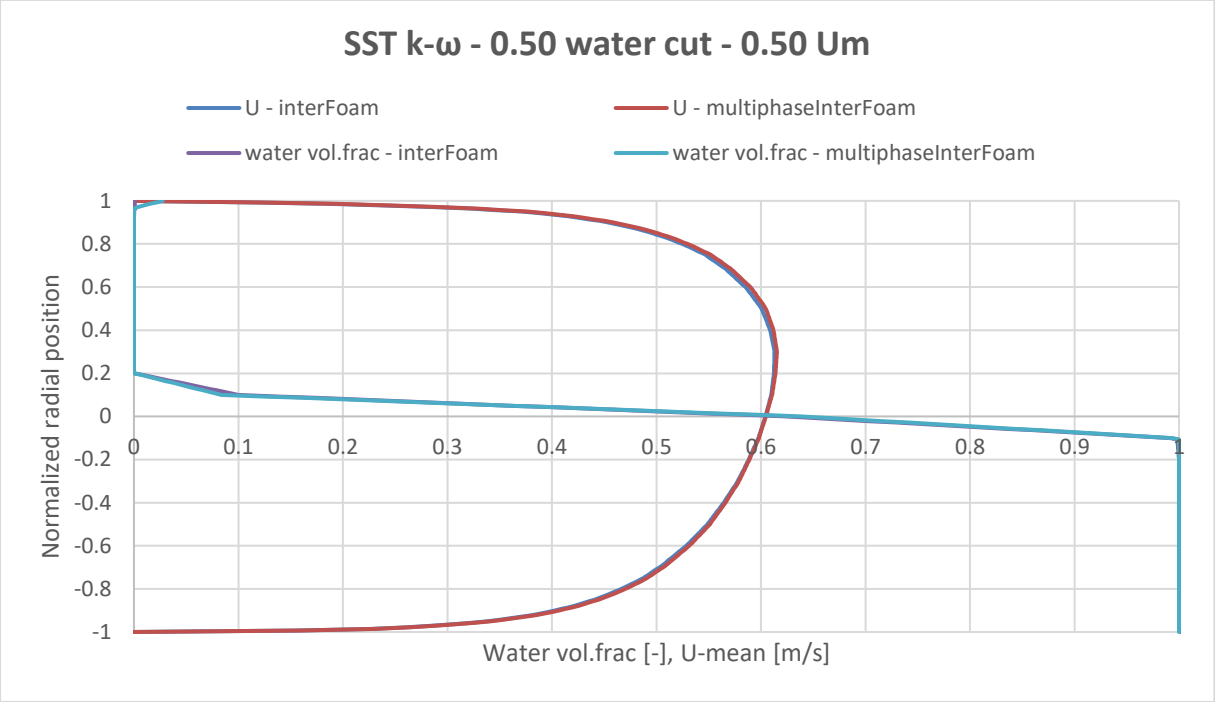

Figure 5.5 - Comparison of interFoam and multiphaseInterFoam for the SST k- ω turbulence model with λw=0.50.

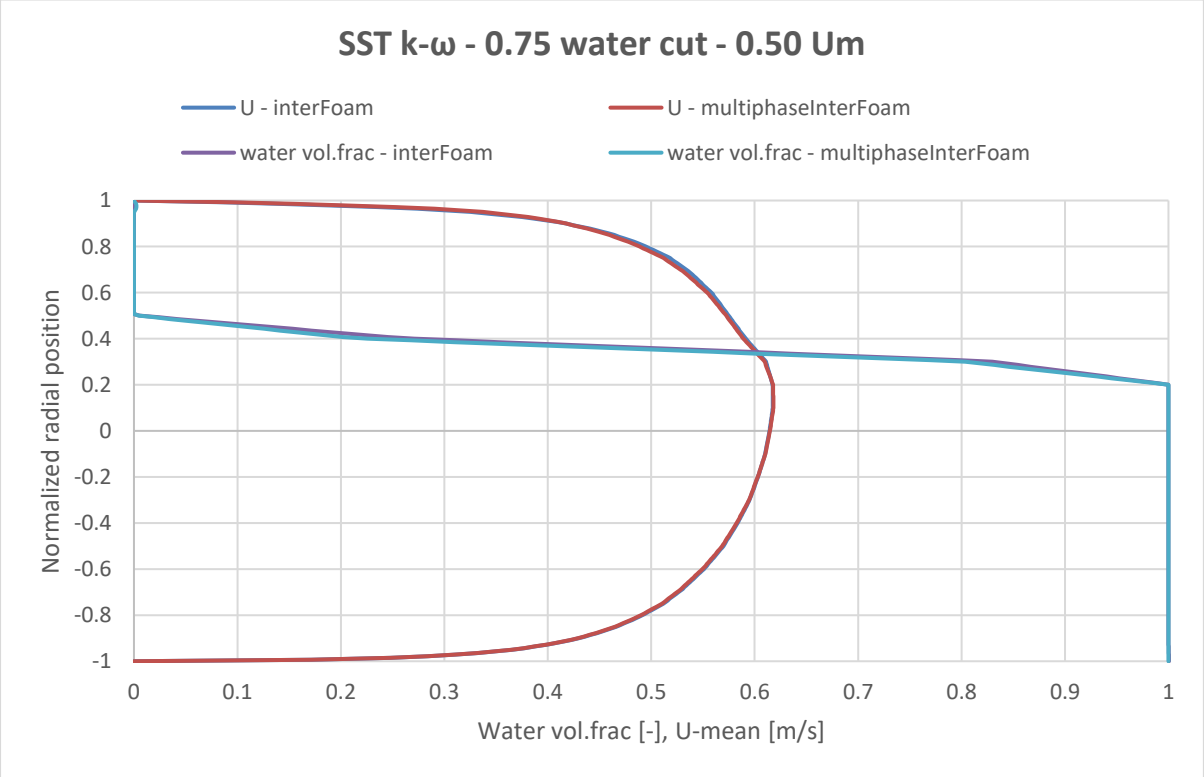

Figure 5.6 - Comparison of interFoam and multiphaseInterFoam for the SST k- ω turbulence model with λw=0.75

Although the differences in the results obtained using interFoam and multiphaseInterFoam are minimal, there was a notable difference in the simulation time. For a mixture velocity of

0.50 m/s using the RNG k-ε model, the interFoam solver used around 8hours, compared to the multiphaseInterFoam solver which used upwards of 14 hours. So, when considering the choice between the two solvers, interFoam was the preferred option due to its shorter computational time. However, if time is not a limiting factor, or more computational power is available, then further research should be done on the performance of the two solvers.

5.3 Turbulence model comparisons

This section compares the simulation results from the three different turbulence models against experimental data from Kumara (2010). The simulations are performed using the interFoam solver. The choice to proceed with interFoam was made due to its reduced simulation time compared to multiphaseInterFoam. The data for the realizable k-ε turbulence model is discussed in its own section as the results showed notable deviations compared to the other two models.

5.3.1 Realizable k-ε

CFD prediction of mean axial velocity, water volume fraction and turbulent kinetic energy are compared against experimental data using the realizable k-ε model at λ_w=0.25 in Figures 5.7 and 5.8. As seen in Figure 5.7 the model slightly overpredicts the position of the oil-water interface. However, it severely overpredicts the oil phase velocity as well as underpredicting the water phase velocity. The model also fails to capture the correct shape of the velocity field in both phases.

The reason for the inaccuracies is not immediately apparent. One potential explanation could be attributed to the C_μ formulation in Equation 3.34 applied by the realizable k-ε model. For this system the model underpredicts the turbulent kinetic energy in the near-wall areas, as seen in Figure 5.8, resulting in a lower turbulent viscosity from the relationship in Equation 3.38. This affects the effective kinematic viscosity for the oil phase that leads to reduced flow resistance and consequently, increased flow velocity. This could account for the significant overprediction of velocity in the oil phase.

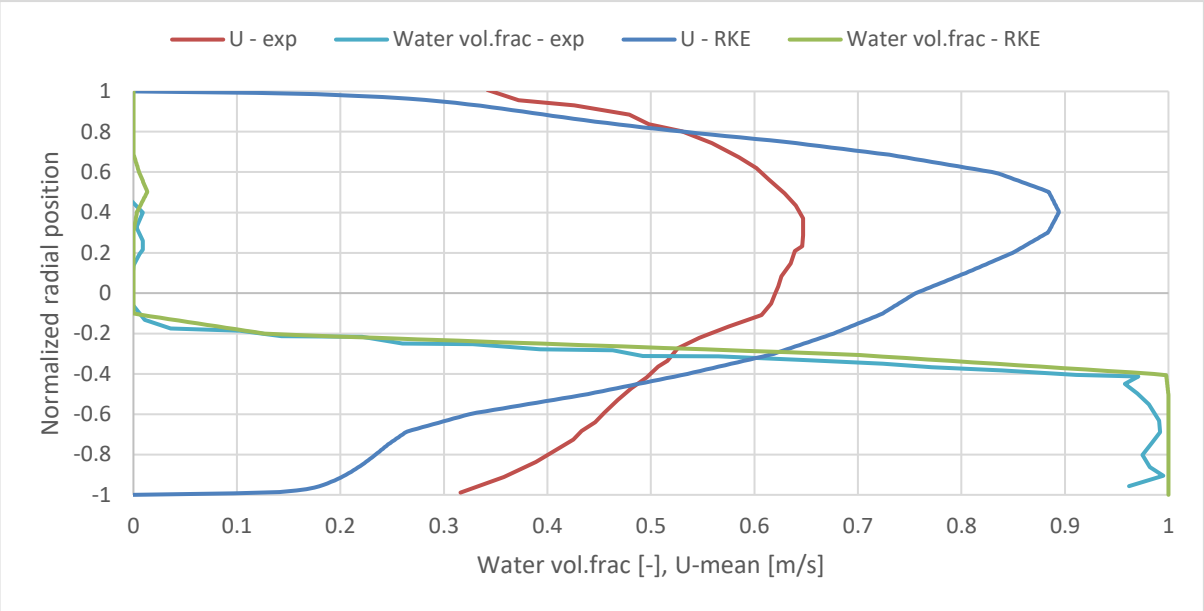

Figure 5.7 - Realizable k-ε turbulence model compared to experimental data at 0.50 m/s mixture velocity and λ_w=0.25.

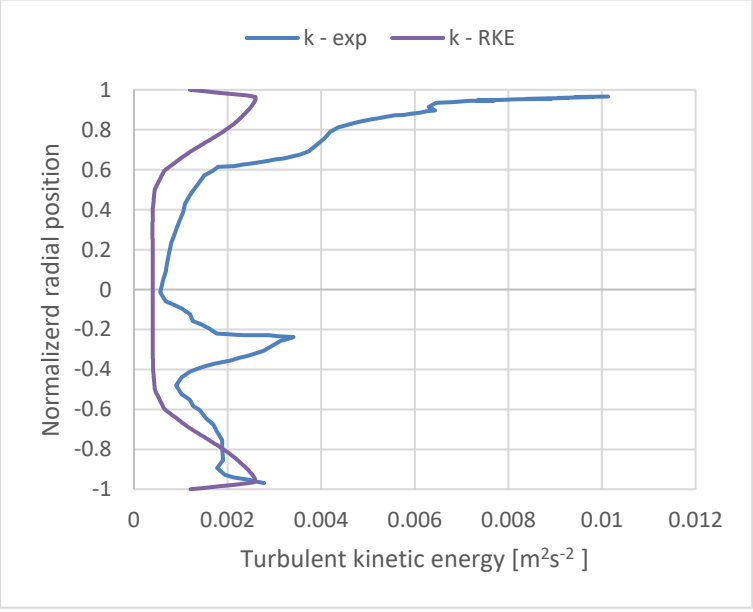

Figure 5.8 - Axial turbulent kinetic energy at 0.50 m/s mixture velocity and λ_w=0.25.

Figure 5.9 shows the simulated and experimental data for mean axial velocity and water volume fraction at λ_w=0.50 compared to experimental data. The data show that the model slightly underpredicts the position of the oil-water interface. Under these conditions, the predicted mean axial velocity is symmetrical compared to the asymmetrical experimental data profile. The highest mean axial velocity is observed at radial position 0.5 in the oil phase. The model accurately predicts the highest velocity but incorrectly predicts it at the oil-water interface. Thus, the model underpredicts the general velocity in the oil-phase. This is

unsurprising as at λ_w=0.50, the initial velocities are equal which induces a "symmetrical start". As discussed in Kumara (2010), an oil-water flow is highly anisotropic meaning that the velocity components and their derivatives are dependent on direction. The oil-water interface acts like a moving wall due to the stable density stratification at the interface. This has a damping effect, causing the axial velocity in the higher velocity phase to slow down as it approaches the interface. This is the reason for the asymmetrical velocity profiles seen in the experimental data. Hence, it's evident that a two-equation model like the realizable k-ε, is insensitive in capturing these effects without incorporating interface treatments, like a turbulence damping function at the oil-water interface.

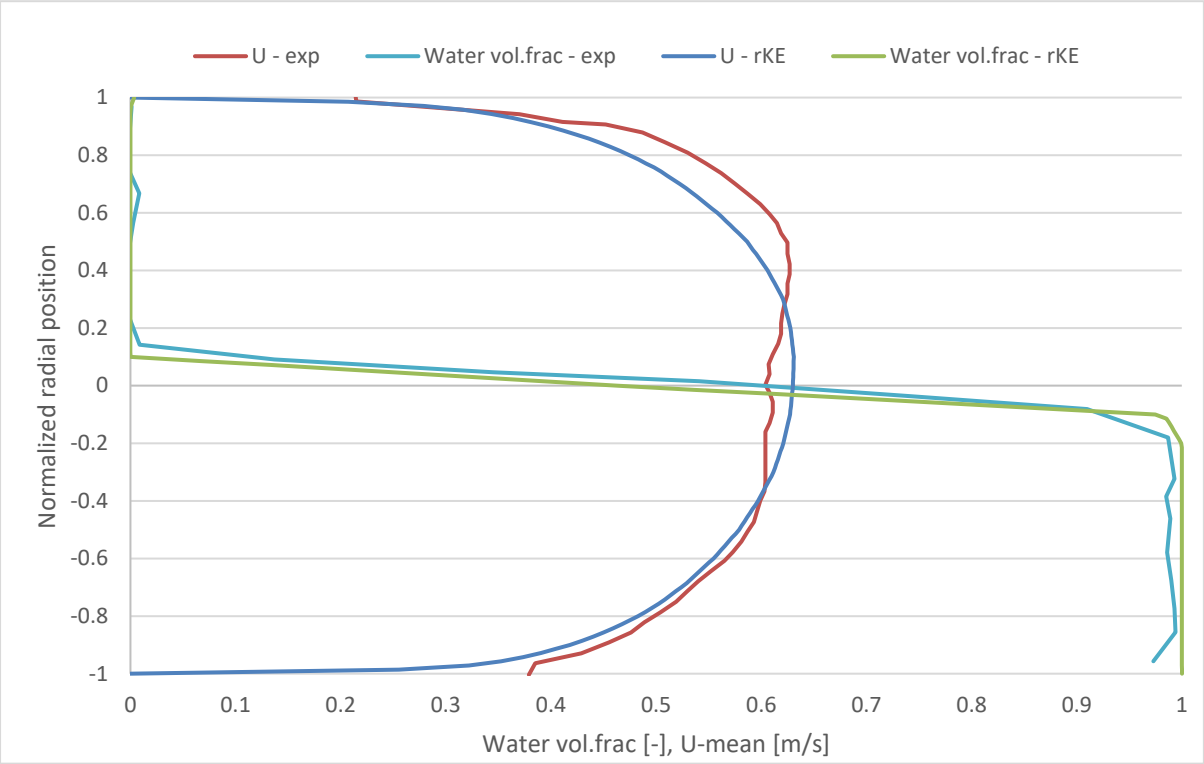

Figure 5.9 - Realizable k-ε turbulence model compared to experimental data at 0.50 m/s mixture velocity and λw=0.50.

Figures 5.10 and 5.11 presents the simulated and experimental data for mean axial velocity, water volume fraction and turbulent kinetic energy at λ_w=0.75. As with λ_w=0.25, the predicted data show major deviations compared to the experimental data. Specifically, the model underpredicts the position of the oil-water interface with deviations between radial position 0.55 and 0.35. While the model accurately predicts the overall shape of the mean axial velocity, it grossly overpredicts the velocity. Figure 5.11 shows that the model is incapable of accurately predicting the turbulent kinetic energy and shows a major underprediction, especially in the oil-phase. Furthermore, the model is insensitive to the fluctuations in the turbulent kinetic energy that occurs in the oil-phase. A cause for the deviations could stem from the mesh. The calculated y+ values for the simulation in Figure 5.12 indicate that the mesh used is not optimized for the realizable k-ε model. With an average y+ value of 3, the mesh is too fine to utilize wall functions, and too coarse to ensure full resolution at the walls without wall functions. Additionally, the k-ε models don't perform well at lower Reynolds numbers which is why wall functions are utilized. Ideally, the y+

should be ~30 when using wall functions and $y+\leq 1$ for full boundary layer resolution. This adds to the possibility that the mesh is too coarse for the realizable k-ε model to calculate the viscous sub-layers correctly. Hence, the mesh should be optimized for wall functions when using this model so that the y+ value is closer to 30.

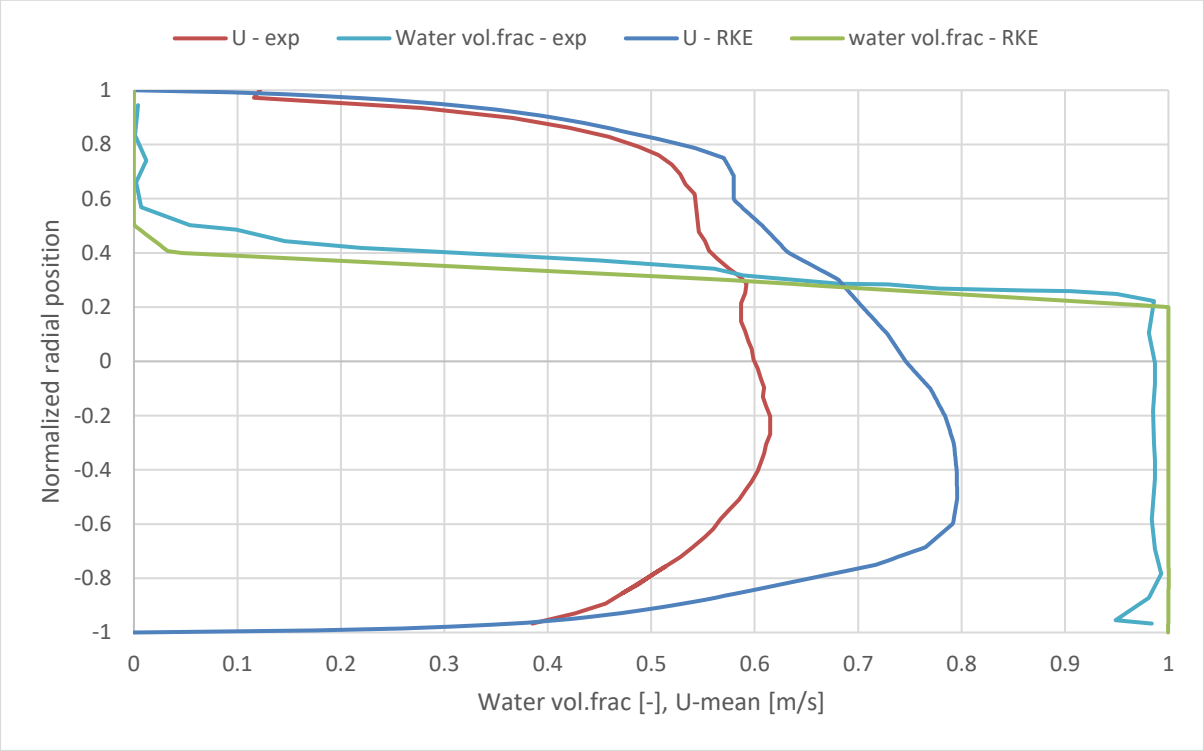

Figure 5.10 - Realizable k-ε turbulence model compared to experimental data at 0.50 m/s mixture velocity and λw=0.75.

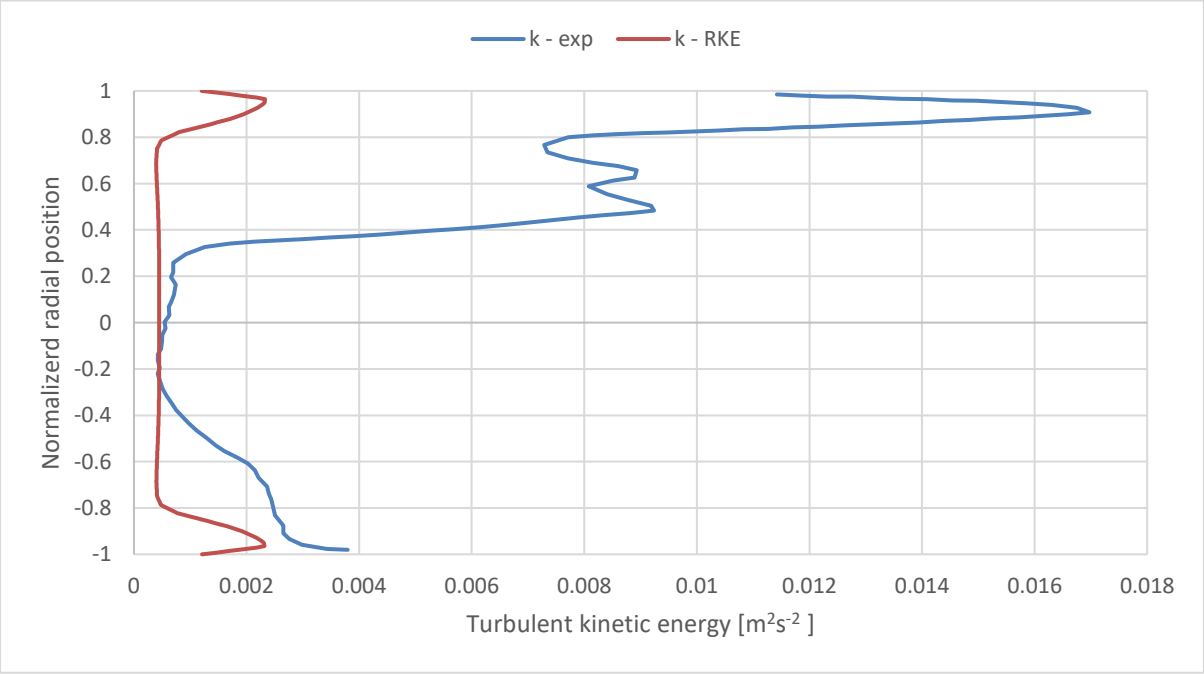

Figure 5.11 - Axial turbulent kinetic energy at 0.50 m/s mixture velocity and λw=0.75.

```
writing object yPlus
patch Wall y+ : min = 0.0410892, max = 5.30664, average = 3.05826
```

Figure 5.12 – y+ values for the realizable k-ε simulation at λ_w=0.75.

The results obtained with the realizable k-ε turbulence model demonstrate that it cannot accurately predict the complex flow behavior of oil-water systems without additional optimization. The model fails to predict the anisotropic behavior of the flow and should be coupled with turbulence damping, or some other form of interface treatment to increase the accuracy. There is a possibility that the model could've produced more accurate results with a different mesh size, but it is difficult to predict by how much. Furthermore, the simulation time posed a significant challenge when utilizing the realizable k-ε model. The simulations at λ_w=0.25 and 0.75 did not reach a steady state until approximately 400 seconds of simulation time, which equated to around 200 real-time hours. Therefore, the use of the realizable k-ε turbulence model for oil-water flow is not recommended at this point.

5.3.2 Mixture velocity 0.50 m/s

5.3.2.1 0.25 water cut

Simulation results using the SST k-ω and RNG k-ε turbulence model at λ_w=0.25 are compared to experimental data in Figures 5.13 and 5.14. The results show that both turbulence models underpredict the position of the oil-water interface but the SST k-ω model show the better accuracy of the two. The SST k-ω model also performs the best out of the two in predicting the mean axial velocity. The SST k-ω closely matches the experimental data until radial position 0.6, after which its accuracy decreases. The accuracy in the near-wall area suggests that the boundary layers are properly resolved. Beyond this point, the SST k-ω model begins to underpredict the velocity until approximately radial position -0.15. After this point it starts to overpredict the velocity and shows insensitivity to changes in the velocity field. In contrast, the RNG k-ε model predicts a symmetrical velocity profile and shows poor accuracy in the near-wall areas. This could indicate that the wall functions do not properly resolve the boundary layers.

The poor accuracy of the models is evident when examining Figure 5.14, where neither model manages to accurately predict the turbulent kinetic energy, especially the fluctuations around the oil-water interface. Both models overpredict the turbulent kinetic energy between radial position 0.6 and -1. This is also the same area where the models are most inaccurate in terms of axial velocity. The turbulent kinetic energy increases around the interface but the velocity decreases for the oil-phase and increases for the water phase. This indicates that due to viscous effects at the interface, the turbulence dissipation rate increases in the oil-phase and decreases for the water phase. This leads to a damping effect in the oil-phase and the opposite for the water phase. Neither model captures this turbulence damping effect at the oil-water interface. The results suggest that the models are incapable of predicting the anisotropic behavior of the oil-water flow and the wall-like effect of the oil-water interface.

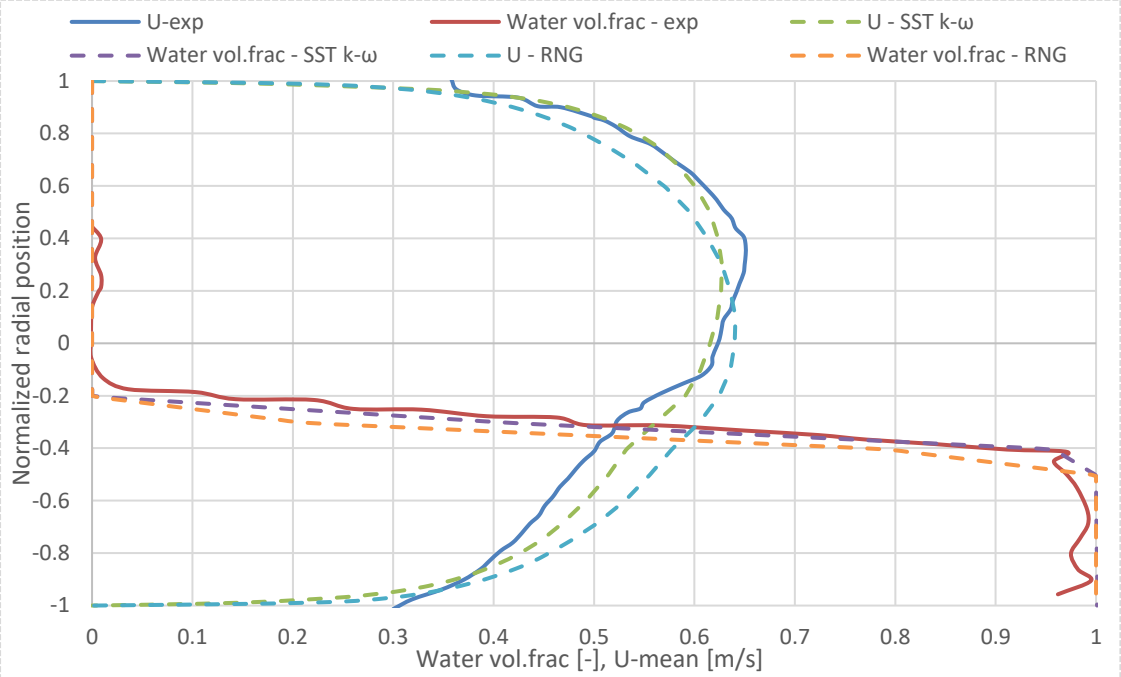

Figure 5.13 – Axial mean velocity and water volume fraction comparison of experimental results and turbulence models at λ_w=0.25.

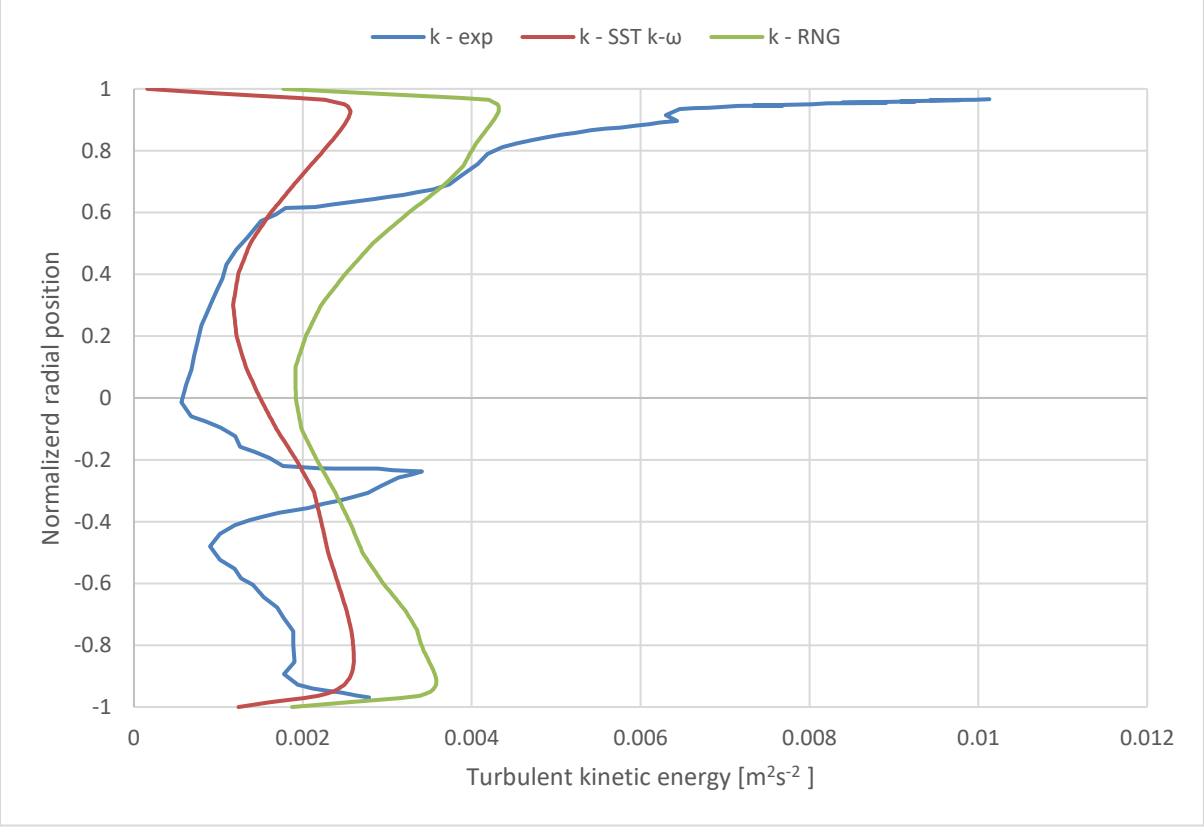

Figure 5.14 – Axial turbulent kinetic energy comparison for experimental data and turbulence models at λ_w=0.25.

5.3.2.2 0.50 water cut

Simulation results using the SST k-ω and RNG k-ε turbulence model at λ_w=0.50 are compared to experimental data in Figures 5.15 and 5.16. Both models are in good agreement with the experimental data regarding the position of the oil-water interface. However, neither model accurately predicts the axial mean velocity. Again, both models fail to capture the fluctuations in the turbulent kinetic energy, which is reflected in their smooth velocity profiles. The RNG k-ε model correctly predicts the highest velocity but predicts it at the oil-water interface, instead of the oil phase. Furthermore, the RNG k-ε model severely underpredicts the oil-phase velocity but show a good agreement in the water phase. In contrast, the SST k-ω model show better agreement in the oil-phase but has larger deviations in the water phase.

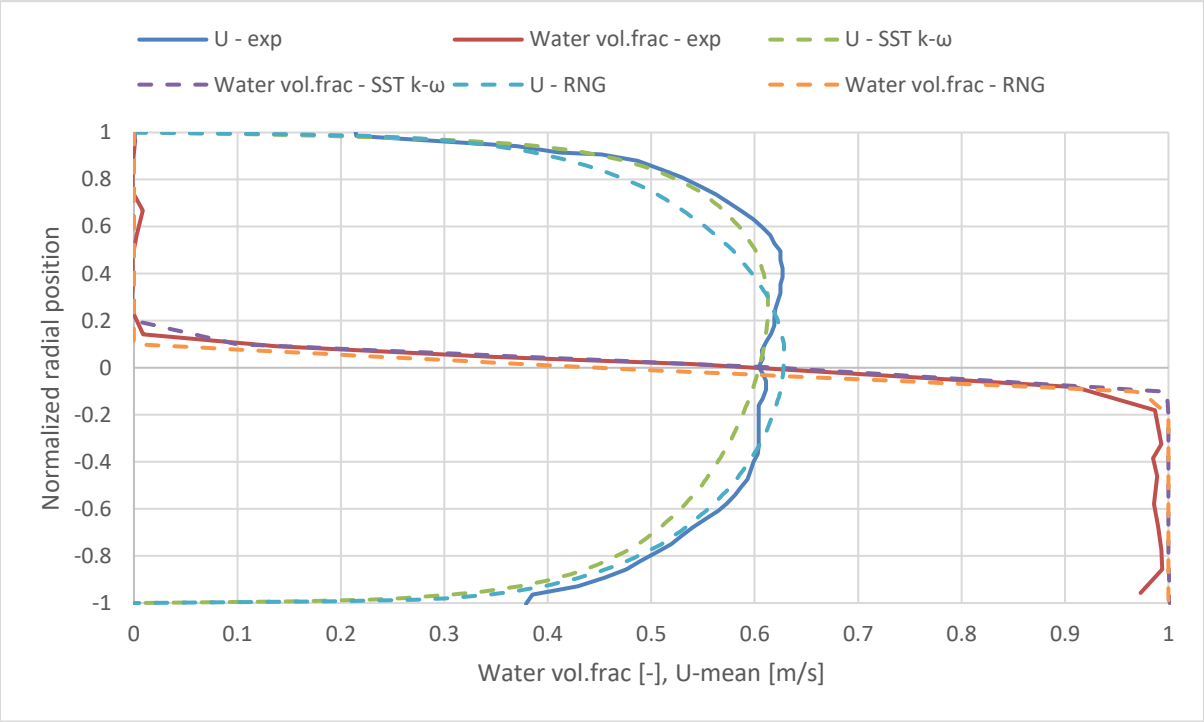

Figure 5.15 - Axial mean velocity and water volume fraction comparison of experimental results and turbulence models at λ_w=0.50.

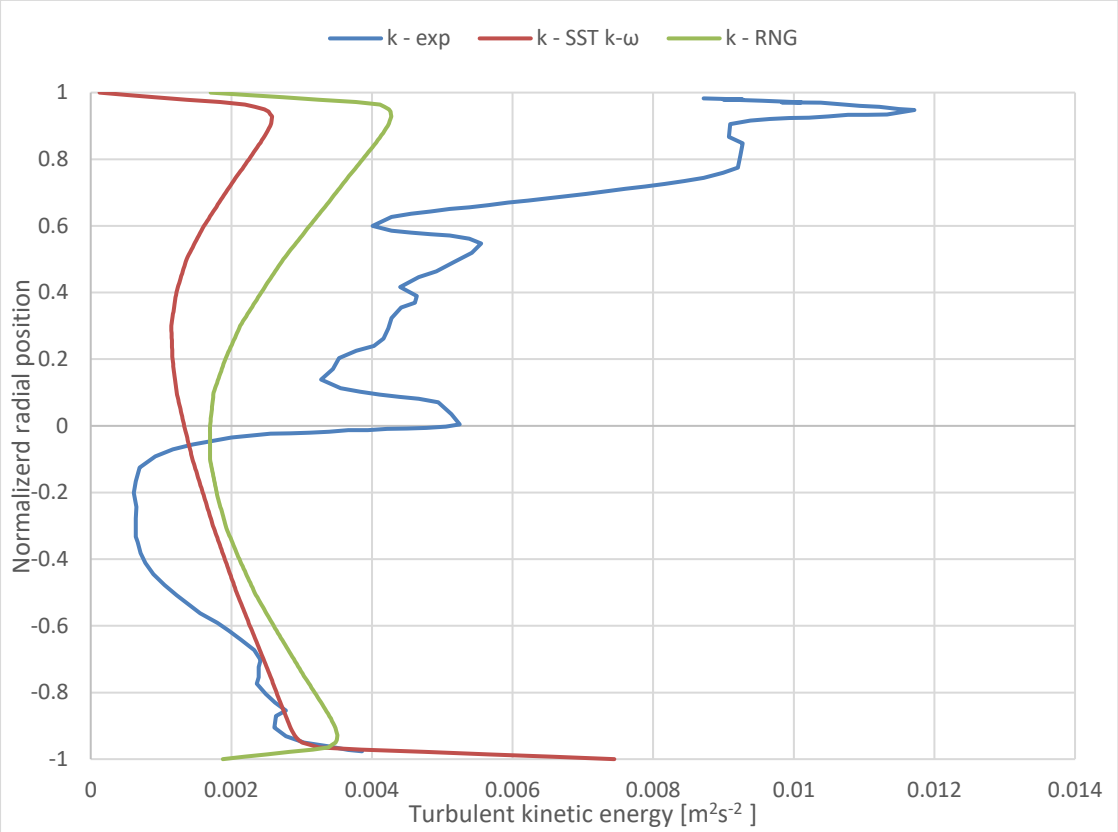

Figure 5.16 - Axial turbulent kinetic energy comparison for experimental data and turbulence models at λ_w=0.25.

5.3.2.3 0.75 water cut

Figures 5.17 and 5.18 display the predicted mean axial velocity, water volume fraction and turbulent kinetic energy data compared to experimental data. Both models closely predict the position of the oil-water interface, albeit with a slight under- and overprediction. However, notable deviations are seen in the predicted mean axial velocity. Neither model is able to capture the damping effect of the oil-water interface, which reflects in the overprediction of the velocity around the interface. It also means that neither model predicts the fluctuations in turbulent kinetic energy which are seen in Figure 5.18. Thus, producing the smooth velocity profiles. The SST k-ω model show the best accuracy in the oil phase, while the RNG k-ε is the more accurate model for the water phase.

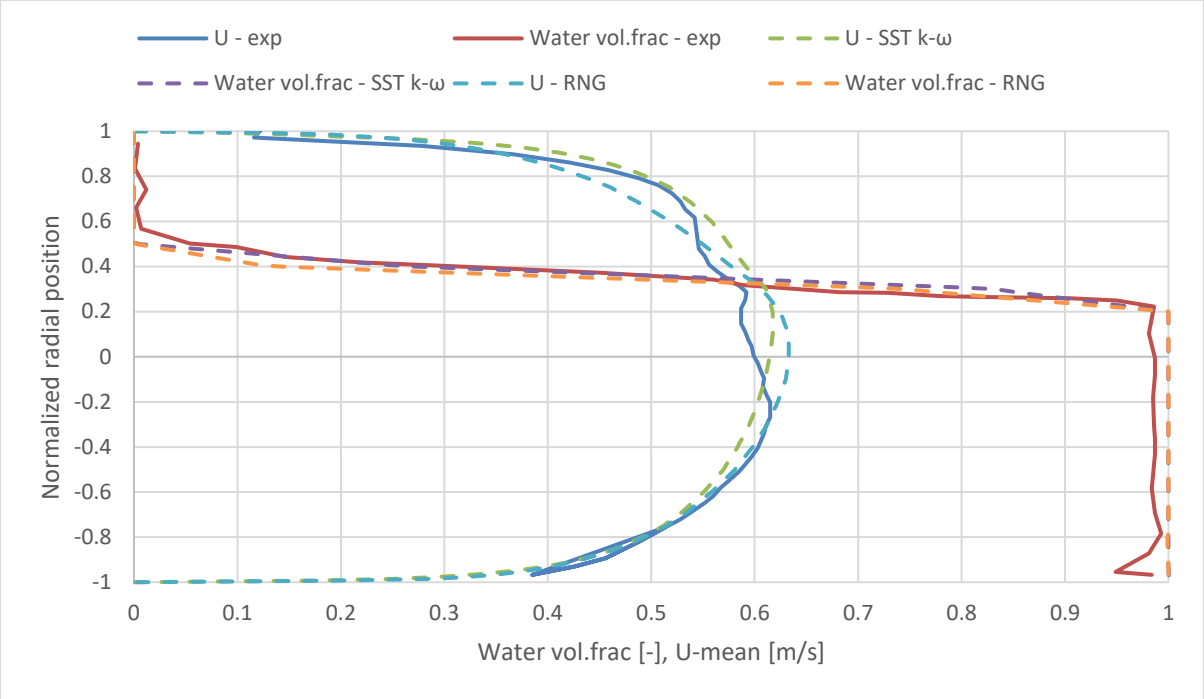

Figure 5.17 - Axial mean velocity and water volume fraction comparison of experimental results and turbulence models at λ_w=0.75.

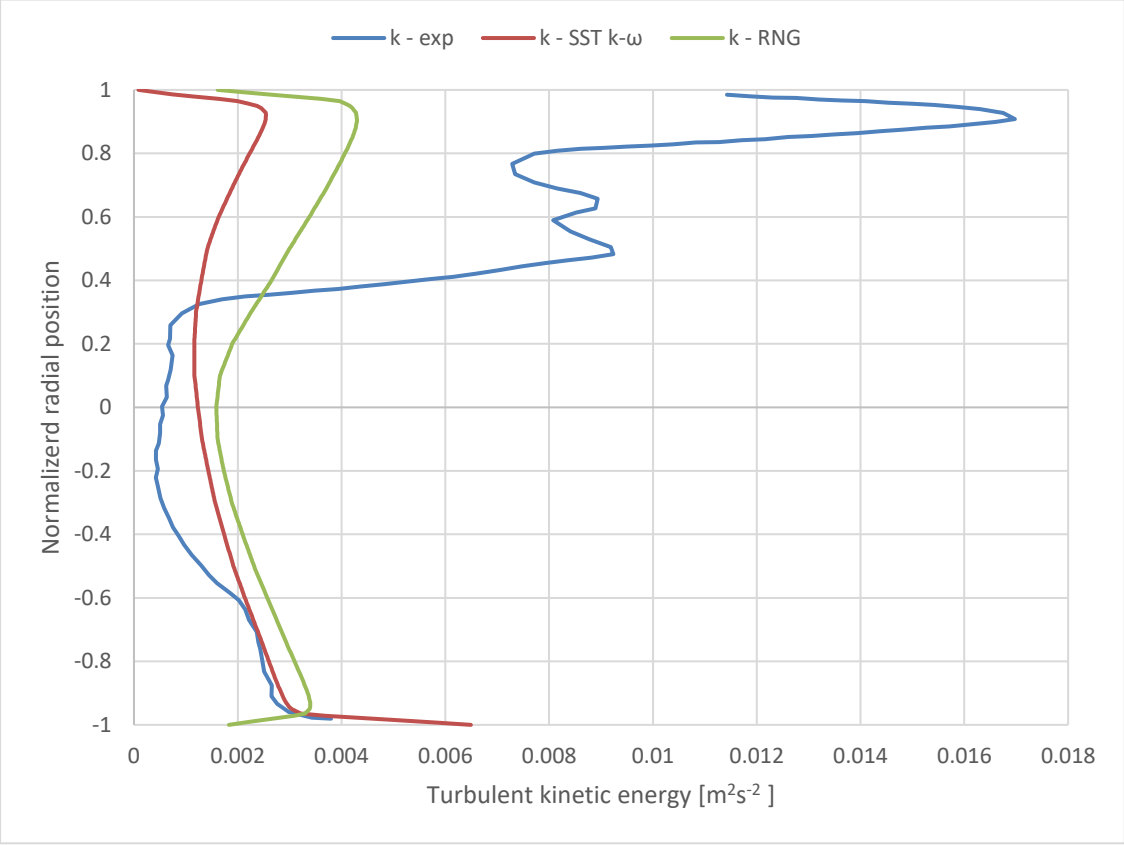

Figure 5.18 - Axial turbulent kinetic energy comparison for experimental data and turbulence models for λ_w=0.25.

5.3.3 Mixture velocity 0.68 m/s

Figures 5.19 and 5.20 compares the experimental data and predicted data for mean axial velocity and water volume fraction data at mixture velocity 0.68 m/s, for λ_w=0.25 and 0.50. Both turbulence models show significant deficiencies in predicting the oil-water interface at this mixture velocity. The models underpredict the position with a large margin, with the largest deviation observed at λ_w=0.25. The same trend is seen for the mean axial velocity. In Figure 5.19, neither model accurately predicts the highest velocity point in the oil-phase and underpredicts the velocity in the oil phase. Figure 5.20 show that both models underpredict the entire velocity flow field. This insensitivity comes from the turbulence model's inability to predict the anisotropic behavior. This issue is increased at U_m=0.68 m/s since the flow is more turbulent compared to U_m=0.50 m/s. Additionally, with a higher mixture velocity, a higher production of droplets is observed at the interface. This effect is seen from Figures 5.21 and 5.22. The increased droplet production affects the model's ability to accurately predict the position of the oil-water interface. This deviation could be caused by the mesh being too coarse to fully capture the droplets. The deviation could also stem from the droplets interfering with the surface tracking employed by interFoam. Furthermore, the droplets will influence the anisotropic behavior in the turbulence field at the interface and affect the local viscosity distribution. Hence, the increased droplet formation significantly decreases the accuracy of the turbulence models.

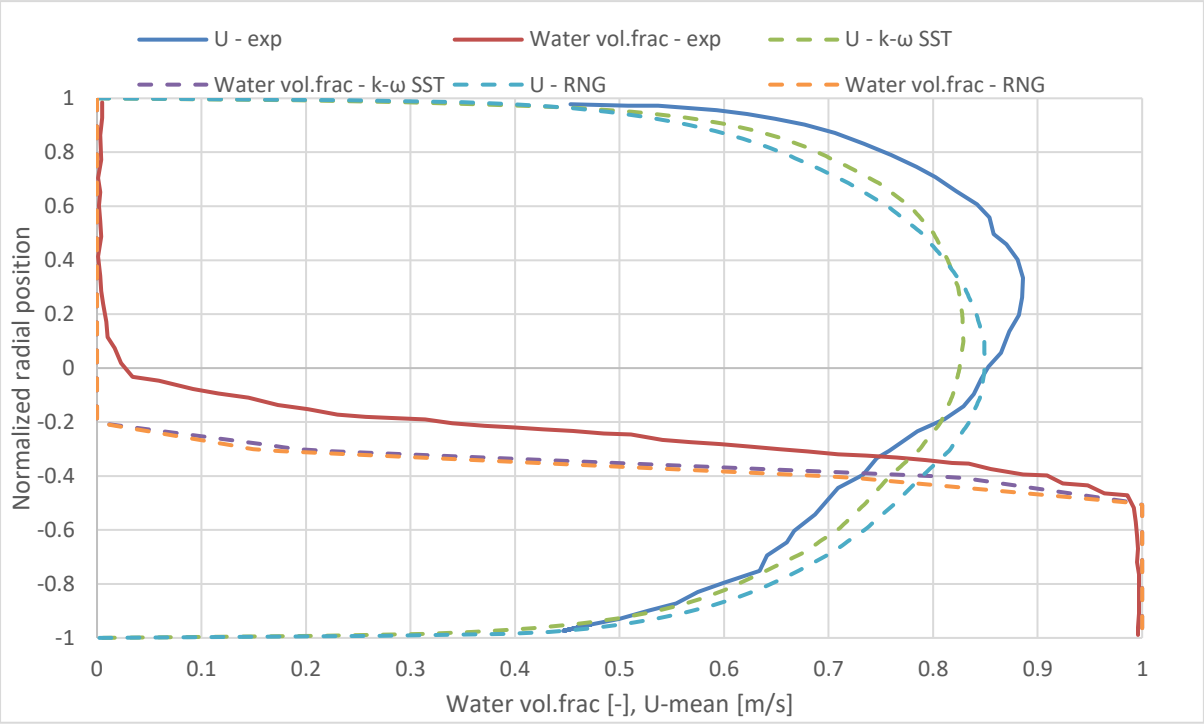

Figure 5.19 - Axial mean velocity and water volume fraction comparison of experimental results and turbulence models at λw=0.25.

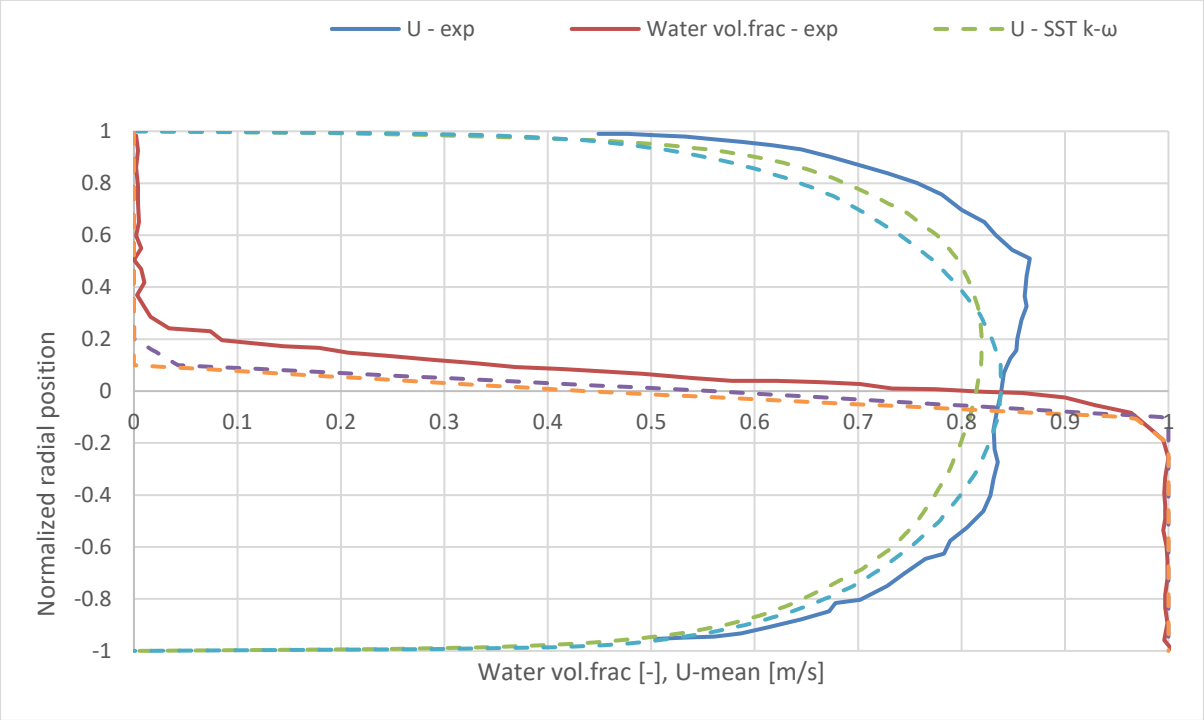

Figure 5.20 - Axial mean velocity and water volume fraction comparison of experimental results and turbulence models at λ_w=0.50.

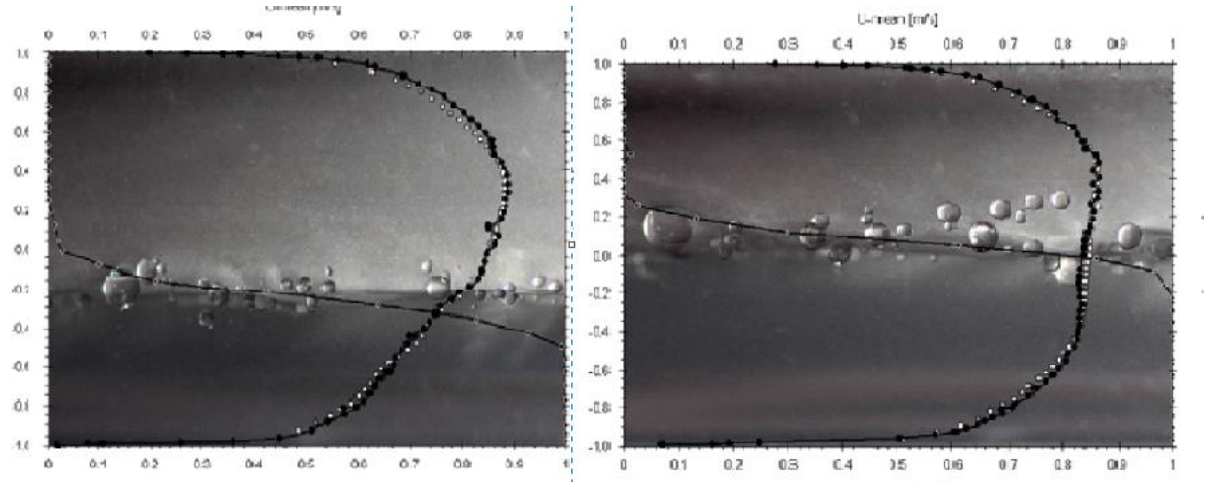

Figure 5.21 – Droplet formation at mixture velocity 0.68 m/s for (a) λ_w=0.25 and (b) λ_w=0.50 (Kumara et al., 2010a).

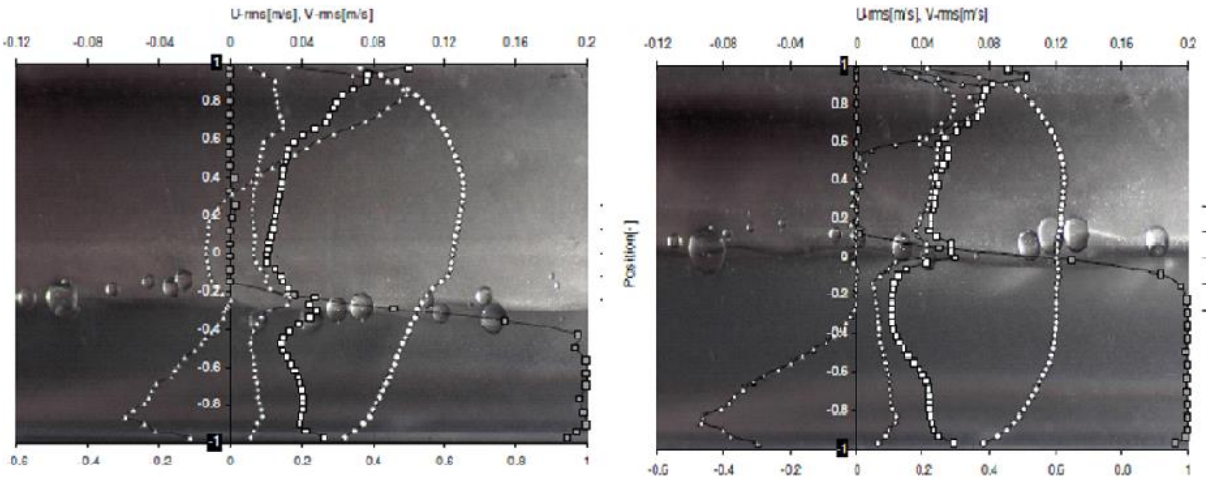

Figure 5.22 – Droplet formation at mixture velocity 0.50 m/s for (a) λ_w=0.25 and (b) λ_w=0.50 (Kumara et al., 2010a).

Overall, the results indicate that the two-equation turbulence models utilized in this study are incapable of accurately predicting the complexity of oil-water flows. Without additional interface treatments, such as turbulence damping, the models are insensitive to the anisotropic behavior of the flow. Additionally, possible challenges related to mesh resolution regarding droplet formation were discovered. Nevertheless, some noteworthy observations emerged from the study. The results in this thesis mirror some of the same velocity field characteristics as discussed in Liu et al. (2022). Specifically, the position of the peak velocity of the dominant phase, is followed by a downward shift in velocity. This effect is likely due to the oil-water interface acting like a moving wall, which causes less flow resistance than near the pipe wall. The peak velocity for the non-dominant phase is located at the oil-water interface. This suggest that the oil-water interface does in fact behave like a wall for the dominant phase, whereas for the non-dominant phase, the interface acts as a force that drags the flow forwards.

6 Conclusion & future work

In this section the conclusions made based on the literature review and comparing the CFD study to experimental data are summarized. Additionally, recommendations for future work are proposed.

6.1 Conclusion

A review of recent advances in oil-water modelling which includes an introduction to basic terminologies in oil-water flow have been presented. In addition to the review, a stratified oil-water flow is numerically simulated using the CFD software OpenFOAM 9. The results produced by the simulations are compared to the experimental data from Kumara (2010).

The review indicated that the numerical modelling of oil-water flows has shown good progress in developing accurate models. However, the review also highlighted areas in the numerical models that require additional attention. The studies presented suggests that the oil-water interface requires additional modelling treatment alongside the chosen turbulence model. What specific treatment that must be implemented will be different for each model and must be customized for the specific system.

The developed CFD model utilized the VOF solver interFoam and employed three different turbulence models: RNG k-ε, realizable k-ε and SST k-ω. The prediction of mean axial velocity, water volume fraction and turbulent kinetic energy were compared to experimental data. Large deviations were seen which highlighted the limitations of the two-equation turbulence models in accurately capturing the complex behavior of the oil-water flow. It is concluded that the turbulence models are unable to capture the anisotropic behavior of the flow and that the flow requires additional treatment to mimic the characteristics of the oil-water interface. The results show that the oil-water interface should be treated as a moving wall and therefore a turbulence damping scheme should be applied to account for the wall-like effect of the interface. The SST k-ω turbulence model performed the best out of the three models used and should be used for this kind of flow problem.

Simulations done at a mixture velocity of 0.68 m/s indicated that the mesh was potentially too coarse to accurately capture the effects of the increased droplet formation at the oil-water interface. Additionally, uncertainties around the surface tracking capabilities of interFoam at this mixture velocity was found. An increased mixture velocity further highlighted the discrepancies between simulated and experimental data.

6.2 Future work

Recommendations for continued research from this thesis should include:

- A larger mesh sensitivity study should be conducted. The generated mesh needs be optimized for the turbulence model chosen. Additionally, the mesh should consider droplet formations if mixture velocities exceeding 0.50 m/s are used.
- Research turbulence damping schemes for multiphase flows and implement it at the oil-water interface.

- Continue the numerical simulations using the SST k-ω turbulence model with an optimized mesh and turbulence damping scheme applied.
- The customized turbulence model discussed in chapter. 4.4 should be developed further. If successfully implemented, extensive testing should be performed.
- Investigate the possibilities of utilizing different solvers and turbulence models.

References

Abubakar, A., Al-Wahaibi, T., Al-Hashmi, A.R., Al-Wahaibi, Y., Al-Ajmi, A., Eshrati, M., 2016. Empirical correlation for predicting pressure gradients of oil-water flow with drag-reducing polymer. Experimental Thermal and Fluid Science 79, 275–282. https://doi.org/10.1016/j.expthermflusci.2016.07.023

Adaze, E., Al-Sarkhi, A., Badr, H.M., Elsaadawy, E., 2019. Current status of CFD modeling of liquid loading phenomena in gas wells: a literature review. J Petrol Explor Prod Technol 9, 1397–1411. https://doi.org/10.1007/s13202-018-0534-4

Ahmed, S.A., John, B., 2018. Liquid – Liquid horizontal pipe flow – A review. Journal of Petroleum Science and Engineering 168, 426–447. https://doi.org/10.1016/j.petrol.2018.04.012

Alias, A., Koto, J., Ahmed, Y., 2015. CFD Simulation for Stratified Oil-Water Two-Phase Flow in a Horizontal Pipe. Journal of Subsea and Offshore -Science and Engineering- 2, 1–6.

Al-Wahaibi, T., 2012. Pressure gradient correlation for oil–water separated flow in horizontal pipes. Experimental Thermal and Fluid Science 42, 196–203. https://doi.org/10.1016/j.expthermflusci.2012.04.021

Amundsen, L., 2011. An experimental study of oil-water flow in horizontal and inclined pipes (Doctoral thesis).

Angeli, P., Hewitt, G.F., 2000. Drop size distributions in horizontal oil-water dispersed flows. Chemical Engineering Science 55, 3133–3143. https://doi.org/10.1016/S0009-2509(99)00585-0

Angeli, P., Hewitt, G.F., 1999. Pressure gradient in horizontal liquid–liquid flows. International Journal of Multiphase Flow 24, 1183–1203. https://doi.org/10.1016/S0301-9322(98)00006-8

ANSYS FLUENT 12.0 Theory Guide [WWW Document], n.d. . ANSYS FLUENT 12.0 Theory Guide. URL https://www.afs.enea.it/project/neptunius/docs/fluent/html/th/main_pre.htm (accessed 2.19.24).

Archibong-Eso, A., Shi, J., Baba, Y.D., Aliyu, A.M., Raji, Y.O., Yeung, H., 2019. High viscous oil–water two–phase flow: experiments & numerical simulations. Heat Mass Transfer 55, 755–767. https://doi.org/10.1007/s00231-018-2461-9

Asyikin, M.T., 2012. CFD Simulation of Vortex Induced Vibration of a Cylindrical Structure (Master thesis). 82. Institutt for bygg, anlegg og transport.

Barnea, D., Taitel, Y., 1992. Structural and interfacial stability of multiple solutions for stratified flow. International Journal of Multiphase Flow 18, 821–830. https://doi.org/10.1016/0301-9322(92)90061-K

Beggs, D.H., Brill, J.P., 1973. A Study of Two-Phase Flow in Inclined Pipes. Journal of Petroleum Technology 25, 607–617. https://doi.org/10.2118/4007-PA

Blakeslee, M., n.d. Wilcox k-ω Model [WWW Document]. URL https://2021.help.altair.com/2021/hwsolvers/acusolve/topics/acusolve/training_manual/wilcox_k_model_r.htm (accessed 3.13.24).

Brackbill, J., Kothe, D., CA, Z., 1992. A Continuum Method for Modeling Surface Tension. Journal of Computational Physics 100. https://doi.org/10.1016/0021-9991(92)90240-Y

Brauner, N., 2003. Liquid-Liquid Two-Phase Flow Systems, in: Bertola, V. (Ed.), Modelling and Experimentation in Two-Phase Flow, International Centre for Mechanical Sciences. Springer, Vienna, pp. 221–279. https://doi.org/10.1007/978-3-7091-2538-0_5

Burlutskii, E., 2018. CFD study of oil-in-water two-phase flow in horizontal and vertical pipes. Journal of Petroleum Science and Engineering 162, 524–531. https://doi.org/10.1016/j.petrol.2017.10.035

C. K. G. Lam, K. Bremhorst, 1982. A Modified Form of the k-e Model for Predicting Wall Turbulence 5.

C. W. Hirt, B. D. Nichols, 1979. Volume of Fluid (VOF) Method for Dynamics of Free Boundaries.

CAD Exchanger SDK: Computational meshers [WWW Document], n.d. URL https://docs.cadexchanger.com/sdk/sdk_meshing_advalgos_usage_page (accessed 4.20.24).

Charles, M.E., Govier, G.W., Hodgson, G.W., 1961. The horizontal pipeline flow of equal density oil-water mixtures. The Canadian Journal of Chemical Engineering 39, 27–36. https://doi.org/10.1002/cjce.5450390106

Chen, J., Anastasiou, C., Cheng, S., Basha, N.M., Kahouadji, L., Arcucci, R., Angeli, P., Matar, O.K., 2023. Computational fluid dynamics simulations of phase separation in dispersed oil-water pipe flows. Chemical Engineering Science 267, 118310. https://doi.org/10.1016/j.ces.2022.118310

Cheng, J., Li, Q., Yang, C., Zhang, Y., Mao, Z., 2018. CFD-PBE simulation of a bubble column in OpenFOAM. Chinese Journal of Chemical Engineering 26, 1773–1784. https://doi.org/10.1016/j.cjche.2017.11.012

David C. Wilcox, 2006. Turbulence modeling for CFD, 3rd ed. DCW Industries.

Davidson, L., 2022. An Introduction to Turbulence Models.

De la Cruz-Ávila, M., Carvajal-Mariscal, I., Sigalotti, L.D.G., Klapp, J., 2022. Numerical Study of Water-Oil Two-Phase Flow Evolution in a Y-Junction Horizontal Pipeline. Water 14, 3451. https://doi.org/10.3390/w14213451

Deen, N.G., Solberg, T., Hjertager, B.H., 2001. Large eddy simulation of the Gas–Liquid flow in a square cross-sectioned bubble column. Chemical Engineering Science, Proceedings of the 5th International Conference on Gas-Liquid and Gas-Liquid-Solid Reactor Engineering 56, 6341–6349. https://doi.org/10.1016/S0009-2509(01)00249-4

Duan, J., Gong, J., Yao, H., Deng, T., Zhou, J., 2014. Numerical modeling for stratified gas–liquid flow and heat transfer in pipeline. Applied Energy 115, 83–94. https://doi.org/10.1016/j.apenergy.2013.10.050

Duan, J., Liu, H., Wang, N., Gong, J., Jiao, G., 2015. Hydro dynamic modeling of stratified smooth two-phase turbulent flow with curved interface through circular pipe. International Journal of Heat and Mass Transfer 89, 1034–1043. https://doi.org/10.1016/j.ijheatmasstransfer.2015.05.093

Edomwonyi-Otu, L.C., Angeli, P., 2015. Pressure drop and holdup predictions in horizontal oil–water flows for curved and wavy interfaces. Chemical Engineering Research and Design 93, 55–65. https://doi.org/10.1016/j.cherd.2014.06.009

Elseth, G., 2001. An Experimental Study of Oil/Water Flow in Horizontal Pipes (Doctoral thesis). 270. Fakultet for ingeniørvitenskap og teknologi.

Fan, W., Anglart, H., 2020. varRhoTurbVOF: A new set of volume of fluid solvers for turbulent isothermal multiphase flows in OpenFOAM. Computer Physics Communications 247, 106876. https://doi.org/10.1016/j.cpc.2019.106876

Fan, W., Anglart, H., 2019. Progress in Phenomenological Modeling of Turbulence Damping around a Two-Phase Interface. Fluids 4, 136. https://doi.org/10.3390/fluids4030136

Frank M. White, 1991. VISCOUS FLUID FLOW, 2nd ed. McGraw-Hill.

Gada, V.H., Sharma, A., 2012. Analytical and level-set method based numerical study on oil–water smooth/wavy stratified-flow in an inclined plane-channel. International Journal of Multiphase Flow 38, 99–117. https://doi.org/10.1016/j.ijmultiphaseflow.2011.08.015

Gao, H., Gu, H.-Y., Guo, L.-J., 2003. Numerical study of stratified oil–water two-phase turbulent flow in a horizontal tube. International Journal of Heat and Mass Transfer 46, 749–754. https://doi.org/10.1016/S0017-9310(02)00321-6

Govier, G.W., Omer, M.M., 1962. The horizontal pipeline flow of air-water mixtures. The Canadian Journal of Chemical Engineering 40, 93–104. https://doi.org/10.1002/cjce.5450400303

Greenshields, C.J., 2023. OpenFOAM handbook.

Han, H., 2005. A study of entrainment in two-phase upward cocurrent annular flow in a vertical tube. University of Saskatchewan.

Hapanowicz, J., 2010. Phase inversion in liquid–liquid pipe flow. Flow Measurement and Instrumentation, Special Issue: Validation and Data Fusion for Process Tomographic Flow Measurements 21, 284–291. https://doi.org/10.1016/j.flowmeasinst.2010.03.001

Hibiki, T., Rassame, S., 2019. Analytical model for predicting oil fraction in horizontal oil–water two-phase flow. Exp. Comput. Multiph. Flow 1, 73–84. https://doi.org/10.1007/s42757-019-0013-2

H.K. Versteeg, W. Malalasekera, 2007. An introduction to Computational Fluid Dynamics, second edition. ed, The Finite Volume Method. Pearson.

Hu, H.H., Patankar, N.A., Zhu, M.Y., 2001. Direct Numerical Simulations of Fluid–Solid Systems Using the Arbitrary Lagrangian–Eulerian Technique. Journal of Computational Physics 169, 427–462. https://doi.org/10.1006/jcph.2000.6592

Ibarra, R., Matar, O., Markides, C., Zadrazil, I., 2015. An experimental study of oil-water flows in horizontal pipes. Presented at the 17th International Conference on Multiphase Technology, Cannes, France.

Ibarra, R., Zadrazil, I., Matar, O.K., Markides, C.N., 2018. Dynamics of liquid–liquid flows in horizontal pipes using simultaneous two–line planar laser–induced fluorescence and particle velocimetry. International Journal of Multiphase Flow 101, 47–63. https://doi.org/10.1016/j.ijmultiphaseflow.2017.12.018

Ishii, M., Zuber, N., 1979. Drag coefficient and relative velocity in bubbly, droplet or particulate flows. AIChE Journal 25, 843–855. https://doi.org/10.1002/aic.690250513

Ismail, A.S.I., Ismail, I., Zoveidavianpoor, M., Mohsin, R., Piroozian, A., Misnan, M.S., Sariman, M.Z., 2015. Review of oil–water through pipes. Flow Measurement and Instrumentation 45, 357–374. https://doi.org/10.1016/j.flowmeasinst.2015.07.015

Kamp, J., Villwock, J., Kraume, M., 2017. Drop coalescence in technical liquid/liquid applications: a review on experimental techniques and modeling approaches. Reviews in Chemical Engineering 33, 1–47. https://doi.org/10.1515/revce-2015-0071

Kang, Q., Gu, J., Qi, X., Wu, T., Wang, S., Chen, S., Wang, W., Gong, J., 2021. Hydrodynamic Modeling of Oil–Water Stratified Smooth Two-Phase Turbulent Flow in Horizontal Circular Pipes. Energies 14, 5201. https://doi.org/10.3390/en14165201

Kumara, W.A.S., Elseth, G., Halvorsen, B.M., Melaaen, M.C., 2010a. Comparison of Particle Image Velocimetry and Laser Doppler Anemometry measurement methods applied to the oil–water flow in horizontal pipe. Flow Measurement and Instrumentation 21, 105–117. https://doi.org/10.1016/j.flowmeasinst.2010.01.005

Kumara, W.A.S., Halvorsen, B.M., Melaaen, M.C., 2010b. Particle image velocimetry for characterizing the flow structure of oil–water flow in horizontal and slightly inclined pipes. Chemical Engineering Science 65, 4332–4349. https://doi.org/10.1016/j.ces.2010.03.045

Kumara, W.A.S., Halvorsen, B.M., Melaaen, M.C., 2009. Pressure drop, flow pattern and local water volume fraction measurements of oil–water flow in pipes. Meas. Sci. Technol. 20, 114004. https://doi.org/10.1088/0957-0233/20/11/114004

Lande, A.M., 2021. Complex Mesh Generation with OpenFOAM. University of South-Eastern Norway.

Launder, B., Spalding, D.B., 1974. The Numerical Computation of Turbulent Flow Computer Methods. Computer Methods in Applied Mechanics and Engineering 3, 269–289. https://doi.org/10.1016/0045-7825(74)90029-2

Lian, J., Yang, X., Ma, B., Gou, W., 2022. A novel method for bounding the phase fractions at both ends in Eulerian multi-fluid model. Computers & Fluids 243, 105512. https://doi.org/10.1016/j.compfluid.2022.105512

Liu, H., Duan, J., Li, J., Gu, K., Lin, K., Wang, J., Yan, H., Guan, L., Li, C., 2022. Numerical quasi-three dimensional modeling of stratified oil-water flow in horizontal circular pipe. Ocean Engineering 251, 111172. https://doi.org/10.1016/j.oceaneng.2022.111172

Lopez de Bertodano, M., Lahey, R.T., Jones, O.C., 1994. Turbulent bubbly two-phase flow data in a triangular duct. Nuclear Engineering and Design 146, 43–52. https://doi.org/10.1016/0029-5493(94)90319-0

Lovick, J., Angeli, P., 2004. Droplet size and velocity profiles in liquid–liquid horizontal flows. Chemical Engineering Science 59, 3105–3115. https://doi.org/10.1016/j.ces.2004.04.035

Lum, J.Y.-L., Al-Wahaibi, T., Angeli, P., 2006. Upward and downward inclination oil–water flows. International Journal of Multiphase Flow 32, 413–435. https://doi.org/10.1016/j.ijmultiphaseflow.2006.01.001

Luo, H., Wen, J., Jiang, R., Shao, Q., Wang, Z., 2022. Modeling of the Phase Inversion Point of Crude Oil Emulsion by Characterization of Crude Oil Physical Properties. ACS Omega 7, 39136–39146. https://doi.org/10.1021/acsomega.2c04989

Medina, H., Beechook, A., Saul, J., Porter, S., Aleksandrova, S., Benjamin, S., 2015. Open source Computational Fluid Dynamics using OpenFOAM. https://doi.org/10.13140/RG.2.1.1930.9843

Menter, F., Kuntz, M., Langtry, R., 2003. Ten years of industrial experience with the SST turbulence model. Heat and Mass Transfer 4.

Menter, F.R., 1994. Two-equation eddy-viscosity turbulence models for engineering applications. AIAA Journal 32, 1598–1605. https://doi.org/10.2514/3.12149

Mesh Quality [WWW Document], n.d. . SimScale. URL https://www.simscale.com/docs/simulation-setup/meshing/mesh-quality/ (accessed 4.20.24).

Mohammaed A. Al-Yaari, Basel F. Abu-Sharkh, 2015. CFD Prediction of Stratified Oil-Water Flow in a Horizontal Pipe. yumpu.com 01.

Multi-phase flow simulations in OpenFOAM [WWW Document], n.d. URL https://www.cfdyna.com/Home/of_multiPhase.html (accessed 2.21.24).

Newton, C.H., Behnia, M., 2000. Numerical calculation of turbulent stratified gas–liquid pipe flows. International Journal of Multiphase Flow 26, 327–337. https://doi.org/10.1016/S0301-9322(99)00010-5

Niotis, A., Vassalos, D., Boulougouris, E., Cichowicz, J., Atzampos, G., Paterson, D., 2019. Verification of damage ship survivability with computational fluid dynamics.

Panagiota Angeli, object, 1996. Liquid-liquid dispersed flows in horizontal pipes. University of London.

Passoni, S., Carraretto, I.M., Mereu, R., Colombo, L.P.M., 2023. Two-phase stratified flow in horizontal pipes: A CFD study to improve prediction of pressure gradient and void fraction. Chemical Engineering Research and Design 191, 38–49. https://doi.org/10.1016/j.cherd.2023.01.016

Pouraria, H., Park, K.-H., Seo, Y., 2021. Numerical Modelling of Dispersed Water in Oil Flows Using Eulerian-Eulerian Approach and Population Balance Model. Processes 9, 1345. https://doi.org/10.3390/pr9081345

Pouraria, H., Seo, J.K., Paik, J.K., 2016. Numerical modelling of two-phase oil–water flow patterns in a subsea pipeline. Ocean Engineering 115, 135–148. https://doi.org/10.1016/j.oceaneng.2016.02.007

Rahman, M., Siikonen, T., 2005. Low Reynolds number k-epsilon model for near-wall flow. International Journal for Numerical Methods in Fluids 47, 325–338. https://doi.org/10.1002/fld.809

Rodriguez, O.M.H., Oliemans, R.V.A., 2006. Experimental study on oil–water flow in horizontal and slightly inclined pipes. International Journal of Multiphase Flow 32, 323–343. https://doi.org/10.1016/j.ijmultiphaseflow.2005.11.001

Santos, D.S., Faia, P.M., Garcia, F.A.P., Rasteiro, M.G., 2019. Oil/water stratified flow in a horizontal pipe: Simulated and experimental studies using EIT. Journal of Petroleum Science and Engineering 174, 1179–1193. https://doi.org/10.1016/j.petrol.2018.12.002

Setaih, K., Mohammed, M.A., Hamza, N., S., D., Townshend, T., 2010. Crafting and Assessing Urban Environments Using Computational Fluid Dynamics. Presented at the ASCAAD, p. 8.

Shi, J., Gourma, M., Yeung, H., 2017. CFD simulation of horizontal oil-water flow with matched density and medium viscosity ratio in different flow regimes. Journal of Petroleum Science and Engineering 151, 373–383. https://doi.org/10.1016/j.petrol.2017.01.022

Shih, T.-H., Liou, W.W., Shabbir, A., Yang, Z., Zhu, J., 1995. A new *k*-ϵ eddy viscosity model for high reynolds number turbulent flows. Computers & Fluids 24, 227–238. https://doi.org/10.1016/0045-7930(94)00032-T

Shuard, A.M., Mahmud, H.B., King, A.J., 2016. Comparison of Two-Phase Pipe Flow in OpenFOAM with a Mechanistic Model. IOP Conf. Ser.: Mater. Sci. Eng. 121, 012018. https://doi.org/10.1088/1757-899X/121/1/012018

Song, X., Li, D., Sun, X., Mou, X., Cheng, Y.F., Yang, Y., 2021. Numerical modeling of the critical pipeline inclination for the elimination of the water accumulation on the pipe floor in oil-water fluid flow. Petroleum 7, 209–221. https://doi.org/10.1016/j.petlm.2020.07.001

Speziale, 1990. Analytical Methods for the Development of Reynolds Stress Closures in Turbulence. INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING HAMPTON 60.

Sun, R., Xiao, H., 2015. Diffusion-based coarse graining in hybrid continuum–discrete solvers: Applications in CFD–DEM. International Journal of Multiphase Flow 72, 233–247. https://doi.org/10.1016/j.ijmultiphaseflow.2015.02.014

Sunday, N., Settar, A., Chetehouna, K., Gascoin, N., 2023. Numerical modeling and parametric sensitivity analysis of heat transfer and two-phase oil and water flow characteristics in horizontal and inclined flowlines using OpenFOAM. Petroleum Science 20, 1183–1199. https://doi.org/10.1016/j.petsci.2022.10.008

Tawekal, J.R., 2015. CFD simulation of the flow over a 2-dimensional pipe and vortex induced vibration of the pipe with 1 degree of freedom (Master thesis). University of Stavanger, Norway.

Trallero, J.L., Sarica, C., Brill, J.P., 1997. A Study of Oil/Water Flow Patterns in Horizontal Pipes. SPE Production & Facilities 12, 165–172. https://doi.org/10.2118/36609-PA

Tryggvason, G., Bunner, B., Esmaeeli, A., Juric, D., Al-Rawahi, N., Tauber, W., Han, J., Nas, S., Jan, Y.-J., 2001. A Front-Tracking Method for the Computations of Multiphase Flow. Journal of Computational Physics 169, 708–759. https://doi.org/10.1006/jcph.2001.6726

Urdahl, O., Fredheim, A.O., Løken, K.-P., 1997. Viscosity measurements of water-in-crude-oil emulsions under flowing conditions: A theoretical and practical approach. Colloids and Surfaces A: Physicochemical and Engineering Aspects, Frontiers in Colloid Chemistry an International Festschrift to Professor Stig E. Friberg 123–124, 623–634. https://doi.org/10.1016/S0927-7757(96)03801-0

V. Hernandez-Perez, M. Abdulkadir, B.J Azzopardi, 2011. Grid Generation Issues in the CFD Modelling of Two-Phase Flow in a Pipe. The Journal of Computational Multiphase Flows 3, 14.

Vindenes, Eikeseth, Ramachandran, 2021. Review and computational modelling of oil-water flow in pipes (Project work No. MP-01-21). University of South-Eastern Norway.

W. Amaranath Sena Kumara, 2010. An Experimental Study of Oil-Water Flow in Pipes. University of South-Eastern Norway, Porsgrunn.

Walvekar, R.G., Choong, T.S.Y., Hussain, S.A., Khalid, M., Chuah, T.G., 2009. Numerical study of dispersed oil–water turbulent flow in horizontal tube. Journal of Petroleum Science and Engineering 65, 123–128. https://doi.org/10.1016/j.petrol.2008.12.019

What is a good Mesh?, 2014.

Xu, X.-X., 2007. Study on oil–water two-phase flow in horizontal pipelines. Journal of Petroleum Science and Engineering 59, 43–58. https://doi.org/10.1016/j.petrol.2007.03.002

Yakhot, Orszag, 1986. Renormalization Group Analysis of Turbulence. J Sci Comput 1, 3–51. https://doi.org/10.1007/BF01061452

Yang, J., Li, P., Zhang, X., Lu, X., Li, Q., Mi, L., 2021. Experimental investigation of oil–water flow in the horizontal and vertical sections of a continuous transportation pipe. Sci Rep 11, 20092. https://doi.org/10.1038/s41598-021-99660-8

Zhu, H.P., Zhou, Z.Y., Yang, R.Y., Yu, A.B., 2007. Discrete particle simulation of particulate systems: Theoretical developments. Chemical Engineering Science, Frontier of Chemical Engineering - Multi-scale Bridge between Reductionism and Holism 62, 3378–3396. https://doi.org/10.1016/j.ces.2006.12.089

Appendices

Appendix A – Project topic description

University of South-Eastern Norway

Faculty of Technology, Natural Sciences and Maritime Sciences, Campus Porsgrunn

FMH606 Master's Thesis

Title: Computational modelling of oil-water two-phase flow in pipes

USN supervisor: Amaranath S. Kumara

External partner: -

Task background:
Liquid-liquid two-phase flow in pipes are a common occurrence in petrochemical industry applications. The current situation in existing oil reservoirs is that the water cut in the production wells will increase as the reservoir matures. How the oil-water flow is influenced by an increased water cut with regards to pressure-drop, velocity and turbulence profile is of great interest to the industry. The accurate prediction of velocity profile, pressure gradient and void fraction in oil-water two-phase flow is of both scientific and technological interest.

Task description:
The main objective of this thesis is to develop a computational model for oil-water two-phase flow in a horizontal pipe using OpenFOAM. The following key activities will be done:

1. Perform a literature review of existing models for two-phase oil-water flow in horizontal pipes.
2. Implementation of computational model for oil-water flow using OpenFOAM. The model will attempt to accurately predict the velocity profile of both water and oil at different water cuts. The OpenFOAM model will be developed using existing literature and models available.
3. Validation of the CFD model will be based on the experimental work performed by Kumara et al. 2009 and 2010 (https://www-sciencedirect-com.ezproxy2.usn.no/science/article/pii/S000925091000206X https://iopscience-iop-org.ezproxy2.usn.no/article/10.1088/0957-0233/20/11/114004/pdf).

Student category: PT

Is the task suitable for online students (not present at the campus)? Yes

Practical arrangements:
OpenFOAM simulations will be done in the campus pc lab.

Supervision:
As a general rule, the student is entitled to 15-20 hours of supervision. This includes necessary time for the supervisor to prepare for supervision meetings (reading material to be discussed, etc).

Signatures:

Supervisor (date and signature):

Student (write clearly in all capitalized letters): MARTIN HAMRE

Student (date and signature): Martin Hamre

Figure A.1: Task description

Appendix B – Mesh quality

```
Mesh stats
    points:           607944
    faces:            2117126
    internal faces:   2074074
    cells:            755040
    faces per cell:   5.55096
    boundary patches: 4
    point zones:      0
    face zones:       0
    cell zones:       1
```

Figure B.1 – Mesh stats.

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

Checking patch topology for multiply connected surfaces...
    Patch               Faces    Points   Surface topology
    Wall                41600    41640    ok (non-closed singly connected)
    outlet              726      584      ok (non-closed singly connected)
    inlet_oil           363      309      ok (non-closed singly connected)
    inlet_water         363      309      ok (non-closed singly connected)

Checking geometry...
    Overall domain bounding box (0 -0.028 -0.028) (15 0.028 0.028)
    Mesh has 3 geometric (non-empty/wedge) directions (1 1 1)
    Mesh has 3 solution (non-empty) directions (1 1 1)
    Boundary openness (8.74571e-19 -1.88285e-16 -1.45525e-17) OK.
    Max cell openness = 2.72488e-16 OK.
    Max aspect ratio = 45.8969 OK.
    Minimum face area = 1.90869e-06. Maximum face area = 8.78749e-05.  Face area magnitudes OK.
    Min volume = 1.90869e-08. Max volume = 1.05433e-07.  Total volume = 0.0367934.  Cell volumes OK.
    Mesh non-orthogonality Max: 31.4928 average: 3.70971
    Non-orthogonality check OK.
    Face pyramids OK.
    Max skewness = 0.257498 OK.
    Coupled point location match (average 0) OK.
```

Figure B.2 – Geometry topology.

Appendix C: '0' folder for – $U_M = 0.50, \lambda_w = 0.25$ – SST k-ω

Alpha.orig

```
/*--------------------------------*- C++ -*----------------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*---------------------------------------------------------------------
---------*/
FoamFile
{
    format      ascii;
    class       volScalarField;
    location    "0";
    object      alpha.oil;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet_oil
    {
        type            fixedValue;
        value           uniform 0;
    }
    inlet_water
    {
        type            fixedValue;
        value           uniform 1;
    }

    Wall
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    defaultFaces
    {
        type            empty;
```

```
    }
}
```

Epsilon

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

/*--------------------------------*- C++ -*----------------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*---------------------------------------------------------------------
---------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * //

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

internalField   uniform 0.007;

boundaryField
{
    inlet_oil
    {
        type            fixedValue;
        value           uniform 0.0001;
    }

    inlet_water
    {
        type            fixedValue;
        value           uniform 0.0001;
    }

    Wall
    {
        type            epsilonWallFunction;
        value           uniform 0.000015;
    }
```

```
    outlet
    {
        type            zeroGradient;

    }

    defaultFaces
    {
        type            empty;

    }
}

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

K

```
/*--------------------------------*- C++ -*----------------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*---------------------------------------------------------------------
---------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * //

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

internalField   uniform 0.003;

boundaryField
{
    inlet_oil
    {
        type            fixedValue;
        value           uniform 0.0017;
    }

    inlet_water
    {
        type            fixedValue;
        value           uniform 0.00022;
    }
```

```
    Wall
    {
        type            kqRWallFunction;
        value           $internalField;
    }

    outlet
    {
        type            zeroGradient;
    }

    defaultFaces
    {
        type            empty;
    }
}

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

Nut

```
/*--------------------------------*- C++ -*----------------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*---------------------------------------------------------------------
---------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * //

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

internalField   uniform 0;

boundaryField
{
    Wall
    {
        type            nutkWallFunction;
        value           uniform 0;
```

```
    }

    ".*"
    {
        type            calculated;
        value           uniform 0;
    }

    defaultFaces
    {
        type            empty;
    }
}


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

Omega

```
/*--------------------------------*- C++ -*-------------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*------------------------------------------------------------------
---------*/
FoamFile
{
    format      ascii;
    class       volScalarField;
    object      omega.water;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * //

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

internalField   uniform 0.9;

boundaryField
{
    inlet_oil
    {
       type            fixedValue;
       value           uniform 1.3472;
    }

    inlet_water
    {
       type            fixedValue;
       value           uniform 0.5136;
```

```
    }
    outlet
    {
        type            zeroGradient;
    }

    wall
    {
        type            omegaWallFunction;
        value           uniform 1;
    }

    defaultFaces
    {
        type            empty;

    }
}

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

P_rgh

```
/*--------------------------------*- C++ -*----------------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*---------------------------------------------------------------------
---------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p_rgh;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * //

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

internalField   uniform 100000;

boundaryField
{
    inlet_oil
    {
        type            fixedFluxPressure;
        gradient        uniform 0;
```

```
        value           uniform 100000;

    }

    inlet_water
    {
        type            fixedFluxPressure;
        gradient        uniform 0;
        value           uniform 100000;

    }

    Wall
    {
        type            fixedFluxPressure;
        gradient        uniform 0;
        value           uniform 100000;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 100000;
    }

    defaultFaces
    {
        type            empty;
    }
}

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

U

```
/*--------------------------------*- C++ -*----------------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*---------------------------------------------------------------------
---------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
```

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

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

internalField   uniform (1 0 0);

boundaryField
{
    inlet_oil
    {
        type            fixedValue;
        value           uniform (0.75 0 0);

    }
        inlet_water
    {
        type            fixedValue;
        value           uniform (0.25 0 0);

    }

    Wall
    {
        type            noSlip;
    }

    outlet
    {
        type            zeroGradient;
    }

    defaultFaces
    {
        type            empty;
    }
}

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

Appendix D: 'Constant' folder – $U_M = 0.50, \lambda_w = 0.25$ – SST k-ω

g

```
/*--------------------------------*- C++ -*----------------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*---------------------------------------------------------------------
---------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    location    "constant";
    object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * //

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

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

momentumTransport

```
/*--------------------------------*- C++ -*----------------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*---------------------------------------------------------------------
---------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      momentumTransport;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * //

simulationType  RAS;
```

```
RAS
{
    model           kOmegaSST;

    turbulence      on;

    printCoeffs     on;
}


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

transportProperties

```
/*--------------------------------*- C++ -*----------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*---------------------------------------------------------------
---------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * //

phases (water oil);

water
{
    transportModel  Newtonian;
    nu              1e-06;
    rho             996;
}

oil
{
    transportModel  Newtonian;
    nu              1.64e-06;
    rho             790;
}

sigma           0.043;
```

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

Appendix E: 'system' folder - – $U_M = 0.50, \lambda_w = 0.25$ – SST k-ω

controlDict

```
/*--------------------------------*- C++ -*----------------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*---------------------------------------------------------------------
---------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * //

application     interFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         60;

deltaT          0.001;

writeControl    adjustableRunTime;

writeInterval   5;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;
```

```
adjustTimeStep  yes;

maxCo           0.5;
maxAlphaCo      0.5;
maxDeltaT       0.1;

functions
{
    #includeFunc fieldAverage(U, p, prime2Mean = yes)
}


//
*********************************************************************
***** //
/*--------------------------------*- C++ -*----------------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*---------------------------------------------------------------------
---------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * //

numberOfSubdomains 6;

method          simple;

simpleCoeffs
{
    n               (6 1 1);
    delta           0.001;
}

hierarchicalCoeffs
{
    n               (2 2 2);
    delta           0.001;
    order           xyz;
}

manualCoeffs
```

```
{
    dataFile        "";
}

distributed     no;

roots           ( );


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

fvSchemes

```
/*--------------------------------*- C++ -*----------------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*-------------------------------------------------------------------------
---------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default             none;
    div(rho*phi,U)  Gauss linearUpwind grad(U);
    div(phi,alpha)  Gauss PLIC interfaceCompression vanLeer 1;
//    div(phirb,alpha) Gauss interfaceCompression;
    div(phi,k)      Gauss upwind;
    div(phi,omega) Gauss upwind;
//    div(phi,R)      Gauss upwind;
//    div(R)          Gauss linear;
```

```
//    div(phi,nuTilda) Gauss upwind;
//    div((nuEff*dev(T(grad(U))))) Gauss linear;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
    div(rhoPhi,U)       Gauss linearUpwind grad(U);
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

wallDist
{
    method meshWave;
}


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

fvSolution

```
/*--------------------------------*- C++ -*----------------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*---------------------------------------------------------------------
---------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * //
```

```
solvers
{
    "alpha.water.*"
    {
        nAlphaCorr      1;
        nAlphaSubCycles 1;
      cAlpha          2;
        MULESCorr       yes;
        nLimiterIter    3;

        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-8;
        relTol          0;
    }

    "pcorr.*"
    {
        solver          PCG;
        preconditioner
        {
            preconditioner  GAMG;
            tolerance       1e-5;
            relTol          0;
            smoother        GaussSeidel;
        }
        tolerance       1e-5;
        relTol          0;
        maxIter         50;
    }

    p_rgh
    {
        solver           GAMG;
        tolerance        5e-9;
        relTol           0.01;
        smoother         GaussSeidel;
        maxIter          50;
    };

    p_rghFinal
    {
        $p_rgh;
        tolerance       5e-9;
        relTol          0;
    }

    "(U|k|omega)"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-6;
        relTol          0;
        nSweeps         1;
```

```
    }

    "(U|k|omega)Final"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-08;
        relTol          0;
    }
}


PIMPLE
{
    momentumPredictor no;
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
}

relaxationFactors
{
    equations
    {
        ".*" 1;
    }
}


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

setFieldsDict

```
/*--------------------------------*- C++ -*----------------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*------------------------------------------------------------------------
---------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * //
```

```
defaultFieldValues
(
    volScalarFieldValue alpha.water 0
);

regions
(
    boxToCell
    {
        box (-10 -20 -10) (50 20 2.2);
        fieldValues
        (
            volScalarFieldValue alpha.water 1
        );
    }
);


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

decomposeParDict

```
/*--------------------------------*- C++ -*----------------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*---------------------------------------------------------------------
---------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * //

numberOfSubdomains 6;

method          simple;

simpleCoeffs
{
    n               (6 1 1);
    delta           0.001;
}
```

```
hierarchicalCoeffs
{
    n                 (2 2 2);
    delta             0.001;
    order             xyz;
}

manualCoeffs
{
    dataFile          "";
}

distributed     no;

roots           ( );


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

Appendix F: Customized k-ε turbulence model

```
/*---------------------------------------------------------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Copyright (C) 2011-2021 OpenFOAM
Foundation
     \\/     M anipulation  |
-----------------------------------------------------------------------
-----------
License
This file is part of OpenFOAM.
OpenFOAM is free software: you can redistribute it and/or modify it
under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
at your option) any later version.

OpenFOAM is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
General Public License for more details. You should have received a
copy of the GNU General Public License along with OpenFOAM.  If not,
see <http://www.gnu.org/licenses/>.

\*---------------------------------------------------------------------
---------*/

#include "mykEpsilon.H"
#include "fvModels.H"
#include "fvConstraints.H"
#include "bound.H"
#include "fvCFD.H"
#include "twoPhaseMixture.H"

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

namespace Foam
{
namespace RASModels
{

// * * * * * * * * * * * * * * Protected Member Functions  * * * * * * *
* * * * //

template<class BasicMomentumTransportModel>
void mykEpsilon<BasicMomentumTransportModel>::correctNut()
{

    if (!this->turbulence_)
    {
        return;
```

```
    }
    // Access mesh object
    const fvMesh& mesh = this->mesh_;

    // Define threshold for interface detection (e.g., alpha = 0.5)
    const scalar interfaceThreshold = 0.5;

    // Iterate over cells
    forAll(mesh.cells(), cellI)
    {
        // Check if the volume fraction at the current cell is close
to the interface
        if (fabs(this->alpha_[cellI] - interfaceThreshold) < 0.1) //
Adjust tolerance as needed
        {
            // Set turbulent kinetic energy and dissipation rate to
0.0001
            this->k_[cellI] = 0.0001;
            this->epsilon_[cellI] = 0.0001;
        }
    }

    // Correct boundary conditions for nut outside the loop
    this->nut_ = Cmu_ * sqr(this->k_) / this->epsilon_;
    this->nut_.correctBoundaryConditions();
    fvConstraints::New(this->mesh_).constrain(this->nut_);
}


template<class BasicMomentumTransportModel>
tmp<fvScalarMatrix>
mykEpsilon<BasicMomentumTransportModel>::kSource() const
{
    return tmp<fvScalarMatrix>
    (
        new fvScalarMatrix
        (
            k_,
            dimVolume*this->rho_.dimensions()*k_.dimensions()
            /dimTime
        )
    );
}


template<class BasicMomentumTransportModel>
tmp<fvScalarMatrix>
mykEpsilon<BasicMomentumTransportModel>::epsilonSource() const
{
    return tmp<fvScalarMatrix>
    (
        new fvScalarMatrix
        (
            epsilon_,
```

```
            dimVolume*this->rho_.dimensions()*epsilon_.dimensions()
            /dimTime
        )
    );
}


// * * * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * *
* * * * //

template<class BasicMomentumTransportModel>
mykEpsilon<BasicMomentumTransportModel>::mykEpsilon
(
    const alphaField& alpha,
    const rhoField& rho,
    const volVectorField& U,
    const surfaceScalarField& alphaRhoPhi,
    const surfaceScalarField& phi,
    const transportModel& transport,
    const word& type
)
:
    eddyViscosity<RASModel<BasicMomentumTransportModel>>
    (
        type,
        alpha,
        rho,
        U,
        alphaRhoPhi,
        phi,
        transport
    ),

    Cmu_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "Cmu",
            this->coeffDict_,
            0.09
        )
    ),
    C1_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "C1",
            this->coeffDict_,
            1.44
        )
    ),
    C2_
    (
        dimensioned<scalar>::lookupOrAddToDict
```

```
        (
            "C2",
            this->coeffDict_,
            1.92
        )
    ),
    C3_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "C3",
            this->coeffDict_,
            0
        )
    ),
    sigmak_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "sigmak",
            this->coeffDict_,
            1.0
        )
    ),
    sigmaEps_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "sigmaEps",
            this->coeffDict_,
            1.3
        )
    ),

    k_
    (
        IOobject
        (
            IOobject::groupName("k", alphaRhoPhi.group()),
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        this->mesh_
    ),
    epsilon_
    (
        IOobject
        (
            IOobject::groupName("epsilon", alphaRhoPhi.group()),
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::MUST_READ,
```

```
            IOobject::AUTO_WRITE
        ),
        this->mesh_
    )
{
    bound(k_, this->kMin_);
    bound(epsilon_, this->epsilonMin_);

    if (type == typeName)
    {
        this->printCoeffs(type);
    }
}


// * * * * * * * * * * * * * * * * Member Functions  * * * * * * * * *
* * * * //

template<class BasicMomentumTransportModel>
bool mykEpsilon<BasicMomentumTransportModel>::read()
{
    if
(eddyViscosity<RASModel<BasicMomentumTransportModel>>::read())
    {
        Cmu_.readIfPresent(this->coeffDict());
        C1_.readIfPresent(this->coeffDict());
        C2_.readIfPresent(this->coeffDict());
        C3_.readIfPresent(this->coeffDict());
        sigmak_.readIfPresent(this->coeffDict());
        sigmaEps_.readIfPresent(this->coeffDict());

        return true;
    }
    else
    {
        return false;
    }
}


template<class BasicMomentumTransportModel>
void mykEpsilon<BasicMomentumTransportModel>::correct()
{
    if (!this->turbulence_)
    {
        return;
    }

    // Local references
    const alphaField& alpha = this->alpha_;
    const rhoField& rho = this->rho_;
    const surfaceScalarField& alphaRhoPhi = this->alphaRhoPhi_;
    const volVectorField& U = this->U_;
    volScalarField& nut = this->nut_;
```

```
    const Foam::fvModels& fvModels(Foam::fvModels::New(this-
>mesh_));
    const Foam::fvConstraints& fvConstraints
    (
        Foam::fvConstraints::New(this->mesh_)
    );

    eddyViscosity<RASModel<BasicMomentumTransportModel>>::correct();

    const scalar interfaceThreshold = 0.5; // Interface detection
threshold

    forAll(this->mesh_.cells(), cellI)
    {
        if (fabs(this->alpha_[cellI] - interfaceThreshold) < 0.1)
        {
            this->nut_[cellI] = 0.0001; // Set eddy viscosity near
interfaces
        }
    }

    this->nut_.correctBoundaryConditions();

    volScalarField::Internal divU
    (
        fvc::div(fvc::absolute(this->phi(), U))()
    );

    tmp<volTensorField> tgradU = fvc::grad(U);
    volScalarField::Internal G
    (
        this->GName(),
        nut()*(dev(twoSymm(tgradU().v())) && tgradU().v())
    );
    tgradU.clear();

    // Update epsilon and G at the wall
    epsilon_.boundaryFieldRef().updateCoeffs();

    // Dissipation equation
    tmp<fvScalarMatrix> epsEqn
    (
        fvm::ddt(alpha, rho, epsilon_)
      + fvm::div(alphaRhoPhi, epsilon_)
      - fvm::laplacian(alpha*rho*DepsilonEff(), epsilon_)
     ==
        C1_*alpha()*rho()*G*epsilon_()/k_()
      - fvm::SuSp(((2.0/3.0)*C1_ - C3_)*alpha()*rho()*divU,
epsilon_)
      - fvm::Sp(C2_*alpha()*rho()*epsilon_()/k_(), epsilon_)
      + epsilonSource()
      + fvModels.source(alpha, rho, epsilon_)
    );
```

```
    epsEqn.ref().relax();
    fvConstraints.constrain(epsEqn.ref());
    epsEqn.ref().boundaryManipulate(epsilon_.boundaryFieldRef());
    solve(epsEqn);
    fvConstraints.constrain(epsilon_);
    bound(epsilon_, this->epsilonMin_);

    // Turbulent kinetic energy equation
    tmp<fvScalarMatrix> kEqn
    (
        fvm::ddt(alpha, rho, k_)
      + fvm::div(alphaRhoPhi, k_)
      - fvm::laplacian(alpha*rho*DkEff(), k_)
     ==
        alpha()*rho()*G
      - fvm::SuSp((2.0/3.0)*alpha()*rho()*divU, k_)
      - fvm::Sp(alpha()*rho()*epsilon_()/k_(), k_)
      + kSource()
      + fvModels.source(alpha, rho, k_)
    );

    kEqn.ref().relax();
    fvConstraints.constrain(kEqn.ref());
    solve(kEqn);
    fvConstraints.constrain(k_);
    bound(k_, this->kMin_);

    correctNut();
}


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

} // End namespace RASModels
} // End namespace Foam

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

Appendix G: Turbulence damping scheme.

C file

```
/*---------------------------------------------------------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Copyright (C) 2015-2018 OpenFOAM
Foundation
     \\/     M anipulation  |
-----------------------------------------------------------------------
-----------
License
This file is part of OpenFOAM. OpenFOAM is free software: you can
redistribute it and/or modify it under the terms of the GNU General
Public License as published by the Free Software Foundation, either
version 3 of the License, or (at your option) any later version.
OpenFOAM is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY, without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
General Public License for more details. You should have received a
copy of the GNU General Public License along with OpenFOAM.  If not,
see <http://www.gnu.org/licenses/>.

\*---------------------------------------------------------------------
---------*/

#include "turbulenceDamping.H"
#include "fvMatrices.H"
#include "addToRunTimeSelectionTable.H"

// * * * * * * * * * * * * * * Static Member Functions * * * * * * * *
* * * * //

namespace Foam
{
namespace fv
{
    defineTypeNameAndDebug(turbulenceDamping, 0);

    addToRunTimeSelectionTable
    (
        option,
        turbulenceDamping,
        dictionary
    );
}
}

// * * * * * * * * * * * * * * Private Member Functions  * * * * * * *
* * * * //
```

```
volScalarField::Internal
Foam::fv::turbulenceDamping::calculateSource
(
    fvMatrix<scalar>& eqn,
    const label fieldi
)
{
    const volScalarField& Alpha =
        mesh().lookupObject<volScalarField>(primaryPhaseName_);

    const volVectorField grad_Alpha = fvc::grad(Alpha);
    const volScalarField grad_Alpha_mag = mag(grad_Alpha);

    // calculate interfacial area density
    volScalarField::Internal A1 = 2.0*Alpha*grad_Alpha_mag;
    volScalarField::Internal A2 = 2.0*(1.0-Alpha)*grad_Alpha_mag;


    // calculate the inverse of the length scale
    const volScalarField::Internal& V = mesh_.V();

    volScalarField oneByDn
    (
        IOobject
        (
            "oneByDn",
            mesh_.time().timeName(),
            mesh_,
            IOobject::NO_READ,
            IOobject::NO_WRITE
        ),
        mesh_,
        dimensionedScalar("oneByDn", dimless/dimLength, 0.0)
    );

    if (lengthScale_ == "FA")
    {
        const labelUList& owner = mesh_.owner();
        const labelUList& neighbour = mesh_.neighbour();

        const surfaceVectorField& sf = mesh_.Sf();

        forAll(owner, facei)
        {
            oneByDn[owner[facei]] +=
                mag(sf[facei] & grad_Alpha[owner[facei]]);

            oneByDn[neighbour[facei]] +=
                mag(sf[facei] & grad_Alpha[neighbour[facei]]);
        }

        forAll(mesh_.boundary(), patchi)
        {
            const labelUList& pFaceCells =
```

```
                mesh_.boundary()[patchi].faceCells();

            const fvsPatchField<vector>& psf =
sf.boundaryField()[patchi];

            forAll(mesh_.boundary()[patchi], facei)
            {
                oneByDn[pFaceCells[facei]] +=
                    mag(psf[facei] & grad_Alpha[pFaceCells[facei]]);
            }
        }

       forAll(oneByDn, celli)
       {
           if (grad_Alpha_mag[celli] > SMALL)
           {
               oneByDn[celli] *= 0.5/V[celli]/grad_Alpha_mag[celli];
           }
           else
           {
               oneByDn[celli] = 0;
           }
       }

    }

    else if (lengthScale_ == "cubeRoot")
    {
        oneByDn.ref() = pow(V,-1.0/3.0);
    }


    // calculate separate damping terms
    volScalarField::Internal coeffs =
36.0*sqr(B_)/beta_*pow(oneByDn, 3.0);
    volScalarField::Internal source1 = coeffs*A1*rho1_*sqr(nu1_);
    volScalarField::Internal source2 = coeffs*A2*rho2_*sqr(nu2_);

    // calculate the total damping term
    dimensionedScalar heavy("heavy", dimless, 0.0);

    volScalarField::Internal source = 0.0 * source1;

    if (dampingTreatment_ == "heavyNegative")
    {
        if (rho1_ > rho2_)
        {
            heavy = - rho2_/rho1_*sqr(nu2_)/sqr(nu1_);

            source = source1*heavy + source2;
        }

        else
        {
```

```
            heavy = - rho1_/rho2_*sqr(nu1_)/sqr(nu2_);

            source = source1 + source2*heavy;
        }
    }

    else if (dampingTreatment_ == "heavyZero")
    {
        if (rho1_ > rho2_)
        {
            source = source2;
        }

        else
        {
            source = source1;
        }
    }

    else if (dampingTreatment_ == "symmetric")
    {
        source = sign(B_)*(source1 + source2);
    }

    // return source term for omega equation
    if (fieldNames_[0] == "omega")
    {
        return source;
    }
    // return source term for epsilon equation
    else
    {
        const volScalarField& k =
mesh().lookupObject<volScalarField>("k");

        return C2_*sqr(Cmu_)/beta_*k.internalField()*source;
    }
}



// * * * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * *
* * * * //

Foam::fv::turbulenceDamping::turbulenceDamping
(
    const word& sourceName,
    const word& modelType,
    const dictionary& dict,
    const fvMesh& mesh
)
:
    option(sourceName, modelType, dict, mesh),
    B_(readScalar(coeffs_.lookup("B"))),
```

```
    C2_(coeffs_.lookupOrDefault<scalar>("C2", 1.92)),
    beta_(coeffs_.lookupOrDefault<scalar>("beta", 0.075)),
    Cmu_(coeffs_.lookupOrDefault<scalar>("Cmu", 0.09)),
    lengthScale_(coeffs_.lookupOrDefault<word>("lengthScale",
"FA")),
    dampingTreatment_
    (
        coeffs_.lookupOrDefault<word>("dampingTreatment",
"heavyNegative")
    ),
    explicitSourceTreatment_
    (
        coeffs_.lookupOrDefault<Switch>("explicitSourceTreatment",
true)
    ),
    transportProperties
    (
         mesh_.lookupObject<IOdictionary>
        (
            "transportProperties"
        )
    ),
    phase1Name_(wordList(transportProperties.lookup("phases"))[0]),
    phase2Name_(wordList(transportProperties.lookup("phases"))[1]),
    primaryPhaseName_("alpha." + phase1Name_),
    rho1_("rho", dimDensity,
transportProperties.subDict(phase1Name_)),
    nu1_("nu", dimViscosity,
transportProperties.subDict(phase1Name_)),
    rho2_("rho", dimDensity,
transportProperties.subDict(phase2Name_)),
    nu2_("nu", dimViscosity,
transportProperties.subDict(phase2Name_))
{
    coeffs_.lookup("fields") >> fieldNames_;

    if (fieldNames_.size() != 1)
    {
        FatalErrorInFunction
            << "settings are:" << fieldNames_ << exit(FatalError);
    }

    // only omega or epsilon is allowed
    if (fieldNames_[0] != "omega" && fieldNames_[0] != "epsilon")
    {
        FatalErrorInFunction
            << "The field is set to" << fieldNames_
            << ", it should be epsilon or omega!" <<
exit(FatalError);
    }

    // make sure the field name is consistent with the turbulence
model
    // the following line should fail if inconsistent
```

```
    const volScalarField& epsilonOrOmega =
        mesh_.lookupObject<volScalarField>(fieldNames_[0]);

    Info << "Turbulence damping works in " << epsilonOrOmega.name()
         << " mode"<< endl;

    applied_.setSize(fieldNames_.size(), false);

    Info << "B is set to " << B_.value() << endl;

    Info << "C2 is set to " << C2_.value() << endl;

    Info << "beta is set to " << beta_.value() << endl;

    Info << "Cmu is set to " << Cmu_.value() << endl;

    Info << "lengthScale is set to " << lengthScale_ << endl;

    Info << "dampingTreatment is set to " << dampingTreatment_ <<
endl;
}


// * * * * * * * * * * * * * * * * Member Functions  * * * * * * * * *
* * * * //

void Foam::fv::turbulenceDamping::addSup
(
    fvMatrix<scalar>& eqn,
    const label fieldi
)
{
    const volScalarField& Rho =
mesh().lookupObject<volScalarField>("rho");

    const volScalarField& epsilonOrOmega =
        mesh().lookupObject<volScalarField>(fieldNames_[fieldi]);

    volScalarField::Internal source = calculateSource(eqn,
fieldi)/epsilonOrOmega/Rho;

    eqn += fvm::Sp(source, epsilonOrOmega);
}


void Foam::fv::turbulenceDamping::addSup
(
    const volScalarField& rho,
    fvMatrix<scalar>& eqn,
    const label fieldi
)
{
    const dimensionSet& dimEqn = eqn.dimensions();
```

```
    const volScalarField& epsilonOrOmega =
        mesh().lookupObject<volScalarField>(fieldNames_[fieldi]);

    const dimensionSet& dimEorW = epsilonOrOmega.dimensions();

    volScalarField::Internal source = calculateSource(eqn, fieldi);

    // divide density for strict incompressible turbulence models
    if (dimEqn == dimEorW/dimTime*dimVolume)
    {
        const volScalarField& Rho =
mesh().lookupObject<volScalarField>("rho");

        source /= Rho;
    }

    if (explicitSourceTreatment_)
    {
        eqn += source;
    }
    else
    {
        eqn += fvm::Sp(source/epsilonOrOmega, epsilonOrOmega);
    }
}

bool Foam::fv::turbulenceDamping::read(const dictionary& dict)
{
    NotImplemented;

    return false;
}

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

H file

```
/*---------------------------------------------------------------------
---------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Copyright (C) 2015-2019 OpenFOAM
Foundation
     \\/     M anipulation  |
---------------------------------------------------------------------
-----------
License
This file is part of OpenFOAM. OpenFOAM is free software: you can
redistribute it and/or modify it under the terms of the GNU General
Public License as published by the Free Software Foundation, either
version 3 of the License, or (at your option) any later version.
```

```
OpenFOAM is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY, without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
General Public License for more details. You should have received a
copy of the GNU General Public License along with OpenFOAM.  If not,
see <http://www.gnu.org/licenses/>.

Class
    Foam::fv::turbulenceDamping

Description
Calculates and applies the interficial turbulence damping term to
the omega or epsilon equation. Therefore, this fvOptions can only
handle omega- or epsilon-based RANS models. In addition, this
implementation only applies to multiphase flows where the flow is
treated as a (VOF mixture in turbulence modelling. Also, each phase
should have constant properties. For backward compatibility, the
current implementation supports turbulence models with and without
the variable-density effect being considered.

Reference:
\verbatim
        Original model for omega-based equations:
            Egorov, Y. (2004).
            Validation of CFD codes with PTS-relevant test cases.
            5th Euratom Framework Programme ECORA project, pp.
91“116.

        Extension to epsilon-based equations:
            Frederix, E.M.A., Mathur, A., Dovizio, D., Geurts, B.J.,
            Komen, E.M.J. (2018).
            Reynolds-averaged modeling of turbulence damping near a
            large-scale interface in two-phase flow.
            Nuclear Engineering and Design, 333, pp. 122â€“130.

        The current implementation is based on:
            Fan, W. & Anglart, H. (2019).
            Progress in Phenomenological Modeling of Turbulence
Damping
            around a Two-Phase Interface.
            Fluids, 4(3), 136.
\endverbatim


Usage
Example usage:
\verbatim
fields             (omega);   // Name of the field: omega or epsilon

B                  0;         // Damping coefficient, 0 means no
damping

lengthScale        FA;        // Optional parameter to specify the
method
```

```
                                    // to calculate the length scale
for damping.
                                    // The alternative is "cubeRoot".

dampingTreatment   heavyNegative; // Optional parameter to specify
the
                                        // treatment for the heavier
phase.
                                        // Alternatives are
"heavyZero" and
                                        // "symmetric".

explicitSourceTreatment true; // Optional parameter to specify
whether the
                                    // source term should be treated
explicitly.
\endverbatim

Author
    Wenyuan Fan

SourceFiles
    turbulenceDamping.C

\*---------------------------------------------------------------------
---------*/

#ifndef turbulenceDamping_H
#define turbulenceDamping_H
#include "fvCFD.H"
#include "fvOption.H"
#include "uniformDimensionedFields.H"

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

namespace Foam
{
namespace fv
{

/*---------------------------------------------------------------------
---------*\
               Class turbulenceDamping Declaration
\*---------------------------------------------------------------------
---------*/

class turbulenceDamping
:
    public option
{
    // Private Data

        dimensionedScalar B_;
```

```
        dimensionedScalar C2_;

        dimensionedScalar beta_;

        dimensionedScalar Cmu_;

        word lengthScale_;

        word dampingTreatment_;

        Switch explicitSourceTreatment_;

        const dictionary& transportProperties;

        word phase1Name_;

        word phase2Name_;

        const word primaryPhaseName_;

        dimensionedScalar rho1_;

        dimensionedScalar nu1_;

        dimensionedScalar rho2_;

        dimensionedScalar nu2_;

        //- Source term to omega or epsilon equation
        volScalarField::Internal calculateSource
        (
            fvMatrix<scalar>& eqn,
            const label fieldi
        );

public:

    //- Runtime type information
    TypeName("turbulenceDamping");


    // Constructors

        //- Construct from explicit source name and mesh
        turbulenceDamping
        (
            const word& sourceName,
            const word& modelType,
            const dictionary& dict,
            const fvMesh& mesh
        );

        //- Disallow default bitwise copy construction
```

```
        turbulenceDamping(const turbulenceDamping&) = delete;


    // Member Functions

        // Evaluate

            //- Add turbulence damping to strict incompressible
equation
            virtual void addSup
            (
                fvMatrix<scalar>& eqn,
                const label fieldi
            );

            //- Add turbulence damping to full-form equation
            virtual void addSup
            (
                const volScalarField& rho,
                fvMatrix<scalar>& eqn,
                const label fieldi
            );


        // IO

            //- Read source dictionary
            virtual bool read(const dictionary& dict);


    // Member Operators

        //- Disallow default bitwise assignment
        void operator=(const turbulenceDamping&) = delete;
};


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

} // End namespace fv
} // End namespace Foam

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

#endif

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

Appendix H – Experimental data

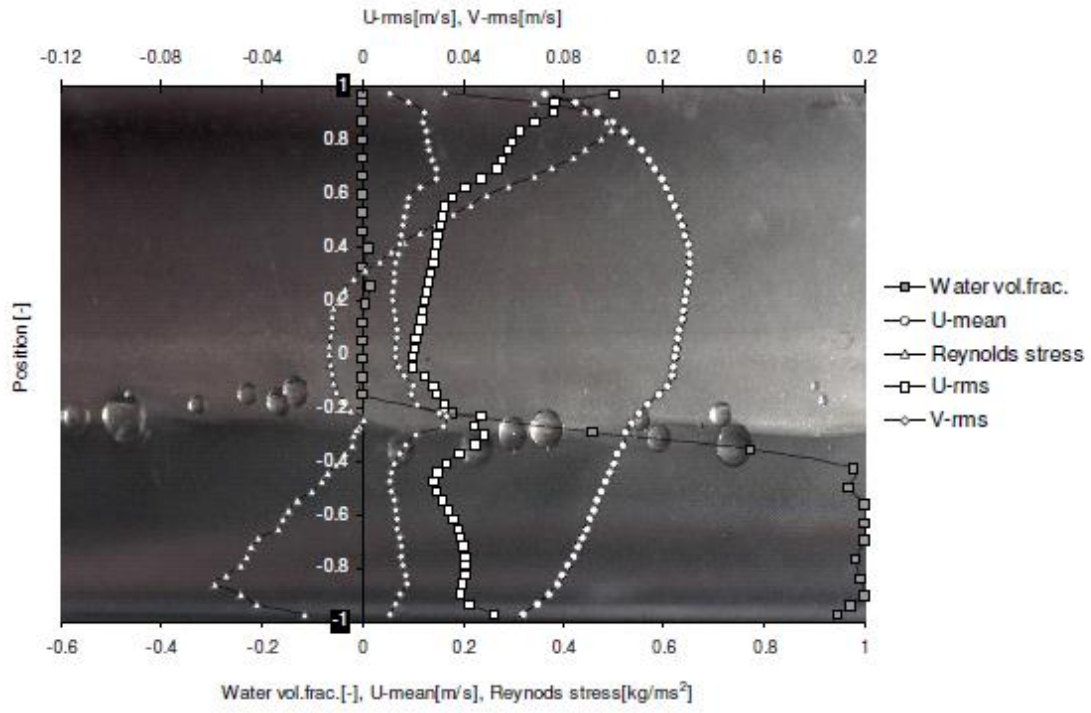

Figure H.1 – Experimental data for U_m=0.50 m/s and λ_w=0.25.

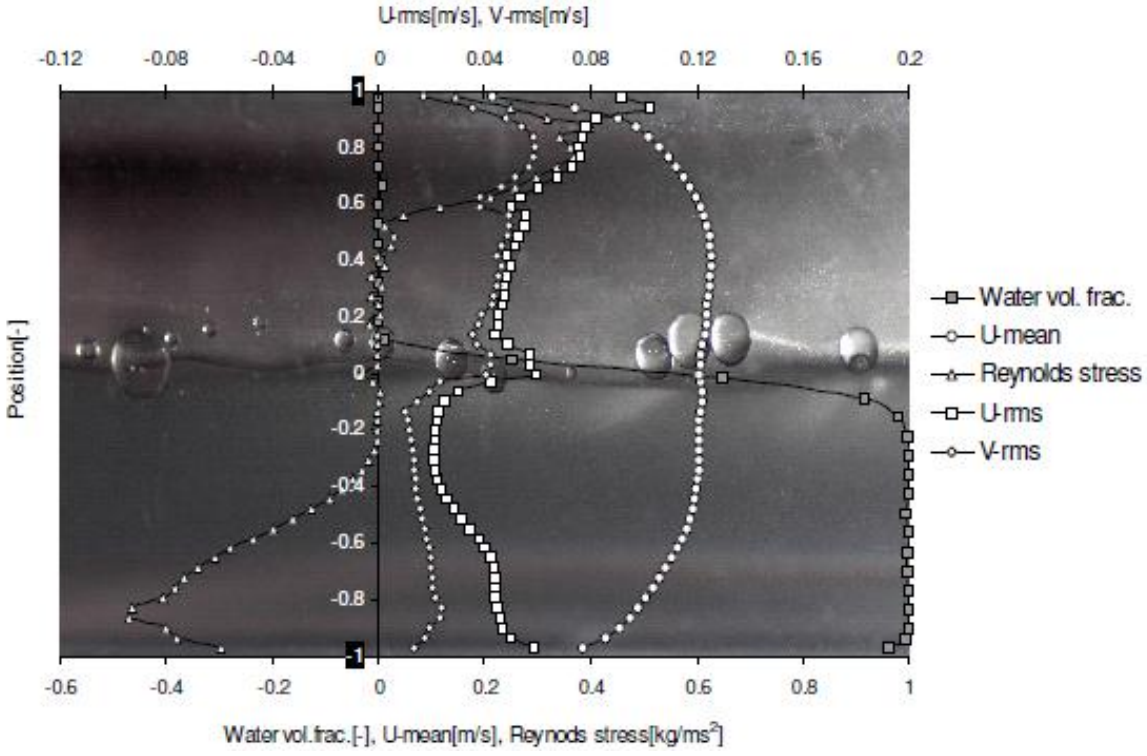

Figure H.2 - Experimental data for U_m=0.50 m/s and λ_w=0.50.

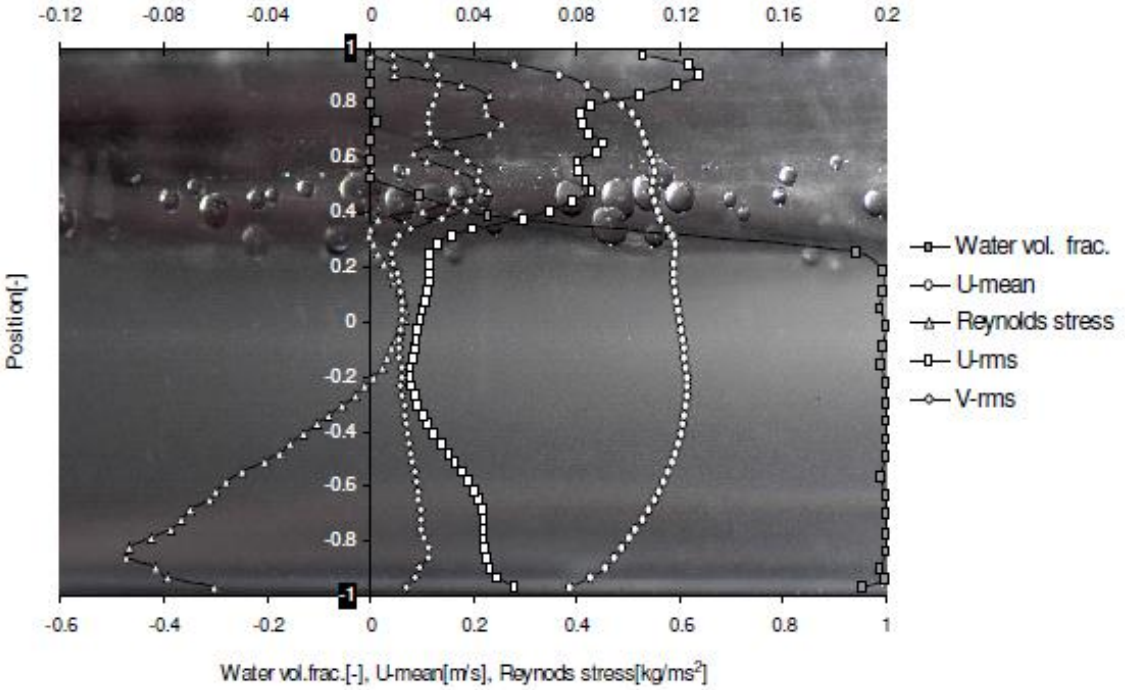

Figure H.3 - Experimental data for U_m=0.50 m/s and λ_w=0.75.

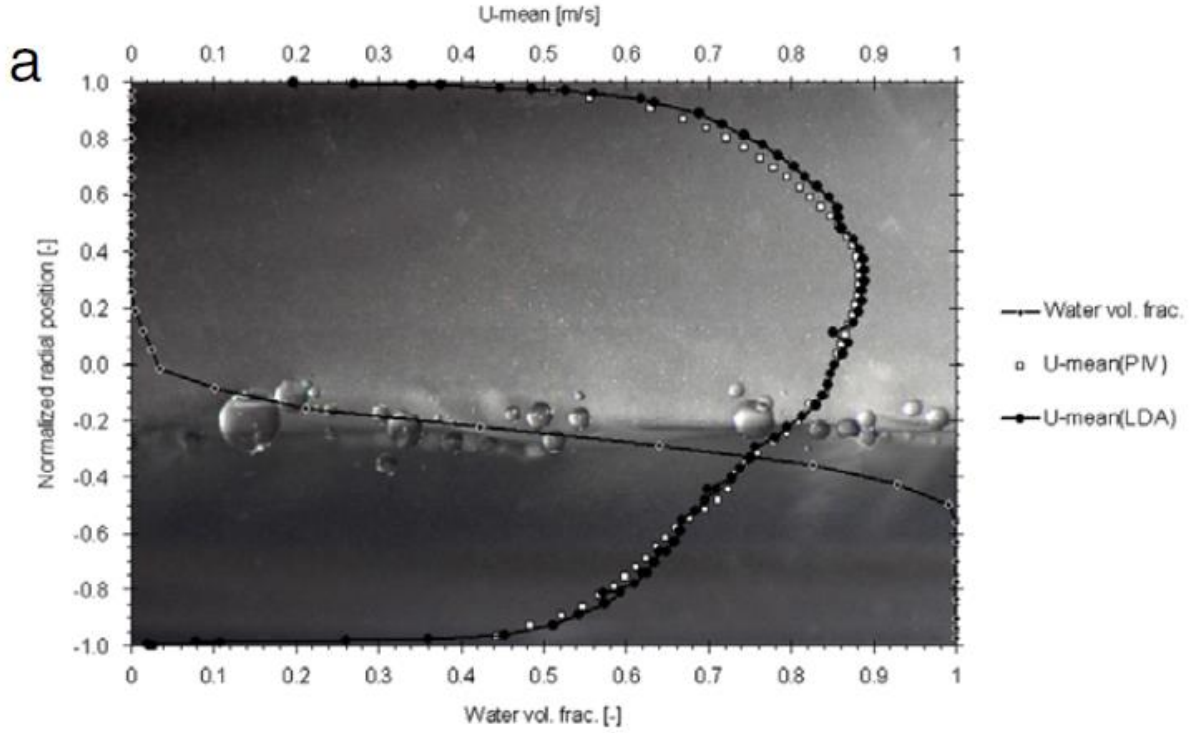

Figure H.4 - Experimental data for U_m=0.68 m/s and λ_w=0.25.

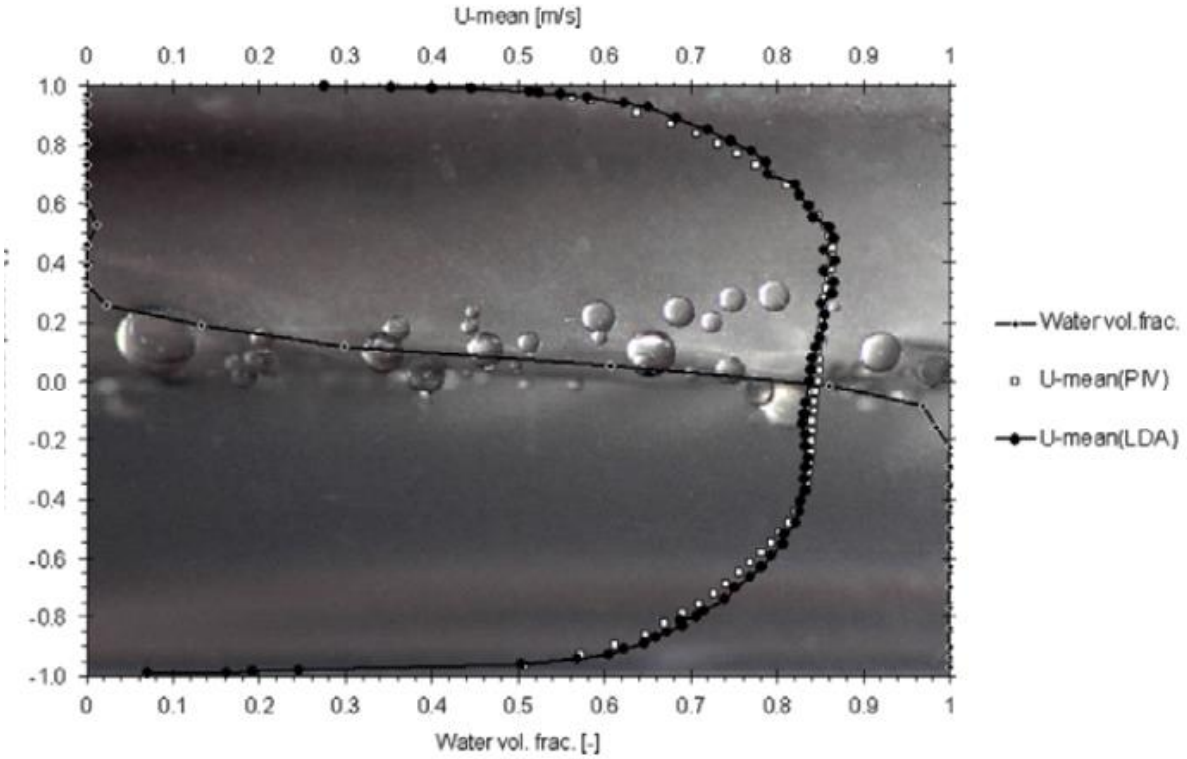

Figure H.5 - Experimental data for U_m=0.68 m/s and λ_w=0.50.