CFD SIMULATION OF SIEVE TRAY HYDRODYNAMICS USING OPENFOAM

by

Ke Tan

Submitted in partial fulfillment of the requirements
for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
September 2022

I would like to dedicate this thesis to my father. I couldn't have done this without his support along the way.

Table of Contents

List of Tables

List of Figures

Abstract

A computational fluid dynamics (CFD) model was used to predict the hydrodynamics of transient 3-dimensional flow on sieve trays. The model is constructed in the Eulerian-Eulerian framework for two interpenetrating phases. The interphase momentum exchange term is modelled as a drag force term and evaluated using a suitable literature correlation. The turbulence properties are evaluated using a turbulence model based on the $k - \epsilon$ equations for the mixture of two phases. The open-source CFD software package OpenFOAM® version 6 was employed to build and solve the model. A set of test cases was conducted to determine the optimized solver settings. A series of simulations for sieve trays of different sizes and under different operating conditions were conducted and validated using previous CFD and experimental studies. The results of the sieve tray simulations are in good agreement with the validation data.

List of Abbreviations and Symbols Used

$\mathbf{d}_f$	Distance between the centers of two neighbouring control volumes that share the common face f, m
$\mathbf{g}$	Gravitational acceleration, $\mathrm{m\,s^{-2}}$
$\mathbf{I}$	Identity matrix, –
$\mathbf{M}$	Interphase momentum transfer rate, $\mathrm{kg\,m^{-2}\,s^{-2}}$
$\mathbf{M}_D$	Momentum transfer due to drag force, $\mathrm{kg\,m^{-2}\,s^{-2}}$
$\mathbf{M}_L$	Momentum transfer due to lift force, $\mathrm{kg\,m^{-2}\,s^{-2}}$
$\mathbf{M}_{TD}$	Momentum transfer due to turbulent dispersion force, $\mathrm{kg\,m^{-2}\,s^{-2}}$
$\mathbf{M}_{VM}$	Momentum transfer due to virtual mass force, $\mathrm{kg\,m^{-2}\,s^{-2}}$
$\mathbf{M}_{WL}$	Momentum transfer due to wall lubrication force, $\mathrm{kg\,m^{-2}\,s^{-2}}$
$\mathbf{n}$	Unit vector normal to the patch face, –
$\mathbf{Pr}_t$	Turbulence Prandtl Number , –
$\mathbf{Re}$	Reynolds number, –
$\mathbf{Re}_t$	Turbulent Reynolds number, –
$\mathbf{R}$	Reynolds stress tensor, $\mathrm{kg\,m^{-1}\,s^{-2}}$
$\mathbf{r}_Q$	Distance form origin to the center of control volume V_Q, m
$\mathbf{R}_{eff}$	Effective stress, $\mathrm{m^2\,s^{-2}}$
$\mathbf{Sc}_k$	Turbulent Schmidt number for the turbulent kinetic energy, –
$\mathbf{Sc}_\epsilon$	Turbulent Schmidt number for the turbulent kinetic energy dissipation rate, –

$\mathbf{s}_f$	The normal vector of patch f, –
$\mathbf{U}$	Mean velocity, $\mathrm{m\,s^{-1}}$
$\mathbf{U}_r$	Relative velocity, $\mathrm{m\,s^{-1}}$
$\mathbf{U}_s$	Superficial velocity, $\mathrm{m\,s^{-1}}$
$\mathbf{U}_{G,in}$	Inlet velocity of gas through holes on a sieve tray, $\mathrm{m\,s^{-1}}$
$\mathbf{U}_{slip}$	Slip velocity, $\mathrm{m\,s^{-1}}$
$\mathbf{v}$	Local velocity, $\mathrm{m\,s^{-1}}$
$\mathbf{x}$	Spatial location, m
A_B	Bubbling area of a sieve tray, $\mathrm{m^2}$
A_d	Drag related coefficient, $\mathrm{kg\,m^{-3}\,s^{-1}}$
A_H	Total area for holes of a sieve tray, $\mathrm{m^2}$
C_d	Drag coefficient, –
C_o	Courant number, –
c_p	Specific heat capacity, $\mathrm{J\,kg^{-1}\,K^{-1}}$
C_s	Swarm correction coefficient, –
C_t	Turbulence response coefficient, –
C_1	Dimensionless constant in $k-\epsilon$ turbulence model, –
C_2	Dimensionless constant in $k-\epsilon$ turbulence model, –
C_μ	Dimensionless constant in $k-\epsilon$ turbulence model, –
$C_{t,0}$	Value of C_t at $\alpha_d \approx 0$, –
d	Bubble diameter, m

e	Internal energy, J
f	face f of a control volume, –
f_1	Blending factor, –
f_2	Blending factor, –
F_{lv}	Flow parameter, –
G_k	Production of turbulent kinetic energy due to the non-isotropic part of the Reynolds stress tensor, $\mathrm{kg\,m^{-1}\,s^{-3}}$
h	Height in the opposite direction to gravity, m
H_f	Heat of fusion, $\mathrm{J\,kg^{-1}}$
h_{ap}	Downcomer clearance, m
k	Turbulent kinetic energy, $\mathrm{m^2\,s^{-2}}$
K_1	Drag coefficients evaluated on the basis of phase 1 as the dispersed phase, –
K_2	Drag coefficients evaluated on the basis of phase 2 as the dispersed phase, –
K_d	Drag related coefficient, $\mathrm{kg\,m^{-4}}$
$K_{1\&2}$	Drag coefficient evaluated for situation in which both phases are considered as partly continuous or partly dispersed, –
L	Characteristic length of the turbulence, m
L_e	Eddy length scale, m
L_w	Weir length, m
N	Center of control volume V_N, –
P	Mean pressure, $\mathrm{kg\,m^{-1}\,s^{-2}}$
p	Local pressure, $\mathrm{kg\,m^{-1}\,s^{-2}}$

p_{rgh}	Pseudo hydrostatic pressure, $\mathrm{kg\,m^{-1}\,s^{-2}}$
Q	Center of control volume V_Q, –
Q_G	Gas load, $\mathrm{m^3\,s^{-1}}$
Q_L	Liquid load, $\mathrm{m^3\,s^{-1}}$
S_ϕ	Source/sink term, $[\phi]\,\mathrm{kg\,m^{-3}\,s^{-1}}$
S_k	Sources of turbulent kinetic energy, $\mathrm{kg\,m^{-1}\,s^{-3}}$
S_ϵ	Sources for the rate of turbulent kinetic energy dissipation, $\mathrm{kg\,m^{-1}\,s^{-4}}$
t	Time, s
V	Volume, $\mathrm{m^3}$
V_Q	Control volume with center Q, –
V_s	Superficial gas velocity based on active area, $\mathrm{m\,s^{-1}}$
y	The coordinate position in the transverse direction to liquid flow, m

Superscript

$'$	Turbulent fluctuation value
^{B}	Averaged value at quasi-steady-state
^{n}	Flow value at new time step
^{o}	Flow value at old time step

Subscript

$_c$	Properties of continuous phase
$_d$	Properties of dispersed phase
$_i$	Properties of phase i

$_m$	Properties of mixture
$_Q$	Properties stored at the center of control volume C_Q
$_{ref}$	Explicitly specified reference value
$_{residual}$	Residual value

Greek letters

α	Volumetric phase fraction, –
α_1	Volumetric phase fraction for phase 1, –
α_2	Volumetric phase fraction for phase 2, –
α_t	Turbulent thermal diffusivity, $\mathrm{m^2\,s^{-1}}$
$\alpha_{1,F}$	The minimum phase fraction value for phase 1 to be considered as a fully continuous phase, –
$\alpha_{1,P}$	The minimum phase fraction value for phase 1 to be considered as a partly continuous phase, –
$\alpha_{2,F}$	The minimum phase fraction value for phase 2 to be considered as a fully continuous phase, –
$\alpha_{2,P}$	The minimum phase fraction value for phase 2 to be considered as a partly continuous phase, –
β	Coefficient in C_t model, –
$\boldsymbol{\tau}$	Viscous stress tensor, $\mathrm{kg\,m^{-1}\,s^{-2}}$
χ	Phase indicator function, –
Δh_f	Heat of formation, $\mathrm{J\,mol^{-1}}$
ϵ	Dissipation rate of turbulent kinetic energy, $\mathrm{m^2\,s^{-3}}$

Γ	Diffusion coefficient, $\mathrm{kg\,m^{-1}\,s^{-1}}$
κ	Thermal conductivity, $\mathrm{W\,m^{-1}\,K^{-1}}$
μ	Dynamic viscosity, $\mathrm{kg\,m^{-1}\,s^{-1}}$
μ_t	Dynamic eddy viscosity, $\mathrm{kg\,m^{-1}\,s^{-1}}$
μ_{eff}	Effective viscosity, $\mathrm{kg\,m^{-1}\,s^{-1}}$
ν_t	Kinematic eddy viscosity, $\mathrm{m^2\,s^{-1}}$
Φ	Tray diameter, m
ϕ	Arbitrary quantity, $[\phi]$
ρ	Density, $\mathrm{kg\,m^{-3}}$

Abbreviations

CFD	Computational fluid dynamics
FVM	Finite volume method
NS	Navier–Stokes
PDE	Partial differential equation
PIMPLE	Combination of PISO and SIMPLE
PISO	Pressure-implicit with splitting of operators
RANS	Reynolds-averaged Navier–Stokes
SIMPLE	Semi-implicit method for pressure-linked equations
SST	Shear stress transport
TVD	Total variation diminishing

Chapter 1

Introduction

1.1 Background

In the petroleum, chemical, and related process industries, distillation is the most widely used separation unit to carryout mass transfer between liquid and gas phases, which consumes the biggest share of capital and operating costs [11]. The mass transfer primarily occurs at the internal structures—e.g., trays—inside the column. The sieve tray is one of the dominant internals for distillation columns owing to various attractive features, including simple structure, low maintenance, high cost-effectiveness, and wide operating range. The separation efficiency and overall performance of distillation columns is strongly influenced by the flow of fluid phases over the trays. These factors have motivated the detailed study of distillation tray hydraulic characteristics.

Early related studies [3, 4, 44–46] were limited to experimental work due to the difficulties associated with modelling the complex behaviour of the multiphase flow on the tray. In recent years, with the software and hardware development of computers, there has been an emerging trend of using computational fluid dynamics (CFD) to model the complex flows on the distillation tray [10, 22, 24, 33] and to optimize tray geometry [21, 29, 47, 51].

Most of the previous CFD studies for distillation tray hydrodynamics were conducted using commercial software packages—e.g., ANSYS CFX—which have been proved to be able to provide reliable results when combined with appropriate interphase momentum closure and turbulence models. In this work, CFD studies were conducted using the OpenFOAM® package, which is an open-source, customizable software toolbox for CFD that was originally created by Henry Weller [13] in 1989.

1.2 Literature review

1.2.1 Experimental studies

Stichlmair and Ulbrich [46] conducted experimental studies using a bubble cap and a sieve tray. The multiphase system was composed of hot water and air. Thermocouples were used to measure local liquid temperatures at multiple locations. Isotherms within the liquid phase were calculated using local temperatures and interpreted as contours of residence time.

Bell [3, 4] conducted experimental studies on commercial-scale distillation trays using the fluorescent tracer and fiber optic techniques to determine liquid-phase residence time distributions on the tray.

Solari and Bell [45] utilized the same experimental techniques and obtained fluid flow patterns and velocity profiles on commercial-scale sieve trays. These experimental results were commonly used as validating criteria in later CFD studies [10, 33].

Schubert et al. [44] used a wire-mesh sensor technique that tracks conductivity tracer pulses during their passage across the tray. The residence time distribution and velocity profile on a tray of 800 mm diameter were obtained with high spatial and temporal resolution.

1.2.2 CFD studies

Liu et al. [30] proposed a simple two-dimensional model for two-phase flow on a sieve tray in an Eulerian-Eulerian framework. The flow variations in the froth regime along the vertical direction were not considered. The interphase momentum exchange was modelled as a resisting force on the liquid phase, evaluated based on the assumption that the gas phase obtains the same velocity as the surrounding liquid phase in the horizontal directions in the froth regime. They modeled the turbulence properties in the liquid phase using a $k - \epsilon$ model. No turbulence model was employed for the gas phase. The simulation results were compared with experimental measurements by the authors. They concluded that the proposed model is suitable for predicting two-dimensional liquid flow patterns on a sieve tray.

Mehta et al. [33] simulated the three-dimensional flows on the sieve tray using a single-phase model in an Eulerian framework for the liquid phase only. The momentum transfer between liquid and gas phases was modelled as a force term acting on the liquid phase. The vertical component of this force terms was calculated from a mean momentum balance in the froth regime, and the horizontal components were evaluated with the assumption that the rising bubbles of the gas phase accelerate to the surrounding liquid velocity in the horizontal directions in the froth regime. The turbulence was modelled using an empirical correlation proposed by Zuiderweg [52]. Simulations of a commercial-scale sieve tray were conducted and the results were evaluated against experimental measurements by Solari and Bell [45]. They found that the simulation results of the proposed model were within 33% of the measured values, indicating the model should be further improved to provide more accurate predictions.

Krishna et al. [22] simulated the three-dimensional flows of gas and liquid phases on a sieve tray using a two-phase model in an Eulerian-Eulerian framework. The momentum transfer term between the two phases was estimated based on the correlation of Bennett et al. [5], which assumes that the drag force dominates the momentum exchange through bubble-liquid interactions. They modelled the turbulence properties of the liquid phase using the standard $k-\epsilon$ model [14]. No turbulence model was employed for the gas phase, thereby assuming that the turbulence of the two-phase system is dominated by the liquid velocity field. They simulated the three-dimensional flows on a small rectangular experimental-scale tray using the package of models previously described. The results were validated against the experimental measurements conducted in the same study. They concluded that the proposed model made reasonable predictions of flow patterns but overpredicted the liquid hold-up in the froth regime.

Krishna and Van Baten [24] extended the same simulation approach proposed by Krishna et al. [22] to simulate the hydrodynamics of a larger experimental-scale sieve tray and containing catalyst filled containers. The results showed good quantitative agreement with the experimental measurements of Van Baten et al. [49]. They concluded that the proposed CFD approach encapsulated the geometry and scale effects properly.

Gesit et al. [10] employed the same models proposed by Krishna et al. [22]

and conducted simulations on a commercial-scale sieve tray configuration. Their simulations were validated using the experimental data from the work of Solari and Bell [45].

Zarei et al. [51] simulated the three-dimensional flows using the same models proposed by Krishna et al. [22] for a sieve tray based on the work of Solari and Bell [45] and a geometrically similar Mini V-Grid valve tray. The concluded that the Mini V-Grid valve tray has a higher capacity compared to the sieve tray. The simulation results of sieve tray are in agreement with the experimental data of Solari and Bell [45]. No experimental data was provided for the Mini V-Grid valve tray.

Li et al. [29] simulated the three-dimensional flows for a new type of fixed valve tray using the same models proposed by Krishna et al. [22] with a few modification: a correlation based on experimental data was introduced into the momentum closure model; and the standard $k - \epsilon$ model [14] was applied to both liquid and gas phases. The macroscopic parameters—e.g., clear liquid height—were in good agreement with experimental measurements by the authors. However, validation of the details of the flow field—e.g., velocities, phase fractions—was not included in their study due to the lack of experimental data.

Jiang et al. [21] developed a two-phase three-dimensional model in an Eulerian-Eulerian framework to predict the hydrodynamics, mass-transfer behavior, and tray efficiency of ripple trays. The interphase momentum exchange terms was modelled as the drag force term. The value of the drag force was evaluated using different correlations [1, 5]. Turbulence properties were modelled using the shear stress transport (SST) model [34] for both the liquid and gas phases. Interphase mass transfer was evaluated using a model developed based on film theory [27, 37] and penetration theory [7, 15]. The CFD predictions were in good agreement with experimental measurements by the authors.

A summary of the reported CFD studies of distillation trays is given in Table 1.1. It can be observed that the majority of the simulations are conducted using two-phase three-dimensional models in an Eulerian-Eulerian framework with momentum and turbulence closure models for sieve trays. The simulated results of hydrodynamics on sieve trays are commonly validated using the experiments of Solari and Bell [45].

Table 1.1: Summary of literature on CFD studies for predicting the tray performance.

Reference	Multiphase model*	Closure models	Tray setup	CFD Software	Validation
Liu et al. [30]	Single-phase 2D, steady-state	Force term evaluated using [30], $k-\epsilon$ model [30] for liquid phase only	Sieve tray Φ=1.2 m	Not specified	[30]
Metha et al. [33]	Single-phase 3D, steady-state	Force term evaluated using [33], eddy diffusivity model [52] for liquid phase only	Sieve tray Φ=1.21 m	CFDS-FLOW 3D	[45]
Krishna et al. [22]	Two-phase 3D, transient	Drag force term evaluated using [5] Standard $k-\epsilon$ model [14] for liquid phase only	Sieve tray 0.39 m $\times$ 0.22 m	CFX 4.2	[22]

Continued on next page

Table 1.1 – continued from previous page

Reference	Multiphase model	Closure models	Tray setup	CFD Software	Validation
Krishna and Van Baten [24]	Two-phase 3D, transient	Drag force term evaluated using [5] Standard $k-\epsilon$ model [14] for liquid phase only	Sieve tray with catalyst containers Φ=1.2 m	CFX 4.2	[49]
Gesit et al. [10]	Two-phase 3D, transient	Drag force term evaluated using [5] Standard $k-\epsilon$ model [14] for liquid phase only	Sieve tray Φ=1.22 m	CFX 5.4	[45]
Zarei et al. [51]	Two-phase 3D, steady-state	Drag force term evaluated using [5] Standard $k-\epsilon$ model [14] for liquid phase only	Sieve tray and Mini V-Grid valve tray Φ=1.22 m	CFX 10.0	Only for sieve tray [45]

Continued on next page

Table 1.1 – continued from previous page

Reference	Multiphase model	Closure models	Tray setup	CFD Software	Validation
Li et al. [29]	Two-phase 3D, transient	Drag force term evaluated using [5] with correlation [29] Standard $k-\epsilon$ model [14] for both phases	Fixed Valve Tray Φ=0.54 m	CFX 12.0	[29]
Jiang et al. [21]	Two-phase 3D, steady-state	Drag force term evaluated using [1, 5] SST model [34] for both phases	Ripple tray Φ=0.31 m	CFX 13.0	[21]

* All multiphase models in the summarized studies are constructed in the Eulerian framework or Eulerian-Eulerian framework.

1.3 Thesis objectives

This research is intended to evaluate an Euler-Euler approach for the two-phase hydrodynamics simulation of sieve trays using the open-source software package, OpenFOAM®, with the following specific objectives:

- Evaluate the suitability of the Euler-Euler multiphase flow solver available in OpenFOAM® for sieve tray simulation.
- Determine appropriate closure models for sieve tray hydrodynamics simulation.
- Conduct sieve tray simulations and validate the results against experimental and simulation data obtained form published studies.

Chapter 2

Numerical Methodology

2.1 Governing equations

The two-fluid model is governed by two sets of conditional-averaged partial differential equations expressing the conservation of mass and momentum using the Eulerian approach for each fluid phase.

2.1.1 Eulerian approach

In the Eulerian approach, the description of the fluid field is focused on a certain fixed location in space and its properties change as time passes and fluid flows through that location. The instantaneous value of an arbitrary quantity ϕ is governed by the following equation:

$$\underbrace{\frac{\partial\left(\rho\phi\right)}{\partial t}}_{\text{accumulation}} = -\underbrace{\nabla\cdot\left(\rho\mathbf{v}\phi\right)}_{\text{convection}} + \underbrace{\nabla\cdot\left(\Gamma\nabla\phi\right)}_{\text{diffusion}} + \underbrace{S_\phi}_{\text{source}} \tag{2.1}$$

where ρ is local fluid density, $\mathbf{v}$ is the local fluid velocity vector, Γ is the diffusion coefficient of quantity ϕ, and S_ϕ represents the source or sink where quantity ϕ been created or consumed.

The instantaneous mass conservation equation can be derived by substituting ϕ with unity and recognizing that there is no generation/consumption or diffusion of total mass:

$$\frac{\partial\rho}{\partial t} = -\nabla\cdot\left(\rho\mathbf{v}\right) \tag{2.2}$$

The instantaneous momentum conservation equation, in which the work done by pressure and gravity have been taken into consideration, can be derived by substituting ϕ with $\mathbf{v}$ and accounting for the additional forces:

$$\frac{\partial\left(\rho\mathbf{v}\right)}{\partial t} = -\nabla\cdot\left(\rho\mathbf{v}\mathbf{v}\right) + \nabla\cdot\boldsymbol{\tau} - \nabla p + \rho\mathbf{g} \tag{2.3}$$

where p is fluid pressure, and $\mathbf{g}$ is gravitational acceleration. For a Newtonian fluid, the viscous stress tensor $\boldsymbol{\tau}$ is defined by the following expression:

$$\boldsymbol{\tau} = \mu \left[(\nabla \mathbf{v}) + (\nabla \mathbf{v})^T \right] - \frac{2}{3} \mu \left(\nabla \cdot \mathbf{v} \right) \mathbf{I} \tag{2.4}$$

where μ is the dynamic viscosity of the fluid, symbol T denotes the transpose operation, and $\mathbf{I}$ is the identity matrix.

Equations (2.2) and (2.3), also know as Navier–Stokes equations (NS equations), are capable of describing turbulent flows. However, it can be quite challenging to resolve all the details in turbulent flows using NS equations directly due to the requirements of extremely fine computational meshs and small time steps resulting unacceptably high computational costs.

2.1.2 Reynolds averaging

In practice, turbulent flows are usually described in a statistical fashion. One such a method, known as Reynolds averaging, was introduced by Osborne Reynolds in 1895. The principle of the Reynolds-averaging approach is to consider that, in a turbulent flow field, at any given time t and location $\mathbf{x}$ an arbitrary quantity ϕ can be decomposed into the average component $\bar{\phi}$ and its turbulent fluctuation component $\phi^{'}$:

$$\phi \left(t, \mathbf{x} \right) = \bar{\phi} \left(t, \mathbf{x} \right) + \phi^{'} \left(t, \mathbf{x} \right) \tag{2.5}$$

Substituting equation (2.5) into equation (2.1) and taking an ensemble average yields the general form of averaged conservation equations:

$$\frac{\partial \left(\bar{\rho} \bar{\phi} \right)}{\partial t} = -\nabla \cdot \left(\bar{\rho} \bar{\mathbf{v}} \bar{\phi} \right) + \nabla \cdot \left(\Gamma \nabla \bar{\phi} \right) + \bar{S_\phi} - \nabla \cdot \left(\rho \overline{\mathbf{v}' \phi'} \right) \tag{2.6}$$

The details of mathematical operations used during the averaging process are well described in literature [50]. Equation (2.6) has the same form as equation (2.1) but with an additional term, $-\nabla \cdot \left(\rho \overline{\mathbf{v}' \phi'} \right)$, which is introduced in the averaging process.

Substituting the expression of forms in equation (2.6) into equations (2.2) and (2.3) yields the Reynolds-averaged mass and momentum conservation equations:

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \left(\rho \bar{\mathbf{v}} \right) \tag{2.7}$$

$$\frac{\partial\left(\rho\bar{\mathbf{v}}\right)}{\partial t}=-\nabla\cdot\left(\rho\bar{\mathbf{v}}\bar{\mathbf{v}}\right)+\nabla\cdot\bar{\boldsymbol{\tau}}-\nabla\bar{p}+\rho\mathbf{g}-\nabla\cdot\left(\rho\overline{\mathbf{v}'\mathbf{v}'}\right) \tag{2.8}$$

In these equations, the term $-\rho\overline{\mathbf{v}'\mathbf{v}'}$ is defined as the Reynolds stress $\mathbf{R}$. Additionally, replacing the mean velocity $\bar{\mathbf{v}}$ and mean static pressure $\overline{p}$ with new notations $\mathbf{U}$ and P yields the following equations:

$$\frac{\partial\rho}{\partial t}=-\nabla\cdot\left(\rho\mathbf{U}\right) \tag{2.9}$$

$$\frac{\partial\left(\rho\mathbf{U}\right)}{\partial t}=-\nabla\cdot\left(\rho\mathbf{U}\mathbf{U}\right)+\nabla\cdot\bar{\boldsymbol{\tau}}-\nabla P+\rho\mathbf{g}+\nabla\cdot\mathbf{R} \tag{2.10}$$

Equations (2.9) and (2.10), also know as Reynolds-averaged Navier–Stokes equations (RANS equations), have similar forms as the NS equations as introduced in Section 2.1.1, with the variables now representing ensemble-averaged values. To represent the effects of turbulence, an additional term known as Reynolds stress tensor $\mathbf{R}$ was introduced into equation (2.10). This term cannot be calculated directly from the averaged field values and therefore requires models for closure.

2.1.3 Conditional-averaged two-phase flow equations

For a two-phase system, both phases can be treated as continua that are interpenetrating and represented by two sets of averaged conservation equations over the same fixed location in space[16]. It is essential to distinguish the amount of contribution to the averaged conservation equations by each individual phase as they are usually unevenly distributed at the same location. The solution to this problem is a technique known as conditional averaging developed based on the work of Dopazo [8] for intermittent turbulent flows.

The key concept of conditional averaging is a phase indicator function, $\chi_i\left(t,\mathbf{x}\right)$, which is defined as follows:

$$\chi_i\left(t,\mathbf{x}\right)=\begin{cases}1, & \text{if phase i presents at } \left(t,\mathbf{x}\right)\\ 0, & \text{if phase i does not present at } \left(t,\mathbf{x}\right)\end{cases} \tag{2.11}$$

The probability that a phase i is present at a certain location in space and time, which is represented by the phase fraction α_i is defined as the ensemble average of χ_i [16]:

$$\alpha_i=\lim_{N\to\infty}\frac{1}{N}\sum_{n=1}^{N}\chi_i \tag{2.12}$$

Multiplying the RANS equations by the ensemble-averaged phase indicator function for both phases in a two-phase system yields:

$$\frac{\partial\left(\alpha_c\rho_c\right)}{\partial t} = -\nabla\cdot\left(\alpha_c\rho_c\mathbf{U}_c\right) \tag{2.13}$$

$$\frac{\partial\left(\alpha_d\rho_d\right)}{\partial t} = -\nabla\cdot\left(\alpha_d\rho_d\mathbf{U}_d\right) \tag{2.14}$$

$$\frac{\partial\left(\alpha_c\rho_c\mathbf{U}_c\right)}{\partial t} = -\nabla\cdot\left(\alpha_c\rho_c\mathbf{U}_c\mathbf{U}_c\right)+\nabla\cdot\left(\alpha_c\bar{\boldsymbol{\tau}}_c\right)-\alpha_c\nabla P+\alpha_c\rho_c\mathbf{g}+\nabla\cdot\left(\alpha_c\mathbf{R}_c\right)-\mathbf{M} \tag{2.15}$$

$$\frac{\partial\left(\alpha_d\rho_d\mathbf{U}_d\right)}{\partial t} = -\nabla\cdot\left(\alpha_d\rho_d\mathbf{U}_d\mathbf{U}_d\right)+\nabla\cdot\left(\alpha_d\bar{\boldsymbol{\tau}}_d\right)-\alpha_d\nabla P+\alpha_d\rho_d\mathbf{g}+\nabla\cdot\left(\alpha_d\mathbf{R}_d\right)+\mathbf{M} \tag{2.16}$$

where subscript c and d denote the continuous and dispersed phases, $\mathbf{g}$ is gravitational acceleration, and $\mathbf{M}$ represents the momentum transfer between the two phases due to interphase forces. The momentum transfer terms in equations (2.15) and (2.16) have opposite signs, indicating that these are internal forces in the system.

Equations (2.13), (2.14), (2.15), and (2.16) are the governing equations for two-fluid Model. The terms of Reynolds stress tensor $\mathbf{R}$ and interphase momentum transfer term $\mathbf{M}$ are introduced through the averaging process and require modelling, which will be discussed in detail in the subsequent sections.

2.2 Momentum transfer closure model

As shown in equations (2.15) and (2.16), a source term, $\mathbf{M}$, is introduced into the momentum equations to represent the momentum transfer due to forces acting between the continuous and dispersed phases. Depending on the physics involved, there are variety of models and formulations available in literature [23, 31, 32, 42]. Some of the most commonly utilized momentum transfer terms in bubbly flows are listed below:

- Drag force term, $\mathbf{M}_D$: Momentum transfer due to the force acts on the bubble in the direction opposite to the bubble motion with respect to surrounding continuous phase.

- Lift force term, $\mathbf{M}_L$: Momentum transfer due to the force acts on the bubble perpendicular to the bubble motion with respect to surrounding continuous phase.

- Virtual mass force term, $\mathbf{M}_{VM}$: Momentum transfer due to the force required to accelerate the mass of the surrounding continuous phase as an accelerating bubble moves through it.

- Turbulent dispersion force term, $\mathbf{M}_{TD}$: Momentum transfer due to the force describes the spreading/diffusion of the dispersed phase due to turbulent fluctuations in the continuous phase.

- Wall lubrication force term, $\mathbf{M}_{WL}$: Momentum transfer due to the force acts on a bubble near a wall to prevent the bubble from touching the wall.

The interphase momentum transfer term is given by the sum of the momentum transfer components caused by all the forces considered:

$$\mathbf{M} = \mathbf{M}_D + \mathbf{M}_L + \mathbf{M}_{VM} + \mathbf{M}_{TD} + \mathbf{M}_{WL} \tag{2.17}$$

Among these forces, the drag force usually dominates interphase momentum exchange [22]. Therefore, only the drag force was considered in this study.

The momentum transfer due to drag force acting on a bubble can be calculated from the following equation:

$$\mathbf{M}_D = \frac{3}{4}\alpha_d \rho_c \frac{C_d}{d} \left|\mathbf{U}_r\right| \mathbf{U}_r \tag{2.18}$$

where d is the bubble diameter, $\mathbf{U}_r = \mathbf{U}_d - \mathbf{U}_c$ represents the relative velocity between the two phases, and C_d is the drag coefficient that requires modelling.

For a distillation column operating in the churn-turbulent regime, it can be challenging to determine the drag force on bubbles for the following reasons [23]:

- Bubbles breakup and coalesce resulting in non-uniform bubble sizes.

- Large bubbles have non spherical shapes.

- Bubbles do not necessarily have well defined boundaries because gas jets break up in the liquid to form bubbles, and the gas phase becomes the continuous phase above the froth on the distillation tray.

Many studies have been conducted to try to improve the understanding of this flow pattern and develop a model to predict the forces in such conditions; however, the challenge still remains largely open. A comprehensive literature review on this topic can be found in [35].

In this work, the correlation proposed by Krishna et al. [23] was adopted to estimate the drag coefficient acting on a group of bubbles in the churn-turbulent regime:

$$C_d = \frac{4}{3}\frac{\rho_c - \rho_d}{\rho_c}\mathbf{g}\frac{d}{\mathbf{U}_{slip}^2} \tag{2.19}$$

where $\mathbf{U}_{slip}$ is the relative velocity of the bubbles with respect to the liquid. When the tray reaches the quasi-steady-state condition, $\mathbf{U}_{slip}$ is given by the following expression:

$$\mathbf{U}_{slip} = \frac{\mathbf{U}_s}{\alpha_d^B} \tag{2.20}$$

where $\mathbf{U}_s$ is the superficial gas velocity, and α_d^B is the average gas hold-up in the froth when system operates at a quasi-steady-state condition.

Substituting equation (2.20) into equation (2.19) yields:

$$C_d = \frac{4}{3}\frac{\rho_c - \rho_d}{\rho_c}\mathbf{g}d\left(\frac{\alpha_d^B}{\mathbf{U}_s}\right)^2 \tag{2.21}$$

In this work, the correlation established by Bennett et al. [5] was used to estimate the value of α_d^B:

$$\alpha_d^B = 1 - \exp\left[-12.55\left(\mathbf{U}_s\sqrt{\frac{\rho_d}{\rho_c - \rho_d}}\right)^{0.91}\right] \tag{2.22}$$

Substituting equations (2.20) and (2.22) into equation (2.19) yields:

$$C_d = \frac{4}{3}\frac{\rho_c - \rho_d}{\rho_c}\mathbf{g}d\left(\frac{1 - \exp\left[-12.55\left(\mathbf{U}_s\sqrt{\frac{\rho_d}{\rho_c - \rho_d}}\right)^{0.91}\right]}{\mathbf{U}_s}\right)^2 \tag{2.23}$$

Therefore, the interphase momentum transfer term due to drag force can be evaluated by Substituting equation (2.23) into equation (2.18). The resulting formulation is:

$$\mathbf{M}_D = \alpha_d\left(\rho_c - \rho_d\right)\mathbf{g}\left(\frac{1 - \exp\left[-12.55\left(U_s\sqrt{\frac{\rho_d}{\rho_c - \rho_d}}\right)^{0.91}\right]}{U_s}\right)^2 \left|\mathbf{U}_r\right|\mathbf{U}_r \tag{2.24}$$

This formulation is convenient for distillation tray fluid dynamics simulations because it allows the drag force to be calculated without information about bubble diameter d, which is difficult to obtain locally in the churn-turbulent regime.

2.3 Turbulence closure model

As mentioned in Section 2.1.2, the Reynolds stress tensor $\mathbf{R}$ is introduced from the averaging operation of the momentum conservation equations to account for turbulent fluctuations. This term must be appropriately modelled to close the RANS equations.

One of the most common theories that is employed to approximate this unknown term is the Boussinesq hypothesis [6]. Joseph Boussinesq proposed this theory to correlate $\mathbf{R}$ as a linear function of the known deformation rate tensor with the coefficient defined as eddy viscosity:

$$\mathbf{R} = -\rho\overline{\mathbf{v}'\mathbf{v}'} = \mu_t\left[(\nabla\mathbf{U}) + (\nabla\mathbf{U})^T - \frac{2}{3}\mu_t\left(\nabla\cdot\mathbf{U}\right)\mathbf{I}\right] - \frac{2}{3}\overline{\rho}k\mathbf{I} \tag{2.25}$$

where k is turbulent kinetic energy which is defined as follows:

$$k = \frac{1}{2}\overline{\mathbf{U}'\mathbf{U}'} \tag{2.26}$$

the term $2/3\overline{\rho}k\mathbf{I}$ is introduced to maintain the proper trace of the Reynolds stress tensor [50], and μ_t is the eddy viscosity. Unlike the dynamic viscosity of a fluid μ, the value of μ_t is not a constant for a given fluid but rather a quantity that changes with position and status of the flow field. The Boussinesq hypothesis eliminates the unknown Reynolds stress tensor but also introduces new unknowns k and μ_t. The relationship between these two quantities can be described using the following assumption:

$$\mu_t = C_\mu\rho\sqrt{k}L \tag{2.27}$$

where C_μ is a dimensionless constant, and L is the characteristic length of the turbulence. Now, the challenge is relating k and L with known quantities using turbulence models.

In engineering applications, the most commonly used turbulence models are called two-equation models, in which two transport equations are used to calculate the turbulent kinetic energy and characteristic length separately. The well known standard $k-\epsilon$ model [26] falls into this two-equation turbulence model category.

In the standard $k-\epsilon$ model, the turbulent kinetic energy k is obtained by solving the following equation:

$$\frac{\partial\left(\rho k\right)}{\partial t} + \mathbf{U}\cdot\nabla\left(\rho k\right) - \nabla\cdot\left[\left(\frac{\mu_{eff}}{\mathbf{Sc}_k}\right)\nabla k\right] = G_k - \rho\epsilon + S_k \tag{2.28}$$

where μ_{eff} is the effective viscosity, which is calculated as the sum of the dynamic and eddy viscosities:

$$\mu_{eff} = \mu + \mu_t \tag{2.29}$$

It should be mentioned that μ_{eff} will be used in lieu of μ in the momentum equations when a turbulence model is employed. Furthermore, $\mathbf{Sc}_k$ denotes the turbulent Schmidt number for k, G_k is the production of the turbulent kinetic energy due to the non-isotropic part of the Reynolds stress tensor, defined as follows:

$$G_k = \mu_t \left[(\nabla \mathbf{U}) + (\nabla \mathbf{U})^T - \frac{2}{3} (\nabla \cdot \mathbf{U}) \mathbf{I} \right] : \nabla \mathbf{U} \tag{2.30}$$

ϵ is the dissipation rate of k, and S_k represents other sources of turbulent kinetic energy.

In turbulent flows, the kinetic energy of the turbulence will eventually transfer into internal energy due to viscous dissipation. Based on this dissipation phenomena, k and ϵ are related by the following equation:

$$L = \frac{k^{3/2}}{\epsilon} \tag{2.31}$$

Thus, the expression for the eddy viscosity μ_t can be written as follows:

$$\mu_t = C_\mu \rho \sqrt{k} L = C_\mu \rho \sqrt{k} \frac{k^{3/2}}{\epsilon} = C_\mu \rho \frac{k^2}{\epsilon} \tag{2.32}$$

To close the standard $k - \epsilon$ turbulence model, the following equation is employed to obtain the value of ϵ:

$$\frac{\partial (\rho \epsilon)}{\partial t} + \mathbf{U} \cdot \nabla (\rho \epsilon) - \nabla \cdot \left[\left(\frac{\mu_{eff}}{\mathbf{Sc}_\epsilon} \right) \nabla \epsilon \right] = C_1 G_k \frac{\epsilon}{k} - C_2 \rho \frac{\epsilon^2}{k} + S_\epsilon \tag{2.33}$$

where $\mathbf{Sc}_\epsilon$ denotes the turbulent Schmidt number for ϵ, C_1 and C_2 are constants, and S_ϵ represents other sources for the rate of turbulent kinetic energy dissipation.

The model parameters for equations (2.28) and (2.33) have been determined by data fitting for a wide range of turbulent flows [26] and their values are listed in Table 2.1.

Equations (2.28) and (2.33) are the components of the single-phase standard $k-\epsilon$ turbulence model. As presented in Section 1.2, Most of the investigated previous simulation works have employed the standard $k-\epsilon$ turbulence model for liquid phase only and modelled the gas phase as laminar flow. This is based on the assumption that

Table 2.1: Coefficients in standard $k - \epsilon$ turbulence model.

C_μ	C_1	C_2	$\mathbf{Sc}_k$	$\mathbf{Sc}_\epsilon$
0.09	1.44	1.92	1.00	1.30

turbulence is dominated by the continuous phase; therefore, only the continuous phase turbulent kinetic energy k_c needs to be solved. The dispersed phase turbulent kinetic energy k_d will be determined using k_c along with a response coefficient [26]. However, it has been shown by the study of Garnier et al. [9] that the dominance of continuous phase on turbulence is less significant at high dispersed phase fractions. Moreover, in regions where the continuous phase vanishes, the dispersed phase actually becomes the continuous phase. This situation, which is known as phase inversion, introduces additional numerical problems when solving the governing equations.

In the present study, a mixture $k - \epsilon$ model for two-phase flow developed by Behzadi [2] has been adopted for turbulence modelling. The model is based on the theory that at high phase fraction, both phases fluctuate as one mixed phase and the transport equations are solved using the mixture properties for k_m and ϵ_m:

$$\frac{\partial(\rho_m k_m)}{\partial t} + \mathbf{U}_m \cdot \nabla(\rho_m k_m) - \nabla \cdot \left[\left(\frac{\mu_{t,m}}{\mathbf{Sc}_{k,m}}\right)\nabla k_m\right] = G_{k,m} - \rho_m \epsilon_m + S_{k,m} \tag{2.34}$$

$$\frac{\partial(\rho_m \epsilon_m)}{\partial t} + \mathbf{U}_m \cdot \nabla(\rho_m \epsilon_m) - \nabla \cdot \left[\left(\frac{\mu_{t,m}}{\mathbf{Sc}_{\epsilon,m}}\right)\nabla \epsilon_m\right] = C_1 G_{k,m}\frac{\epsilon_m}{k_m} - C_2 \rho_m \frac{{\epsilon_m}^2}{k_m} + S_{\epsilon,m} \tag{2.35}$$

The values of model parameters in equations (2.34) and (2.35) are left unchanged from those in the standard $k - \epsilon$ model and are presented in Table 2.2.

Table 2.2: Coefficients in mixture $k - \epsilon$ turbulence model.

C_μ	C_1	C_2	$\mathbf{Sc}_{k,m}$	$\mathbf{Sc}_{\epsilon,m}$
0.09	1.44	1.92	1.00	1.30

The mixture properties appearing in equations (2.34) and (2.35) are defined as follows:

$$\rho_m = \alpha_c \rho_c + \alpha_d \rho_d \tag{2.36}$$

$$G_{k,m} = \alpha_c G_{k,c} + \alpha_d G_{k,d} \tag{2.37}$$

$$k_m = \left(\alpha_c \frac{\rho_c}{\rho_m} + \alpha_d \frac{\rho_d}{\rho_m} {C_t}^2\right) k_c \tag{2.38}$$

$$\epsilon_m = \left(\alpha_c \frac{\rho_c}{\rho_m} + \alpha_d \frac{\rho_d}{\rho_m} {C_t}^2\right) \epsilon_c \tag{2.39}$$

$$\mathbf{U}_m = \frac{\alpha_c \rho_c \mathbf{U}_c + \alpha_d \rho_d \mathbf{U}_d {C_t}^2}{\alpha_c \rho_c + \alpha_d \rho_d {C_t}^2} \tag{2.40}$$

$$\mu_{t,m} = \frac{\rho_m \left(\alpha_c \mu_{t,c} + \alpha_d \mu_{t,d} {C_t}^2\right)}{\alpha_c \rho_c + \alpha_d \rho_d {C_t}^2} \tag{2.41}$$

$$\mu_{t,d} = C_t^2 \left(\frac{\nu_c}{\nu_d} \frac{\rho_d}{\rho_c}\right) \mu_{t,c} \tag{2.42}$$

the new parameter C_t denotes the turbulence response coefficient [12], which is defined as follows:

$$C_t = \frac{{\mathbf{U}_d}'}{{\mathbf{U}_c}'} \tag{2.43}$$

It is obvious that the value of C_t is required to evaluate equations (2.38) to (2.42). In this study, the procedure used to obtain value of C_t is based on the model originally proposed by Issa [18] and the modification proposed by Rusche [41] to account for correlation with the dispersed phase fraction has been included. The model is summarized as follows:

$$C_t = 1 + (C_{t,0} - 1) \exp(-f(\alpha_d)) \tag{2.44}$$

$$f(\alpha_d) = 180\alpha_d - 4.71 \times 10^3 {\alpha_d}^2 + 4.26 \times 10^4 {\alpha_d}^3 \tag{2.45}$$

$$C_{t,0} = \frac{3 + \beta}{1 + \beta + 2\rho_d/\rho_c} \tag{2.46}$$

$$\beta = \frac{2 A_d L_e^2}{\rho_c \nu_c \mathbf{Re}_t} \tag{2.47}$$

$$A_d = \frac{3 \alpha_d \rho_c C_d \left|\mathbf{U}_r\right|}{4d} \tag{2.48}$$

$$L_e = C_\mu \frac{k_c^{3/2}}{\epsilon_c} \tag{2.49}$$

$$\mathbf{Re}_t = \frac{\sqrt{\frac{2k_c}{3}} L_e}{\nu_c} \tag{2.50}$$

where L_e is the eddy length scale, and $\mathbf{Re}_t$ is the turbulence Reynolds number.

2.4 The reactingTwoPhaseEulerFoam solver in OpenFOAM

A summary of the implementation of the finite volume method in OpenFOAM® is provided in A. In this study, the OpenFOAM® version 6 was used, and the Euler-Euler solver `reactingTwoPhaseEulerFoam` was employed. The source code for this solver is located in the `OpenFOAM-6/applications/solvers/multiphase/reactingEulerFoam/reactingTwoPhaseEulerFoam` directory. The solver is described in the `reactingTwoPhaseEulerFoam.C` file as follows:

Solver for a system of 2 compressible fluid phases with a common pressure, but otherwise separate properties. The type of phase model is run time selectable and can optionally represent multiple species and in-phase reactions. The phase system is also run time selectable and can optionally represent different types of momentum, heat and mass transfer.

The following topics are discussed in this section:

- How the set of governing equations are implemented and solved
- How the drag model is implemented in this study.
- How the turbulence closure model is implemented and solved.

2.4.1 Solver algorithm

The `reactingTwoPhaseEulerFoam` solver employs the `PIMPLE` algorithm, which is a combination of the `PISO` [19] and `SIMPLE` [40] algorithms, to obtain a segregated solution of the pressure and velocity fields. The pressure-implicit with splitting of operators (`PISO`) algorithm is suitable for transient simulations, but the time step size is constrained to a Courant number of less than 1 ($C_o < 1$). The semi-implicit method for pressure-linked equations (`SIMPLE`) algorithm is suitable for steady-state simulations. The `PIMPLE` algorithm is a combination of both `PISO` and `SIMPLE`, which is suitable for transient simulations with less constraints on time step size than imposed by `PISO`.

As shown in equation (A.7), the accumulation term is discretized through a finite-difference approximation, which allows the flow field information to be obtained at a certain time point. When the flow is transient, using a large time step will cause too large of a deference of flow fields between two adjacent time steps, which can make the `PISO` algorithm unstable and/or inaccurate. However, using a small time step means that more computational resources are required, which is always unfavourable.

The principle of the `PIMPLE` algorithm is to use the `SIMPLE` algorithm to perform iterations within each time step to converge the pressure and velocity fields to a desired tolerance. Meanwhile, the `PISO` algorithm is used in an inner loop to advance the solution in time. This combination of inner and outer looping structures within each time step is intended to maintain stability for larger time steps, while also facilitating error control. Figure 2.1 illustrates the structure of the `PIMPLE` algorithm implemented in the `reactingTwoPhaseEulerFoam` solver in OpenFOAM® version 6. In Figure 2.1, `nOuterCorr` stands for the number of outer loops for pressure-momentum coupling; `nCorr` stands for the number of inner loops for pressure correction; `nNonOrthoCorr` stands for the number of loops for pressure gradient correction due to non-orthogonal mesh; and `T` is the end time of the simulation given by the user.

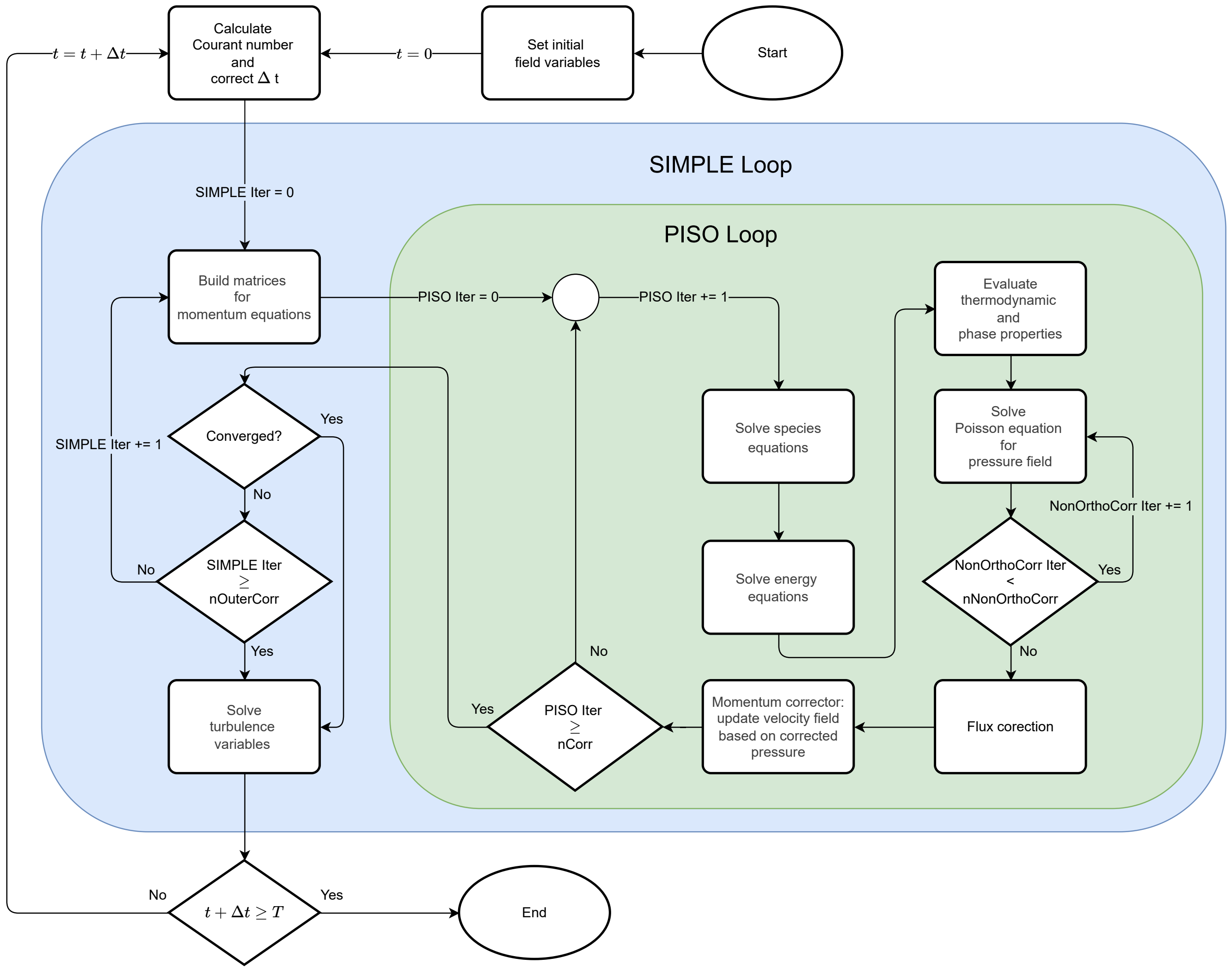

Start
Set initial field variables
$t = 0$
Calculate Courant number and correct Δ t
$t = t + \Delta t$
SIMPLE Loop
SIMPLE Iter = 0
Build matrices for momentum equations
PISO Loop
PISO Iter = 0
PISO Iter += 1
Solve species equations
Solve energy equations
Evaluate thermodynamic and phase properties
Solve Poisson equation for pressure field
NonOrthoCorr Iter += 1
NonOrthoCorr Iter < nNonOrthoCorr
Yes
No
Flux corection
Momentum corrector: update velocity field based on corrected pressure
PISO Iter ≥ nCorr
Yes
No
Converged?
Yes
No
SIMPLE Iter += 1
SIMPLE Iter ≥ nOuterCorr
No
Yes
Solve turbulence variables
$t + \Delta t \geq T$
No
Yes
End

Figure 2.1: Flowchart of the `PIMPLE` algorithm in OpenFOAM®.

The application of this algorithm to the two-fluid model and its sub models is described below.

First, given the similarities in terms of formulation between the viscous stress tensor $\boldsymbol{\tau}$ (see equation (2.4)) and the Reynolds stress tensor $\mathbf{R}$ (see equation (2.25)), it is possible to merge these two terms into a new variable named effective stress $\mathbf{R}_{eff}$, which is given by the following expression:

$$\mathbf{R}_{eff} = \frac{\boldsymbol{\tau} + \mathbf{R}}{\rho} \tag{2.51}$$

and must be closed with a turbulence model.

As stated in Section 2.2, the only interphase force been considered in this study is the drag force, therefor:

$$\mathbf{M} = \mathbf{M}_D \tag{2.52}$$

The drag force term can be split into explicit and implicit contributions as follows:

$$\mathbf{M}_D = K_d|\mathbf{U}_r|\mathbf{U}_d - K_d|\mathbf{U}_r|\mathbf{U}_c \tag{2.53}$$

where $K_d = \frac{3}{4}\alpha_d\rho_c\frac{C_d}{d}$. The purpose of splitting the drag force is that the implicit term promotes diagonal dominance of the resulting matrix after discretization [28], which facilitates convergence when solving the system of algebraic equations.

Substituting equations (2.51), (2.52), and (2.53) into the momentum conservation equations (2.15) and (2.16) yields:

$$\begin{aligned}\frac{\partial\left(\alpha_c\rho_c\mathbf{U}_c\right)}{\partial t} &+ \nabla\cdot\left(\alpha_c\rho_c\mathbf{U}_c\mathbf{U}_c\right) - \nabla\cdot\left(\alpha_c\rho_c\mathbf{R}_{eff,c}\right)\\ &= -\alpha_c\nabla P + \alpha_c\rho_c\mathbf{g} - K_d\mathbf{U}_r\mathbf{U}_c + K_d|\mathbf{U}_r|\mathbf{U}_d\end{aligned} \tag{2.54}$$

$$\begin{aligned}\frac{\partial\left(\alpha_d\rho_d\mathbf{U}_d\right)}{\partial t} &+ \nabla\cdot\left(\alpha_d\rho_d\mathbf{U}_d\mathbf{U}_d\right) - \nabla\cdot\left(\alpha_d\rho_d\mathbf{R}_{eff,d}\right)\\ &= -\alpha_d\nabla P + \alpha_d\rho_d\mathbf{g} - K_d\mathbf{U}_r\mathbf{U}_d + K_d|\mathbf{U}_r|\mathbf{U}_c\end{aligned} \tag{2.55}$$

Performing a partial FVM discretization on equations (2.54) and (2.55) results in an expression that can be presented in a matrix form similar to equation (A.12):

$$a_{Q,c}\mathbf{U}_{Q,c} = H(\mathbf{U}_c^n) - \frac{\alpha_c\nabla P'}{\rho_c} + \frac{Kd}{\rho_c}|\mathbf{U}_r^o|\mathbf{U}_d^o \tag{2.56}$$

$$a_{Q,d}\mathbf{U}_{Q,d} = H(\mathbf{U}_d) - \frac{\alpha_d \nabla P'}{\rho_d} + \frac{Kd}{\rho_d}|\mathbf{U}_r^o|\mathbf{U}_c^o \tag{2.57}$$

where $H()$ is an operator that accounts for the contribution of all terms except the pressure gradient, explicit part of the drag, and gravitational term from the momentum equations [20], and $P' = P - \rho g h$ is a modified pressure. This step corresponds to the "Build matrices for momentum equations" step in Figure 2.1. Equations (2.56) and (2.57) can be further rearranged as follows:

$$\mathbf{U}_c = \frac{H(\mathbf{U}_c)}{a_{Q,c}} - \frac{1}{a_{Q,c}}\left(\frac{\alpha_c \nabla P'}{\rho_c} + \frac{K_d}{\rho_c}|\mathbf{U}_r^o|\mathbf{U}_d^o\right) \tag{2.58}$$

$$\mathbf{U}_d = \frac{H(\mathbf{U}_c)}{a_{P,d}} - \frac{1}{a_{P,d}}\left(\frac{\alpha_d \nabla P'}{\rho_d} + \frac{K_d}{\rho_d}|\mathbf{U}_r^o|\mathbf{U}_c^o\right) \tag{2.59}$$

The phase continuity equations (2.13) and (2.14), can be rearranged to give the following equations:

$$\begin{aligned}
&\frac{\partial(\alpha_c\rho_c)}{\partial t} + \nabla\cdot(\alpha_c\rho_c\mathbf{U}_c) \\
&= \alpha_c\frac{\partial(\rho_c)}{\partial t} + \rho_c\frac{\partial(\alpha_c)}{\partial t} + \alpha_c\mathbf{U}_c\cdot\nabla(\rho_c) + \rho_c\nabla\cdot\ (\alpha_c\mathbf{U}_c) \\
&= \alpha_c\frac{D(\rho_c)}{Dt} + \rho_c\frac{\partial(\alpha_c)}{\partial t} + \rho_c\nabla\cdot(\alpha_c\mathbf{U}_c)
\end{aligned} \tag{2.60}$$

$$\begin{aligned}
&\frac{\partial(\alpha_d\rho_d)}{\partial t} + \nabla\cdot(\alpha_d\rho_d\mathbf{U}_d) \\
&= \alpha_d\frac{\partial(\rho_d)}{\partial t} + \rho_d\frac{\partial(\alpha_d)}{\partial t} + \alpha_d\mathbf{U}_d\cdot\nabla(\rho_d) + \rho_d\nabla\cdot\ (\alpha_d\mathbf{U}_d) \\
&= \alpha_d\frac{D(\rho_d)}{Dt} + \rho_d\frac{\partial(\alpha_d)}{\partial t} + \rho_d\nabla\cdot(\alpha_d\mathbf{U}_d)
\end{aligned} \tag{2.61}$$

These equations can be further rearranged as follows:

$$\nabla\cdot(\alpha_c\mathbf{U}_c) = -\frac{\alpha_c}{\rho_c}\frac{D(\rho_c)}{Dt} - \frac{\partial(\alpha_c)}{\partial t} \tag{2.62}$$

$$\nabla\cdot(\alpha_d\mathbf{U}_d) = -\frac{\alpha_d}{\rho_d}\frac{D(\rho_d)}{Dt} - \frac{\partial(\alpha_d)}{\partial t} \tag{2.63}$$

Summing these two equations and recognizing that $\frac{\partial(\alpha_c+\alpha_d)}{\partial t} = 0$ yields:

$$\nabla\cdot(\alpha_c\mathbf{U}_c + \alpha_d\mathbf{U}_d) = -\frac{\alpha_c}{\rho_c}\frac{D\rho_c}{Dt} - \frac{\alpha_d}{\rho_d}\frac{D\rho_d}{Dt} \tag{2.64}$$

Equation (2.64) is equivalent to the continuity equations in the two-fluid model and it provides a constraint for volume conservation [41].

The pressure Poisson equation can be constructed by substituting equations (2.58) and (2.59) into equation (2.64), which gives the following result:

$$\begin{aligned}
&\nabla \cdot \left[\alpha_c \left(\frac{H(\mathbf{U}_c)}{a_{P,c}} + \frac{Kd}{a_{P,c}\rho_c}|\mathbf{U}_r^o|\mathbf{U}_d^o\right) + \alpha_d \left(\frac{H(\mathbf{U}_d)}{a_{P,d}} + \frac{Kd}{a_{P,d}\rho_d}|\mathbf{U}_r^o|\mathbf{U}_c^o\right)\right] \\
&\quad + \left(\frac{\alpha_c}{\rho_c}\frac{D\rho_c}{Dt} + \frac{\alpha_d}{\rho_d}\frac{D\rho_d}{Dt}\right) \\
&\quad = \nabla \cdot \left[(\frac{\alpha_a}{a_{p,a}\rho_a} + \frac{\alpha_b}{a_{p,b}\rho_b})\nabla P'\right]
\end{aligned} \tag{2.65}$$

Equation (2.65) can be solved for the updated pressure field. This step corresponds to the "Solve Poisson equation" step in the `PIMPLE` algorithm shown in Figure 2.1.

In OpenFOAM®, the gradient terms in equation (2.65) are transformed into a summation of face fluxes using the Gauss theorem. The faces fluxes are obtained through iterations and corrected in the "Flux correction" step in in Figure 2.1. The corrected fluxes will then be used to correct the velocities to obtain a set of new velocities that satisfy the continuity constraint imposed by equation (2.64). This step corresponds to the "Momentum corrector" step in Figure 2.1.

2.4.2 Drag model implementation

The `reactingTwoPhaseEulerFoam` solver provides a set of common drag models. However, the drag model employed in this study is not provided and therefore needs to be implemented by the user. The procedure for adding new sub-models is described in detail by Norouzi [38]. The discussion in this section mainly focuses on the numerical treatment applied to the new drag model.

In OpenFOAM®, the drag force coefficient is returned by a base class `dragModel` as `K()`. The interphase momentum transfer term due to the drag force is then calculated as follows:

$$\mathbf{M}_D = K\mathbf{U}_r \tag{2.66}$$

The source code of `dragModel` is available in the `OpenFOAM-6/applications/solvers/multiphase/reactingEulerFoam/interfacialModels/dragModels/dragModel/dragModel.C` file. The relevant part of the code is given in Figure 2.2.

Thus, the code of `dragModel.C` is given by the following mathematical expression:

$$K = \max(\alpha_d, \alpha_{d,residual})\frac{3}{4}C_d\mathbf{Re}C_s\rho_c\nu_c\frac{1}{d^2} \tag{2.67}$$

```
Foam::tmp<Foam::volScalarField> Foam::dragModel::Ki() const
{
    return
         0.75
        *CdRe()
        *swarmCorrection_->Cs()
        *pair_.continuous().rho()
        *pair_.continuous().nu()
        /sqr(pair_.dispersed().d());
}


Foam::tmp<Foam::volScalarField> Foam::dragModel::K() const
{
    return max(pair_.dispersed(), pair_.dispersed().residualAlpha())*Ki();
}
```

Figure 2.2: The definition of drag coefficient `K()` in `dragModel.C`.

```
Foam::tmp<Foam::volScalarField> Foam::phasePair::magUr() const
{
    return mag(phase1().U() - phase2().U());
}

Foam::tmp<Foam::volScalarField> Foam::phasePair::Re() const
{
    return magUr()*dispersed().d()/continuous().nu();
}

```

Figure 2.3: The definition of bubble Reynolds number `Re` in `phasePair.C`.

where C_s is the swarm correction coefficient that can be used to correct the value of the drag coefficient C_d to account for the effect of a dense swarm of rising bubbles [48]. In this study, no swarm correction is employed, and therefore `Cs()` returns value of unity. **Re** is the bubble Reynolds number, which is calculated as shown in Figure 2.3 and the code can be found in the `OpenFOAM-6/applications/solvers/multiphase/reactingEulerFoam/phaseSystems/phasePair/phasePair/phasePair.C` file.

The code in Figure 2.3 can be translated into the following mathematical equation:

$$\mathbf{Re} = \frac{|\mathbf{U}_c - \mathbf{U}_d|\, d}{\nu_c} \tag{2.68}$$

```
Foam::tmp<Foam::volScalarField>
Foam::dragModels::Krishna::CdRe() const
{
    dimensionedScalar unitconversion
    (
        "unitconversion",dimensionSet(0, -1, 0, 0, 0, 0, 0),1
    );

    return
        (4.0/3.0)
        *(rhoc_-rhod_)/rhoc_
        *9.81
        *pair_.dispersed().d()
        *unitconversion
        *sqr((1-exp(-12.55*pow(Us_*sqrt((rhod_/(rhoc_-rhod_))),0.91)))/Us_)
        *max(pair_.continuous(), scalar(1.0e-6))
        /exp(-12.55*pow(Us_*sqrt((rhod_/(rhoc_-rhod_))),0.91))
        *max(pair_.Re(), residualRe_);
}

```

Figure 2.4: The definition of the drag coefficient in `Krishna.C`.

Substituting equations (2.67) (with C_s=1) and (2.68) into equation (2.66) yields:

$$\begin{aligned}\mathbf{M}_D &= \max(\alpha_d, \alpha_{d,residual})\frac{3}{4}C_d\frac{|\mathbf{U}_c-\mathbf{U}_d|\,d}{\nu_c}C_s\rho_c\nu_c\frac{1}{d^2}\mathbf{U}_r \\ &= \frac{3}{4}\max(\alpha_d, \alpha_{d,residual})\rho_c\frac{C_d}{d}\left|\mathbf{U}_r\right|\mathbf{U}_r\end{aligned} \tag{2.69}$$

Equation (2.69) is in agreement with the conventional formula used to evaluate the drag force given as equation (2.18) with a minor modification on the phase fraction term. $\alpha_{d,residual}$ is a small user-defined value to set the lower limit of phase fraction when calculating drag force. This modification prevents the value of drag force from approaching zero in regions where the continuous phase fraction approaches zero [28].

At line 5 in Figure 2.2, the function named `CdRe()` is called, which returns the product of the drag coefficient C_d and the bubble Reynolds number **Re**. The value returned by this function call depends on the drag model from which C_d is obtained. In this study, the drag coefficient correlation by Krishna et al. [23] was implemented into OpenFOAM®. The relevant source code for this correlation was implemented and the relevant code is shown in Figure 2.4

Translating the code in Figure 2.4 into mathematical form and removing the term

Re yields the following equation:

$$C_d = \frac{4}{3}\frac{\rho_c - \rho_d}{\rho_c}\mathbf{g}d\left(\frac{\alpha_d^B}{\mathbf{U}_s}\right)^2\frac{max(\alpha_c, \alpha_{c,residual})}{1-\alpha_d^B} \tag{2.70}$$

where $\alpha_{c,residual}$ was set to be 1.0×10^{-6} and α_d^B is calculated using equation (2.22). The implemented drag model in equation (2.70) is derived from the theoretical formulation in equation (2.21) by multiplying a term that serves as a weighting factor. This modification, which is similar to the one implemented by Krishna et al. [22], is found to be essential for obtaining reasonable simulation results. A study case on this term is presented in Section 3.4.

2.4.3 Turbulence model implementation

In OpenFOAM® version 6, a built-in turbulence model named `mixtureKEpsilon` is provided. This is the model that was used in the present work. The source code can be found in the `OpenFOAM-6/src/TurbulenceModels/phaseCompressible/RAS/mixtureKEpsilon` directory. It should be noted that the source code is not in exactly the same form as the model provided by Behzadi [2] in Section 2.3. The main differences are described in Appendix B.

Chapter 3

Case Studies

In this chapter, a series of case studies was conducted using the OpenFOAM® solver `reactingTwoPhaseEulerFoam` to evaluate its capacity to simulate the multiphase flow on a simplified sieve tray configuration with a variety of settings.

3.1 Base case setup

3.1.1 OpenFOAM case structure

One of the advantages of using OpenFOAM® over commercial CFD software packages is that the user has the control over almost every detail of the computing process and data flow. But, on the other hand, it also requires the user to have a comprehensive understanding of the physics of problem, proper modelling, and solving algorithm control as well as the OpenFOAM® case structure itself. A basic file structure for an OpenFOAM® case using the `reactingTwoPhaseEulerFoam` solver that contains the minimum set of files required to run a sieve tray simulation is presented in Figure 3.1 and described as follows:

- A `0` directory that contains a set of individual files providing data of initial values and boundary conditions for every flow field of interest—e.g., pressure, velocity—to initialise the simulation. Those data must be specified by the user. The name `0` comes from the fact that simulations usually start at time $t = 0$. OpenFOAM® writes simulation results into directories that share the same structure of `0` directory and named based on the simulated time at which the data is written.

- A `constant` directory that contains files specifying physical properties—e.g., turbulence properties—for the simulated problem and a sub-directory `polyMesh` that provides full information of the computing mesh.

- A `system` directory that contains at least three files: the `controlDict` file that provides general control parameters—e.g.,start/end time—for the simulation; the `fvSchemes` file that specifies numerical schemes for discretization; and the `fvSolution` file that specifies tolerances and controls the algorithm for linear equation solvers.

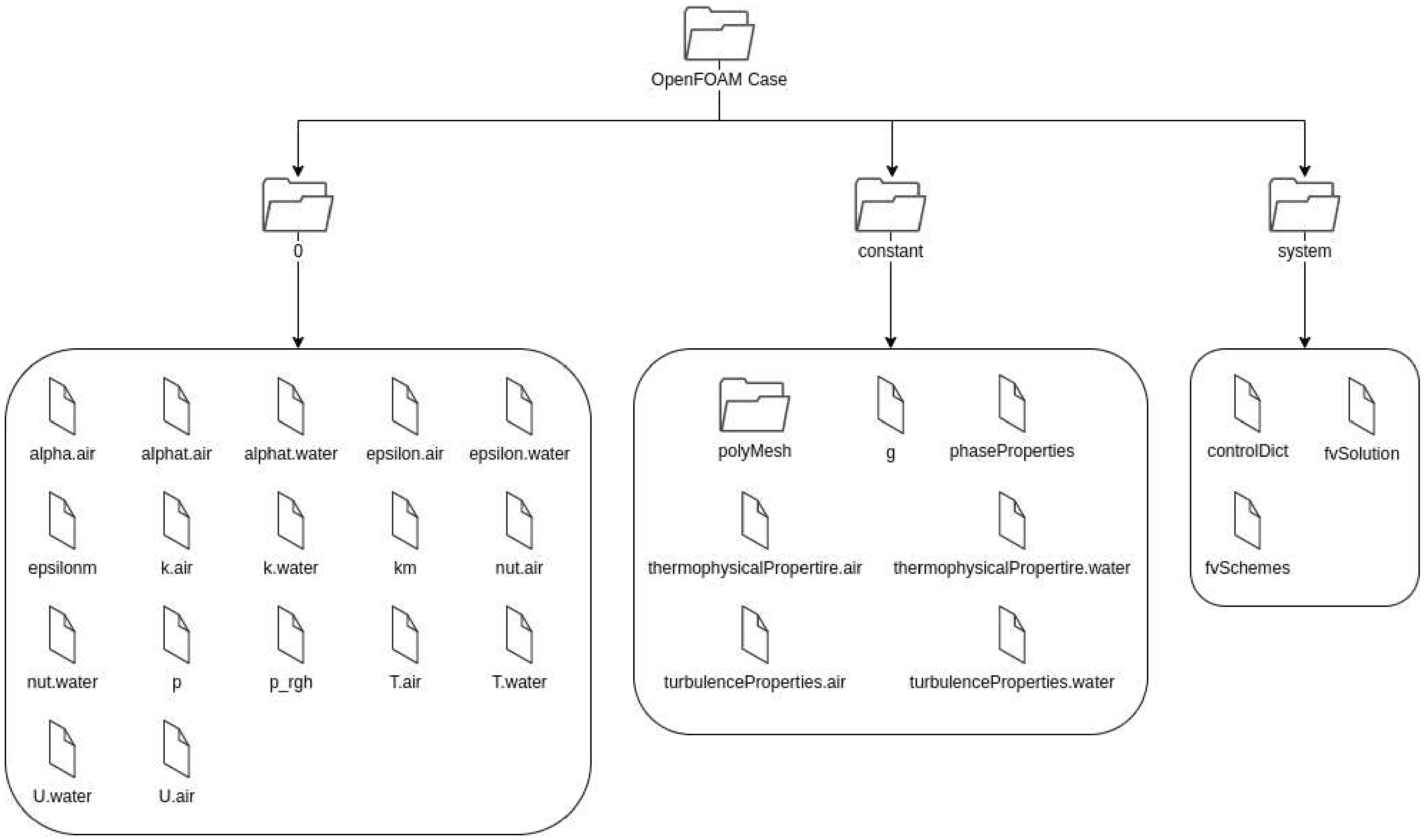


Figure 3.1: Directory structure of a typical `reactingTwoPhaseEulerFoam` case.

3.1.2 Simulation geometry and mesh

A simplified sieve-tray-like geometry was used for solver testing. The design parameters of the simulated tray are shown in Table 3.1. Figure 3.2 shows the schematic diagram of the geometry used in the simulation. The liquid phase enters through the liquid inlet boundary at a liquid load per weir length of $8.25 \times 10^{-4}\,\mathrm{m^2\,s^{-1}}$. The gas phase enters through the gas inlet boundary at a superficial gas velocity of $0.5\,\mathrm{m\,s^{-1}}$.

A fully orthogonal hexahedral grid was created based on this geometry. The total number of grid cells within the computing domain is 7740, with a uniform grid size of $5 \times 10^{-3}\,\mathrm{m}$.

Table 3.1: Design parameters of the simple sieve tray.

Design parameter	Value
Tray length	0.215 m
Tray height	0.18 m
Tray depth	0.025 m
Downcomer clearance	0.015 m
Weir height	0.08 m
Number of holes	10
Square Hole size	0.005 m

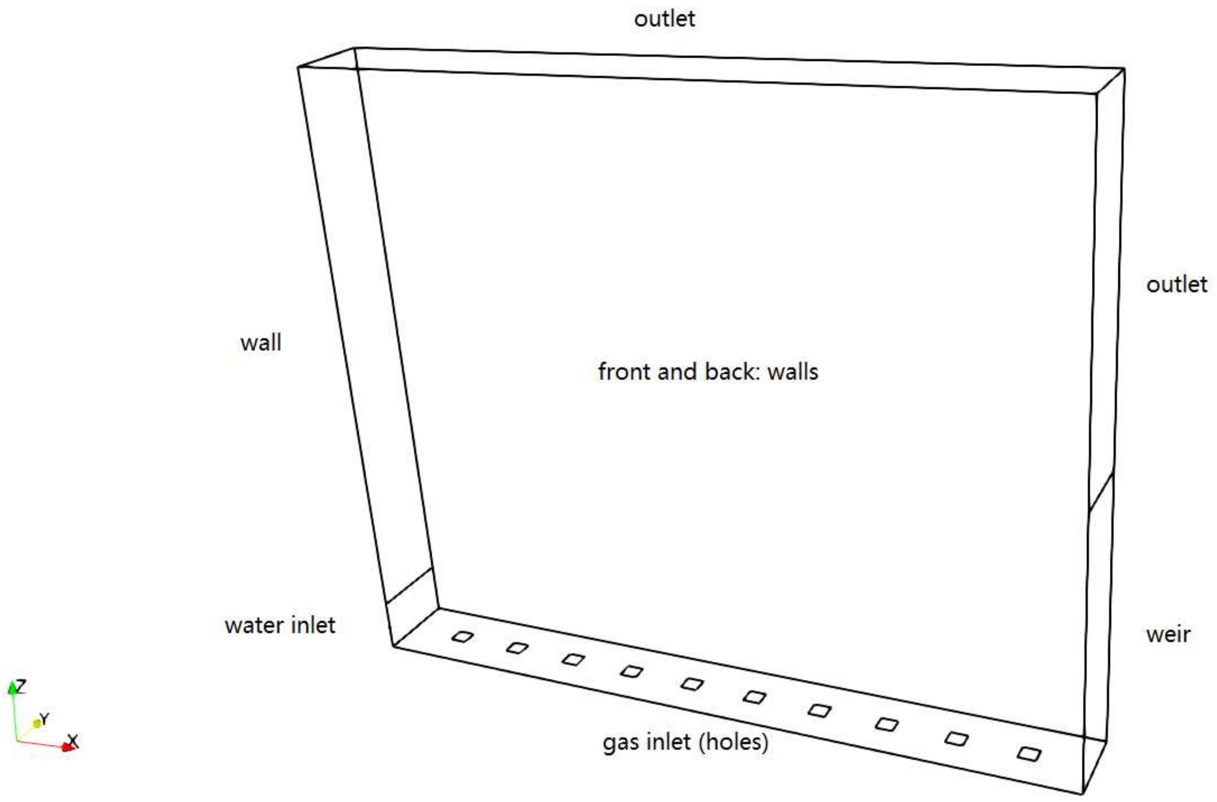


Figure 3.2: Schematic layout of the simple sieve tray geometry.

3.1.3 Boundary and initial conditions

In the `0` directory, a set of files is provided for every flow variable that is involved in the simulation. These flow variables are described in Table 3.2.

Table 3.2: Variables that require input of boundary conditions and initial conditions in the `0` directory of the base case.

Flow variables	Physical meaning
`alpha.air`	Volume fraction of the gas phase
`alphat.air`	Turbulent thermal diffusivity of the gas phase
`alphat.water`	Turbulent thermal diffusivity of the liquid phase
`T.air`	Temperature of the gas phase
`T.water`	Temperature of the liquid phase
`U.air`	Velocity of the gas phase
`U.water`	Velocity of the liquid phase
`p`	Static pressure
`p_rgh`	Pseudo hydrostatic pressure
`k.air`	Turbulent kinetic energy of the gas phase
`k.water`	Turbulent kinetic energy of the liquid phase
`km`	Turbulent kinetic energy of the mixture
`epsilion.air`	Turbulent kinetic energy dissipation rate of the gas phase
`epsilion.water`	Turbulent kinetic energy dissipation rate of the liquid phase
`epsilionm`	Turbulent kinetic energy dissipation rate of the mixture
`nut.air`	Turbulent viscosity of the gas phase
`nut.water`	Turbulent viscosity of the liquid phase

Appropriate boundary conditions are essential to a successful simulation. In practice, the local flow profiles at the liquid and gas inlets are usually unavailable; instead, the averaged quantities like flow rates are given. To set up a simulation with the lack of information, some reasonable assumptions and simplifications are required when specifying the boundary conditions.

OpenFOAM® offers a bank of comprehensive boundary conditions, which inherit from two of the most basic built-in boundary conditions, namely `FixedValue` and `zeroGradient`.

The `fixedValue` boundary condition prescribes a Dirichlet condition, which explicitly specifies the value on the patch as follows:

$$\phi = \phi_{ref} \tag{3.1}$$

Where ϕ is an arbitrary quantity and ϕ_{ref} is the explicitly specified reference value on the patch.

The `zeroGradient` boundary condition prescribes a Neumann condition, which specifies a zero gradient of the field variable in the direction normal to the patch:

$$\frac{\partial \phi}{\partial \mathbf{n}} = 0 \tag{3.2}$$

Where $\mathbf{n}$ is the unit vector normal to the patch face.

In this study, heat transfer was not considered. Thus, the boundary conditions for turbulent thermal diffusivity of both phases—i.e., `alphat.air` and `alphat.water`—were specified using `calculated` conditions, which means the values of these variables will be calculated using relevant flow variables—e.g.,$\alpha_t = \nu_t/\mathbf{Pr}_t$, α_t is turbulent thermal diffusivity, ν_t is kinematic Eddy viscosity, and $\mathbf{Pr}_t$ is Turbulence Prandtl Number—with values set to be 0 on all boundary patches except the walls. On the boundary patch for walls, wall functions were employed. The initial values of these two flow variables in the internal computing region were specified to be 0.

The boundary conditions for temperatures—i.e., `T.air` and `T.water`—were specified using `fixedValue` conditions with values of 273.15 K on all boundary patches. It should be noted that since heat transfer is not considered in this study, the specified values of the temperatures do not affect the simulation results. Initial values of these two flow variables were specified to be 273.15 K for the internal fields as well.

For turbulence variables `k.air`, `k.water`, `km`, `epsilion.air`, `epsilion.water`, and `epsilionm`, the boundary conditions on the inlet patches were specified using `fixedValue` conditions with values estimated from empirical correlations [1]. Wall functions were applied to the wall boundary patches. On the outlet boundary, the boundary conditions for these variable were specified as `inletOutlet` conditions. This boundary condition works as a `zeroGradient` condition for outflow and for backflow it works as a `fixedValue` condition with the value to be explicitly specified by the user. In this case, the values of these turbulence variables in the backflow condition were specified to be the same as the values of initial internal fields. Initial values of these flow variables within the internal region were set to be small values with the same magnitude of the estimated values on the inlet patches.

For the turbulence flow variables `nut.air` and `nut.water`, wall functions were applied to the boundary patch for walls. On the other boundary patches, the boundary conditions were specified as `calculated`. The initial values of these two flow variables were set to be a small value (1×10^{-8}) for the internal fields.

In the solver `reactingTwoPhaseEulerFoam` the pseudo hydrostatic pressure p_{rgh} is solved instead of the regular static pressure p to gain numerical advantages [13]. p_{rgh} is related to p through the following formula:

$$p_{rgh} = p - \rho \mathbf{g}(h - h_{ref}) \tag{3.3}$$

where `g` is the gravitational acceleration, h is the height in the opposite direction to gravity, h_{ref} is the reference height in the opposite direction to gravity. Thus, only explicit boundary conditions are required for `p_rhg` and all boundary conditions in the `p` file can be set to the `calculated` condition. The specifications of boundary and initial conditions for `p_rgh` and the rest three flow variables—i.e., `alpha.air`, `U.air`, and `U.water`—are given in the following paragraphs.

On the boundary patch of walls, the specified boundary conditions for each flow variable are listed below:

- `alpha.air`: `zeroGradient` boundary condition.
- `p_rgh`: `fixedFluxPressure` boundary condition. This boundary condition adjusts the pressure gradient such that the flux on the boundary is that specified by the velocity boundary condition.
- `U.air`: A free-slip wall boundary condition.
- `U.water`: A no-slip wall boundary condition.

On the boundary patch of liquid inlet, the following boundary conditions for each flow variable were specified:

- `alpha.air`: `fixedValue` boundary condition with the value specified to be 0. This assumes that only pure liquid phase is entering the computing domain through the liquid inlet.
- `p _rgh`: `fixedFluxPressure` boundary condition.

- `U.air`: A free-slip wall boundary condition. This assumes that the gas phase does not penetrate the liquid inlet boundary.

- `U.water`: `fixedValue` boundary condition with the value calculated from the known liquid flow rate. The formula is as follows:

$$\mathbf{U}_{L,in} = \begin{bmatrix} \frac{Q_L}{h_{ap} L_w} \\ 0 \\ 0 \end{bmatrix} \tag{3.4}$$

  where Q_L is the liquid load, h_{ap} is the downcomer clearance, and L_w is the weir length.

On the boundary patch of gas inlet, the following boundary conditions for each flow variable were specified:

- `alpha.air`: `fixedValue` boundary condition with the value set to be 1. This assumes that only pure gas phase is entering the computing domain through the gas inlet.

- `p _rgh`: `fixedFluxPressure` boundary condition.

- `U.air`: `fixedValue` boundary condition with the value calculated from the known gas flow rate. The formula is as follows:

$$\mathbf{U}_{G,in} = \begin{bmatrix} 0 \\ 0 \\ |\mathbf{U}_s| \frac{A_B}{A_H} \end{bmatrix} \tag{3.5}$$

  where A_B is the bubbling area, and A_H is the total area for holes.

- `U.water`: A no-slip wall boundary condition. This assumes that the liquid phase does not penetrate the gas inlet boundary.

On the boundary patch of outlet, the following boundary conditions each flow variable were specified:

- `alpha.air`: `inletOutlet` condition. In this case, the value for backflow was specified to be 1, which implies that backflow is pure gas.

- `p_rgh`: `prghTotalPressure` condition.
- `U.air`: `pressureInletOutletParSlipVelocity` condition.
- `U.water`: `pressureInletOutletParSlipVelocity` condition.

Details of the boundary conditions applied to this boundary patch are discussed in Section 3.3.

The initial value of modified pressure, `p_rgh`, was set to 101 325 kg m^{-1} s^{-2} for the internal field. It should be noted that for incompressible fluids simulation, as was done in this study, the numerical value used for the pressure field initialization has no influence on the simulation results. Both the initial values of gas velocity `U.gas` and liquid phase velocity `U.water` were specified to be 0 for internal fields. The gas phase fraction `alpha.air` was initialized to a value of 0.2 from the bottom of the tray to the weir height and initialized to a value of 0 for the rest part of the internal field.

3.1.4 Physical properties and other sub-models

Within the `constant` dierctory, there is a `thermoType` dictionary which provides a set of thermodynamic and transport property models. As an example, the entry for the packages of thermophysical models employed in this study for water is presented in Figure 3.3:

The entries in this `thermoType` dictionary are explained below:

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

```

Figure 3.3: An example entry for `thermoType` dictionary.

- `type`: Thermophysical model type. `heRhoThermo` is a density-based model which is the only valid option for solver `reactingTwoPhaseEulerFoam`[13].

- `mixture`: The model specifies mixture composition, `pureMixture` represents a mixture that has only one chemical component (i.e., no mass transfer or reactions).

- `transport`: The transport model used to evaluates dynamic viscosity μ and thermal conductivity κ. The `const` model specifies a constant dynamic viscosity μ and a constant Prandtl number $\mathbf{Pr}$, value of the thermal conductivity κ is calculated using the following formula:

$$\kappa = \frac{c_p \mu}{\mathbf{Pr}} \tag{3.6}$$

- `thermo`: The thermodynamic model used to evaluate the specific heat capacity c_p . The `hConst` model specifies a constant c_p and a constant heat of fusion H_f.

- `equationOfState`: The `rhoConst` model assumes a constant density ρ.

- `specie`: Defines number of moles, `Moles`, and molecular weight, `molWeight`, of the specie.

- `energy`: The form of energy to be used in the simulation. The key word `sensibleInternalEnergy` indicates that the solution uses internal energy e in forms that the heat of formation Δh_s is not included.

The same set of thermophysical models were selected for air as well. The required numerical inputs for the selected models are summarized in Table 3.3.

Table 3.3: Specifications of thermophysical properties for the base case.

	Air	Water
Density	$1.204\,\mathrm{kg\,m^{-3}}$	$998.21\,\mathrm{kg\,m^{-3}}$
Molecular weight	$28.97\,\mathrm{g\,mol^{-1}}$	$18\,\mathrm{g\,mol^{-1}}$
Specific heat	$1006\,\mathrm{J\,kg^{-1}\,K^{-1}}$	$4184.4\,\mathrm{J\,kg^{-1}\,K^{-1}}$
Heat of fusion	$0\,\mathrm{J\,K^{-1}}$	$0\,\mathrm{J\,K^{-1}}$
Dynamic viscosity	$1.825\times10^{-5}\,\mathrm{kg\,m^{-1}\,s^{-1}}$	$0.39\,\mathrm{kg\,m^{-1}\,s^{-1}}$
Prandlt number	0.7309	6.99

In the `phaseProperties` dictionary within the `constant` directory, a system of two phases was defined for the simulation: water, and air. A constant diameter model was selected for both phases with a diameter of 0.0055 m was specified for air bubbles and water droplets. However, it should be noted that the specified values of the bubble/droplet diameters should have no significant impact on the simulation results because the drag model employed in the simulations is independent of the bubble/-droplet diameter. Drag force is the only interphase momentum transfer mechanism considered in the simulations and air is always considered as the dispersed phase. The drag force model employed in the simulations is described in equation (2.70). The equivalent code of equation (2.70) is described in Section 2.4.2.

In the `turbulenceProperties.air` and `turbulenceProperties.water` dictionaries within the `constant` directory, the `mixtureKEpsilon` turbulence model as described in Section 2.4.3 was specified for both phases.

Gravitational acceleration was included in this study and its value is specified in the `g` dictionary within the `constant` directory.

3.1.5 Discretization schemes

The discretization methods, which discretize the governing equations into algebraic equations over the control volumes, are specified in the `fvSchemes` dictionary within the `system` directory. The specifications of discretization schemes employed are listed in Table 3.4. In this study, transient simulations were performed to maintain numerical stability. However, analysis of the results focused on the time-averaged flow characteristics rather than their instantaneous behaviour. Thus, the implicit `Euler` temporal discretization scheme, which is first-order accurate but bounded, was employed. This scheme is less accurate than other second-order temporal discretization schemes—e.g., `backward`—but more robust. For the discretization of the gradient terms and the divergence of the stress tensor, the `Gauss linear` scheme was employed. This scheme is second-order accurate and unbounded. For the divergence terms involved in mass conservation equations, the `Gauss vanLeer` scheme, which is a second-order accurate bounded scheme in the class of total variation diminishing (TVD) schemes, was employed. This scheme is specially formulated to provide oscillation-free solutions in regions with rapidly changing gradients. The `Gauss` `limitedLinear 1` scheme was employed for turbulence and energy equations. This scheme has second-order accuracy and requires an explicitly specified coefficient with a value between 0 and 1, where 1 provides the strongest limiting towards the `Gauss upwind` scheme and 0 provides the least limiting towards the `Gauss linear` scheme. In this study, the coefficient was specified to be 1. The `Gauss limitedLinearV 1` scheme was applied to the divergence terms of momentum equations. It is similar to the `Gauss limitedLinear 1` scheme with an additional `V` in its name indicating that this scheme is applied separately for each component of the vector field. The `uncorrected` scheme was employed for surface normal gradient terms. This scheme is only valid for computing meshes with very low non-orthogonality, which is appropriate because the mesh prepared for the case studies is fully orthogonal. The scheme specified for Laplacian terms is `Gauss linear uncorrected`, which means that the `Gauss` scheme is employed for discretization, the `linear` scheme is employed for interpolation, and `uncorrected` scheme is specified for the surface normal gradient part of the Laplacian terms.

There are over fifty options of different schemes provided in OpenFOAM$^{®}$ [13]. Configuring the optimized combination of schemes which gives the best numerical performance for a given simulation is out of the scope of this study and many combinations of schemes may perform similarly. The presented combination of schemes was obtained through a process of trial and error following a few general guidelines:

- Use schemes of at least second-order accuracy for the flow fields of interest.
- Use schemes that ensure the boundedness of certain flow properties—e.g., phase fraction.
- Use surface normal gradient schemes that are appropriate based on the orthogonality of the computing grid.

Table 3.4: Specifications of discretization schemes for the base case.

Category	Syntax of related terms in the equations	Discretization scheme
Time schemes	`ddt()`	`Euler`
Gradient schemes	`grad()`	`Gauss linear`

Divergence schemes	`div(phi,alpha.air)`
	`div(phir,alpha.air)`
		`Gauss vanLeer`

Divergence schemes*	`div(alphaRhoPhi,U)`
	`div(phi,U)`
		`Gauss limitedLinearV 1`

Divergence schemes*	`div(alphaRhoPhi,(h|e))`
	`div(alphaRhoPhi,K)`
	`div(alphaPhi,p)`
	`div(alphaRhoPhi,(k|epsilon))`
	`div(phim,(k|epsilon)m)`
		`Gauss limitedLinear 1`

Divergence schemes*	`div((alpha*rho*nuEff)*dev2(T(grad(U))))`	`Gauss linear`
Laplacian schemes	`laplacian()`	`Gauss linear uncorrected`
Interpolation schemes	N/A	`linear`
Surface normal gradient schemes	`snGrad()`	`uncorrected`

* These schemes apply to equation terms for both phases—e.g. `div(phi.air,U.air)` and `div(phi.water,U.water)`—the suffixes are omitted for simplicity.

3.1.6 Solution and algorithm control

The algebraic equations created as a result of the discertization process must be solved using linear equation solvers that are explicitly specified within the `fvSolutions` dictionary in the `system` directory. An example of the syntax is presented in Figure 3.4.

The descriptions of relevant parameters are as follows:

- `p_rgh`: The name of the flow variable being solved. In this example, the linear solver is specified for the pseudo hydrostatic pressure calculation.
- `solver`: Specification of the linear solver. In this example, `GAMG` stands for the generalised geometric-algebraic multi-grid linear solver.
- `smoother`: Specification of the smoother being applied to the linear solver. In this example, the `GaussSeidel` smoother is specified.
- `tolerance`: A value that serves as the absolute convergence criteria. If the solver error evaluated by substituting the current solution back into the original algebraic equations being solved and taking the normalized magnitude of the difference between left and right hand sides falls below this specified value, the solver will stop iterating.
- `relTol`: A value that serves as the relative convergence criteria. If the relative error evaluated by the normalized magnitude of the ratio of current solver error and the solver error from last iteration falls below this specified value, the solver will stop iterating. In OpenFOAM®, the linear solver will stop iterating when any of the specified tolerances have been reached. In transient simulations, this value is commonly set to be 0 to maintain the accuracy of solutions in each time step.

We can apply different linear solver settings for each flow variable, and the specifications used for the base case are summarized in Table 3.5.

For the base case, the `PIMPLE` algorithm, as described in Section 2.4.1, was employed for velocity-pressure coupling. Control parameters of the algorithm are specified in the `fvSolutions` dictionary and the syntax is presented in Figure 3.5.

The descriptions of relevant parameters are as follows:

```
p_rgh
{
    solver          GAMG;
    smoother        GaussSeidel;
    tolerance       1e-8;
    relTol          0;
}

```

Figure 3.4: An example syntax of linear solver specification entry.

Table 3.5: Specifications of linear solvers for the base case.

Flow field	Linear solver	Residual control
Phase fraction	solver: `smoothSolver` smoother: `symGaussSeidel`	1e-8
Pressure	solver: `GAMG` smoother: `DIC`	1e-7
Velocity	solver: `smoothSolver` smoother: `symGaussSeidel`	1e-8
Energy	solver: `smoothSolver` smoother: `symGaussSeidel`	1e-8
Turbulence	solver: `smoothSolver` smoother: `symGaussSeidel`	1e-8

```
PIMPLE
{
    nOuterCorrectors 3;
    nCorrectors      2;
    nNonOrthogonalCorrectors 0;
}

```

Figure 3.5: The syntax of `PIMPLE` algorithm specifications

- `nOuterCorrectors`: The number of times the entire system of equations must be solved for each time step, namely the outer loop.
- `nCorrectors`: The number of times the pressure equation and momentum corrector must be solved within each outer loop for each time step, namely the inner loop.
- `nNonOrthogonalCorrectors`: The iterations of the explicit non-orthogonal correction applied to pressure equation within each inner loop.

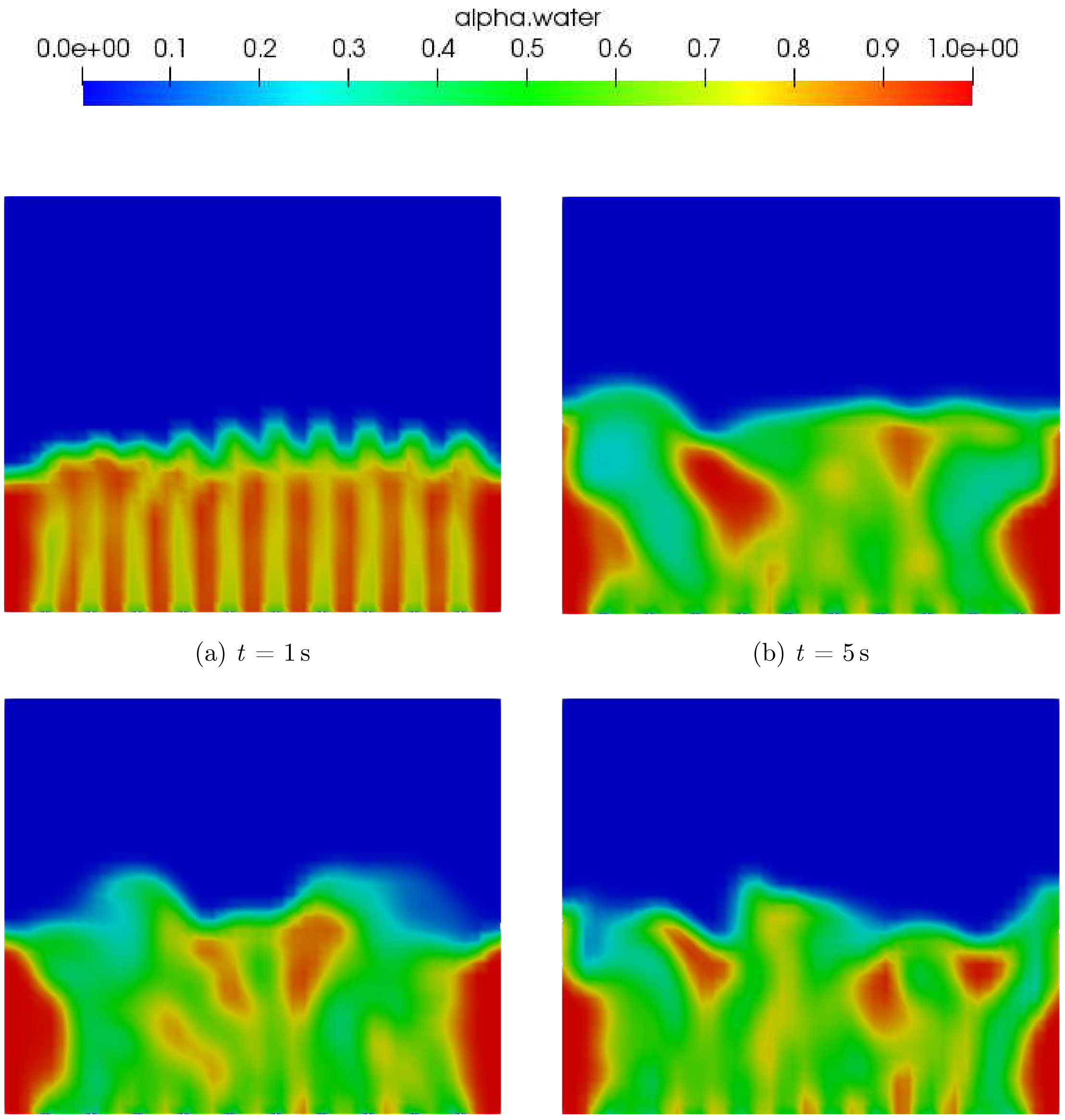
The under-relaxation factor, a numerical treatment that improves the stability of a simulation[36], can be specified in the `fvSolutions` dictionary as well. For the base case the under-relaxation factor was specified to be 1, meaning no under-relaxation is applied.

The time step size of simulation is specified in the `controlDict` dictionary in the `cystem` directory. For the case study `adjustTimeStep` was used, which allows the solver to adjust the time step size during the simulation based on an explicitly specified maximum Courant number `maxCo`. For the base case, the value of `maxCo` was set to be 0.5.

The reader is referred to the OpenFOAM® documentation [13] for detailed descriptions of the mentioned numerical settings as well as other parameters that are not presented in this section.

3.1.7 Base case results

A transient simulation was conducted for the previously described base case. The liquid phase fraction profiles at various time steps are shown in Figure 3.6. It is shown that, at the beginning of the simulation, the air phase injects into the domain through the holes of the sieve tray as gas jets. This is because the initialized phases on the tray are stationary, which leads to a large relative velocity between the phases on the tray and the gas entering the tray. As the simulation continues, the two phases become well mixed in the tray center and high liquid hold-up can be observed at the lower corners of the tray due to the constraints of walls. Time-averaged liquid phase fraction profiles are presented in Figure 3.7. It shows that the time-averaged liquid phase fraction at the given plane does not change appreciably after a certain time step, indicating the simulation has reached a quasi-steady-state condition. The overall liquid hold-up in the system was monitored throughout the simulation and the results are shown in Figure 3.8. It shows that the instantaneous value of overall liquid hold-up is time dependent but the time-averaged overall liquid hold-up dose not change appreciably after time $t = 25\,\mathrm{s}$. Therefor, the base case simulation is deemed to have converged to a quasi-steady-state condition after 20 seconds of simulated time.



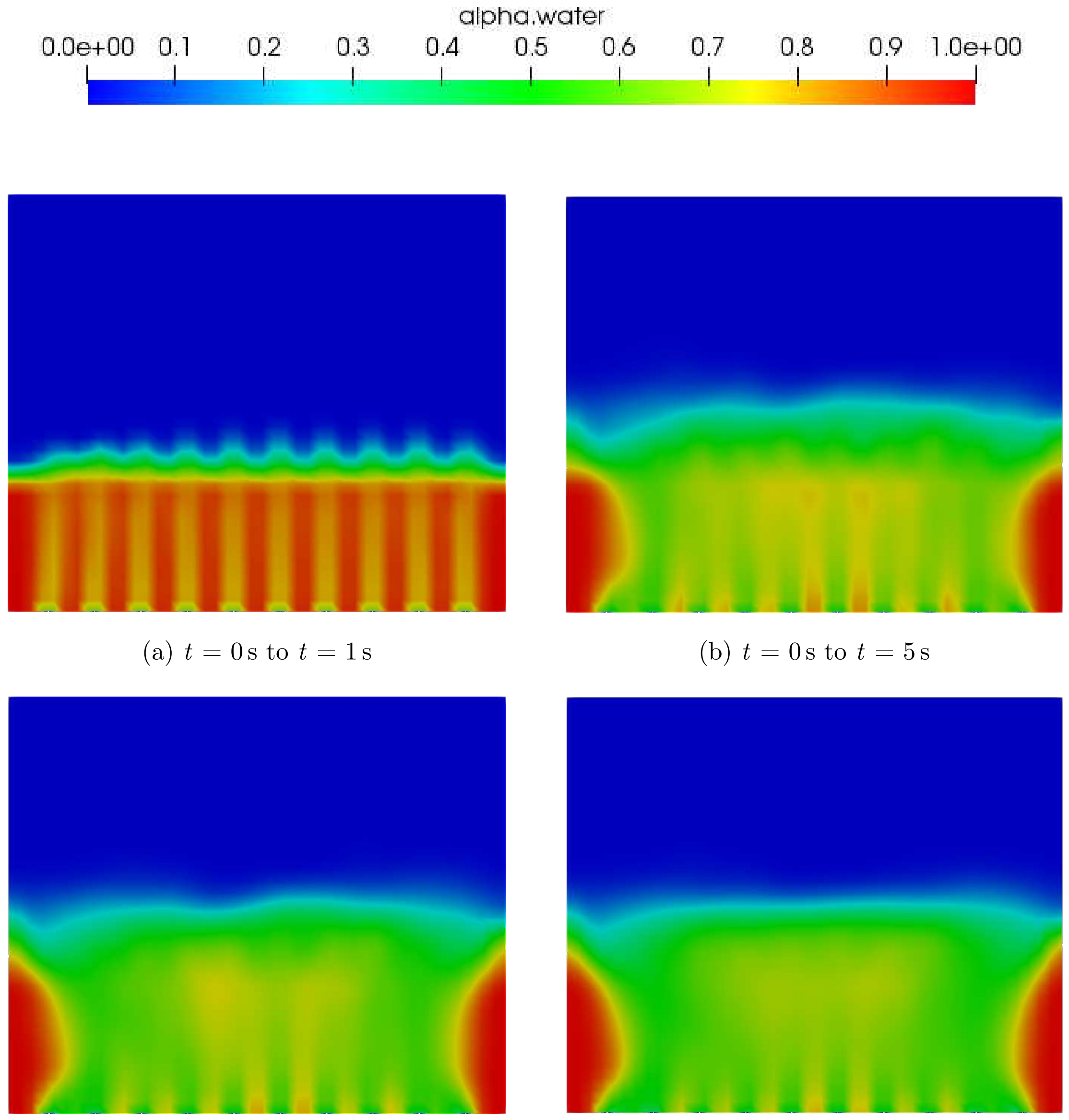
(a) $t = 1\,\mathrm{s}$

(b) $t = 5\,\mathrm{s}$

(c) $t = 10\,\mathrm{s}$

(d) $t = 20\,\mathrm{s}$

Figure 3.6: Instantaneous Liquid phase fraction profiles on the XZ plane across the tray center at different simulated times for the base case.



(a) $t = 0\,\text{s}$ to $t = 1\,\text{s}$

(b) $t = 0\,\text{s}$ to $t = 5\,\text{s}$

(c) $t = 0\,\text{s}$ to $t = 10\,\text{s}$

(d) $t = 0\,\text{s}$ to $t = 20\,\text{s}$

Figure 3.7: Time-averaged liquid phase fraction profiles on the XZ plane across the tray center over different time intervals for the base case.

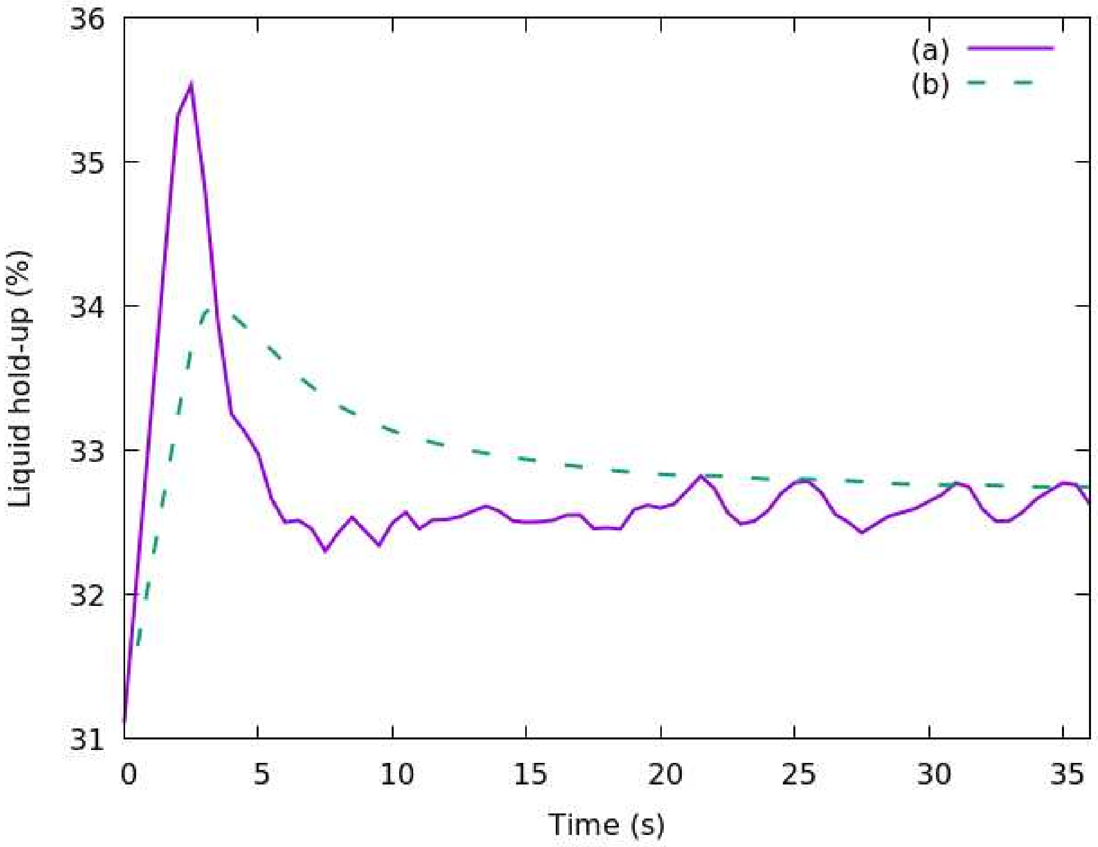
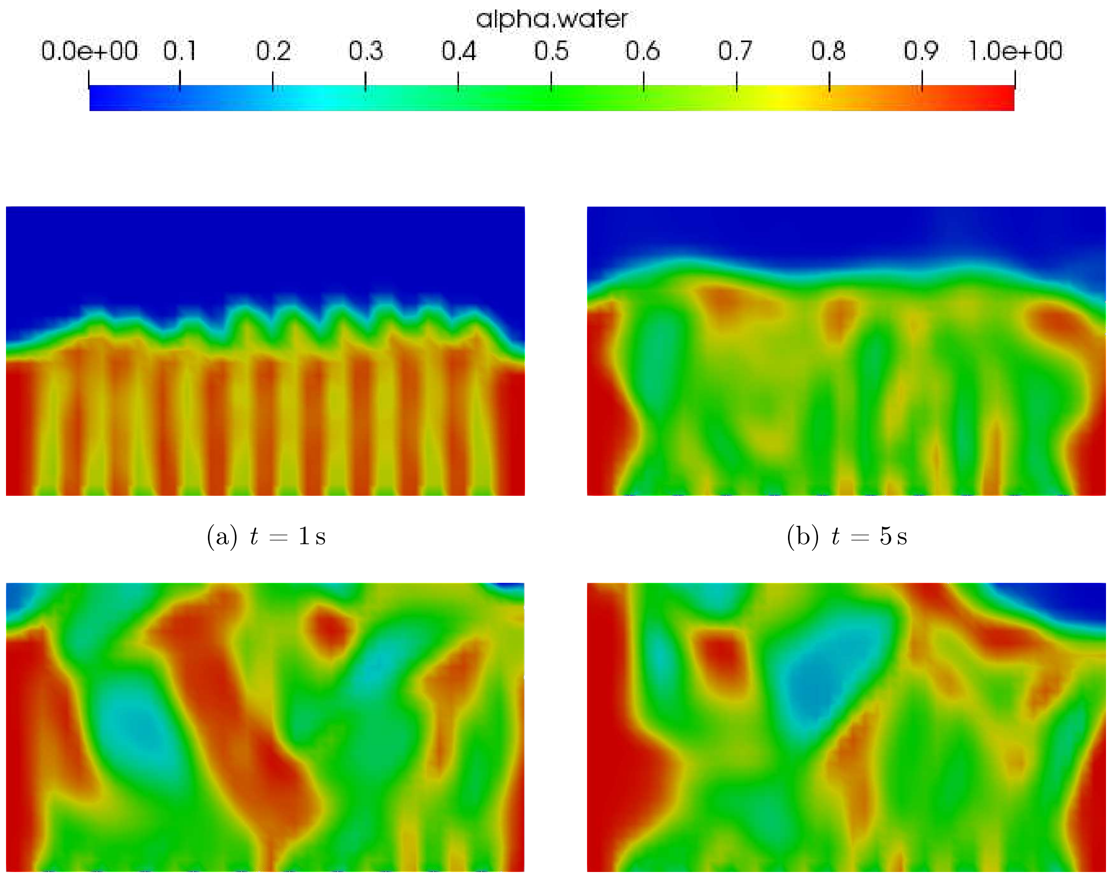


Figure 3.8: Overall liquid hold-up as a function of time for the base case. (a) Instantaneous values at time t, (b) Time-averaged values over the time range from $t = 0\,\mathrm{s}$ to time t.

3.2 Flow regime case study

As seen in Figure 3.6, the top one third of the tray region is practically occupied by gas phase only. A test case with a reduced tray height was examined to explore the possibility of reducing computing cost by reducing the size of the geometry because the region of interest is mainly the mixed froth regime above the plate.

The geometry of this test case is modified from the based case with a reduced tray height of 0.12 m. Other settings were kept the same as the base case. The instantaneous phase fraction results are shown in Figure 3.9. The simulation results show that liquid phase will reach the top boundary of the tray with reduced height of the simulation domain, which is an unfavoured situation for the adopted boundary condition settings. The `prghTotalPressure` boundary condition was used for pressure field on the top boundary, which allows possible backflow into the simulation regime with a user-defined phase composition. For a sieve tray with enough height, it is safe to assume the backflow through the top boundary will be pure air as water droplets above the froth regime will eventually drop back due to gravity. However, with reduced height of the simulation domain causing both phases to be able to leave the region through the top boundary in a heterogeneous manner, the phase composition of the possible backflow becomes unpredictable. Moreover, the reduced height brings difficulties when it comes to determine the dispersion height. The results shown in Figure 3.9 suggest that the simulated sieve tray geometry should have enough vertical space such that the single-phase backflow boundary condition is suitable and to calculation of the dispersion height from the simulation.



(a) $t = 1\,\mathrm{s}$

(b) $t = 5\,\mathrm{s}$

(c) $t = 10\,\mathrm{s}$

(d) $t = 20\,\mathrm{s}$

Figure 3.9: Instantaneous liquid phase fraction profiles on the XZ plane across the tray center at different simulated times for the flow regime test case.

3.3 Boundary condition case study

OpenFOAM®requires a set of boundary conditions to be specified on each boundary patch for every flow variables, which includes temperature, turbulence parameters, phase fraction, velocity, and pressure. On the boundary patch for the outlet, the flow is complex and there is no predetermined information available about the phase fraction and velocity profiles for each phase. Therefore, the following boundary conditions were tested for each flow variable:

- `p _rgh`:

  1. The `prghPressure` boundary condition provides an explicitly specified value for the static pressure p similar to the `fixedValue` condition.
  2. The `prghTotalPressure` boundary condition provides an explicitly specified reference pressure p_{ref}, and the value for p_{rgh} is then calculated as follows:

$$p_{rgh} = p - \rho \mathbf{g}(h - h_{ref})$$
$$p = \begin{cases} p_{ref} & \text{for outflow} \\ p_{ref} - 0.5\rho|\mathbf{U}|^2 & \text{for inflow} \end{cases} \quad (3.7)$$

  where p is static pressure, h is the height in the opposite direction to gravity, h_{ref} is the reference height in the opposite direction to gravity.

- `U.air`:

  1. `zeroGradient` boundary condition.
  2. `pressureInletOutletVelocity` boundary condition. This boundary condition performs as a `zeroGradient` condition for outflow. For backflow, the velocity in the normal direction is calculated from the flux and the tangential velocity can be optionally specified. In this case, the tangential velocity was specified as zero.
  3. `pressureInletOutletParSlipVelocity` boundary condition that performs as a `zeroGradient` condition for outflow. For backflow, the velocity is calculated from the flux and a slip condition is applied tangential to the patch.

- `U.water`: The boundary condition for this flow variable was kept the same as the one used for `U.air`.

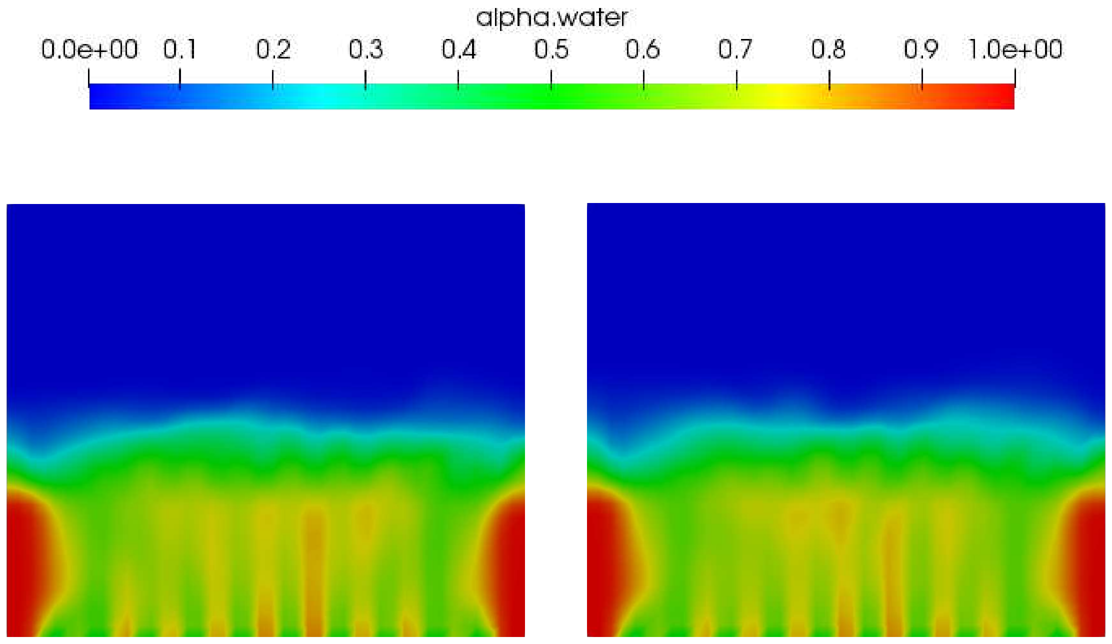
The results of test cases employing different pressure and velocity boundary condition combinations on the outlet boundary patch are presented in Table 3.6.

Table 3.6: Results of boundary condition case studies.

Case	Pressure boundary condition at outlet	Velocity boundary condition at outlet	Stability
01	`prghPressure`	`pressureInletOutletVelocity`	unstable
02	`prghPressure`	`pressureInletOutletParSlipVelocity`	unstable
03	`prghPressure`	`zeroGradient`	unstable
04	`prghTotalPressure`	`pressureInletOutletVelocity`	stable
05	`prghTotalPressure`	`pressureInletOutletParSlipVelocity`	stable
06	`prghTotalPressure`	`zeroGradient`	unstable

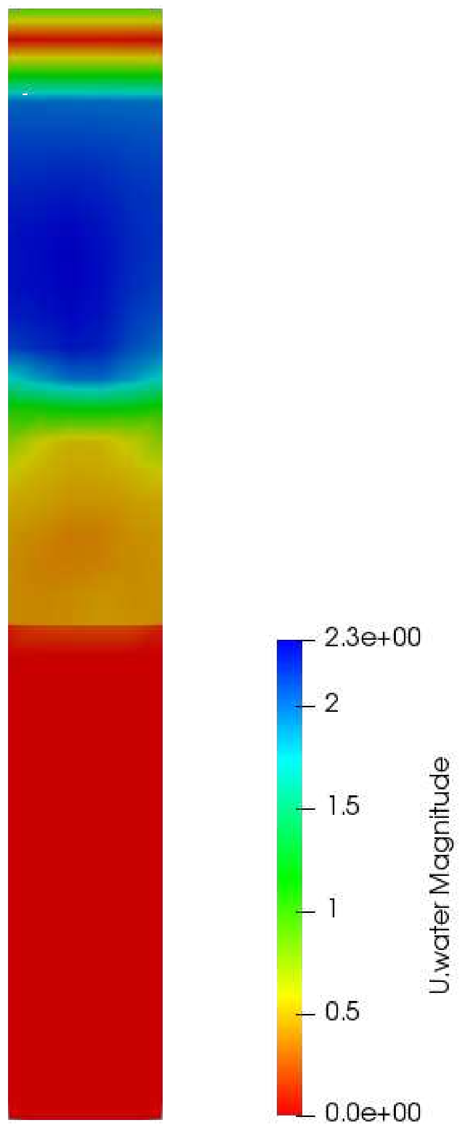
The results indicate that a pressure boundary condition which allows backflow is required at the outlet patch. The `prghPressure` boundary condition leads to unstable simulations because it can not uphold continuity while handling backflow. For velocity at the outlet patch, `zeroGradient` is an unsuitable option because the backflow flux at the outlet is not constrained. Both `pressureInletOutletVelocity` and `pressureInletOutletParSlipVelocity` deliver stable simulation and give virtually identical prediction of phase fraction profiles as illustrated in Figure 3.10. However, these two velocity boundary conditions lead to different predicted velocity fields near the outlet patch due to the methods they utilize to calculate the backflow velocity. As illustrated in Figure 3.11, `pressureInletOutletVelocity` leads to a nonsensical velocity profile around the top corner of the outlet patch and the `pressureInletOutletParSlipVelocity` gives a more realistic prediction.

Unless otherwise indicated, the boundary conditions `prghTotalPressure` and `PressureInletOutletParSlipVlocity` are specified for the pressure and velocity fields at the outlet patch for all subsequent simulations conducted in this study.

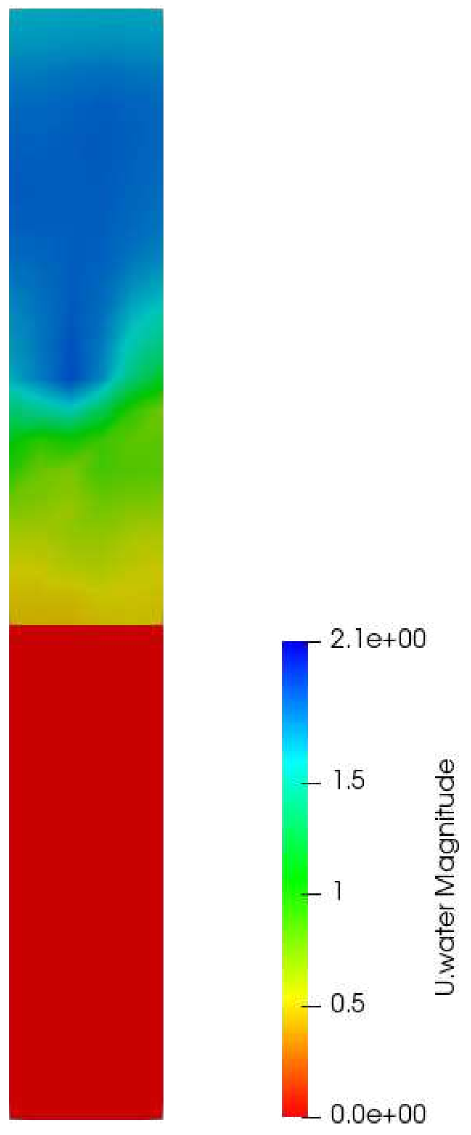


(a) (b)

Figure 3.10: Time-averaged liquid phase fraction profiles on the XZ plane across the tray center. (a) Case 04 in Table 3.6, (b) Case 05 in Table 3.6. Over time interval from $t = 0\,\mathrm{s}$ to $t = 5\,\mathrm{s}$.



(a)



(b)

Figure 3.11: Instantaneous liquid phase velocity profile at $t = 5\,\mathrm{s}$ on the YZ plane across the outlet patch. (a) case 04 in Table 3.6, (b) case 05 in Table 3.6.

3.4 Drag model case study

Recall the model for drag force calculation developed by Krishna et al. [23] in its original form:

$$C_d = \frac{4}{3}\frac{\rho_c - \rho_d}{\rho_c}\mathbf{g}d\left(\frac{\alpha_d^B}{\mathbf{U}_s}\right)^2 \tag{2.21 revisited}$$

As stated in Section 2.4.2, this model was implemented into OpenFOAM® with the following modification:

$$C_d = \frac{4}{3}\frac{\rho_c - \rho_d}{\rho_c}\mathbf{g}d\left(\frac{\alpha_d^B}{\mathbf{U}_s}\right)^2 \frac{max(\alpha_c, \alpha_{c,residual})}{1-\alpha_d^B} \tag{2.70 revisited}$$

The same modification was suggested by Krishna et al. [22] for the reason that it helps the simulation to overcome numerical instability. A simulation was conducted using the base case with the drag model switched from the modified version to its original form. The results are shown in Figure 3.12. It is shown that the simulation presents no numerical instability using the unmodified drag model, but the predicted continuous phase hold-up in the computing domain decreases over time, which is nonsensical at the given operating condition.

It is obvious that the drag force calculated using equation (2.21) is proportional to the dispersed phase fraction α_d and inversely proportional to the continuous phase fraction α_c when other flow properties are set to be constants. This leads to over-predicted drag force in regions where the local continuous phase fraction is relatively low. The over-predicted drag force promotes momentum transfer from the continuous phase to the dispersed phase. As a result, the continuous phase will carried away by the dispersed phase in such regions. This positive feedback loop will eventually remove most of the continuous phase from the computing domain. The root of this problem is miscalculated momentum exchange when phase inversion occurs. When phase inversion occurs in certain regions, the original dispersed phase becomes the continuous phase and thus the drag force is acting on the previous continuous phase now. This phase inversion effect has not been considered when using equation (2.70).

In OpenFOAM®, the phase inversion problem can be handled with built-in blending methods, which calculates the drag coefficient K from three parts:

$$K = f_1 K_{1,2} + f_2 K_{2,1} + (1 - f_1 - f_2)\, K_{1\&2} \tag{3.8}$$

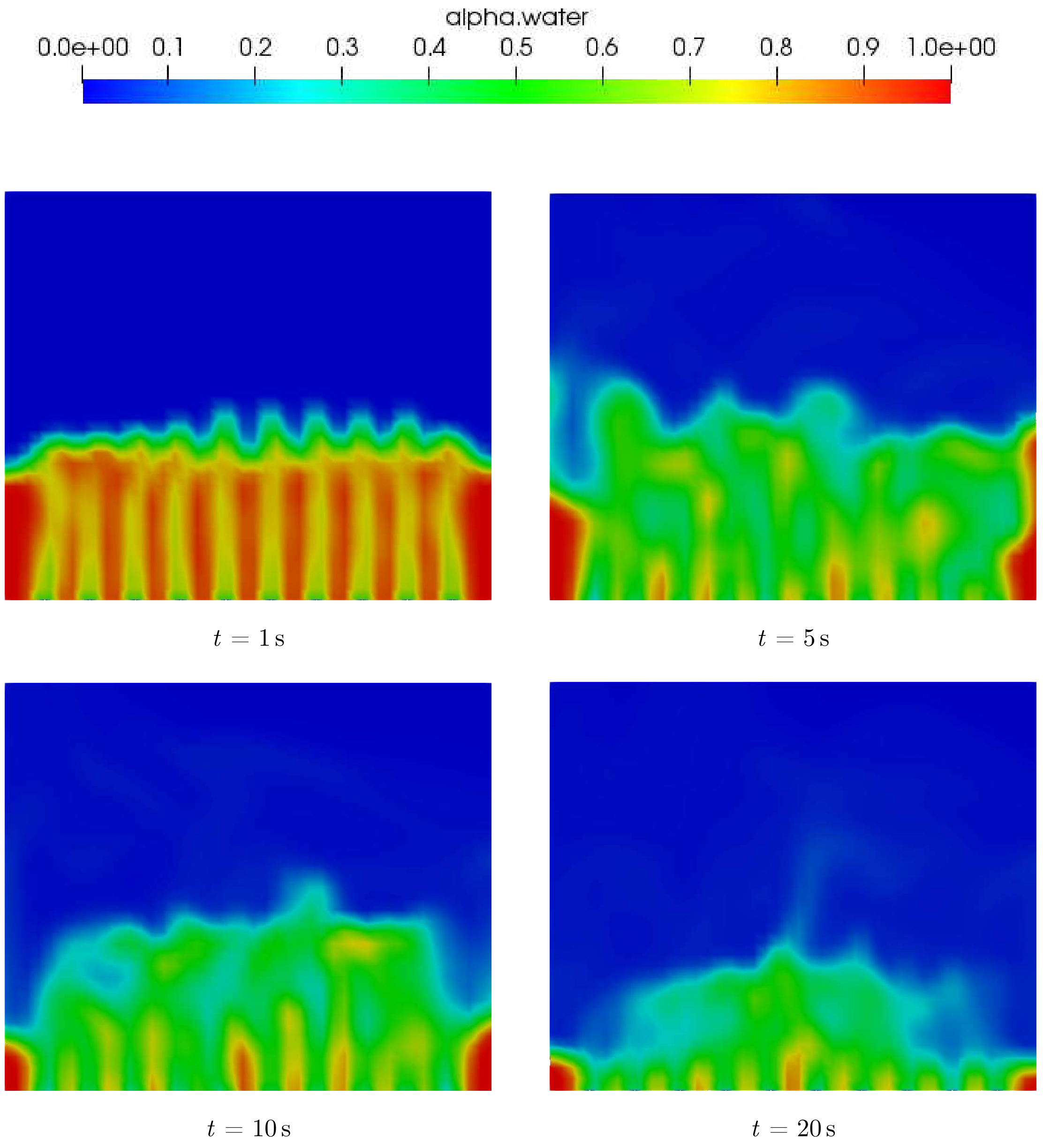


Figure 3.12: Instantaneous liquid phase fraction profiles on the XZ plane across the tray center at different simulated times using drag model proposed by Krishna et al. [23] in its original form without blending.

where f_1 and f_2 are blending factors, $K_{1,2}$ and $K_{2,1}$ are drag coefficients evaluated on the basis of phase 1 and phase 2 as the dispersed phases respectively, and $K_{1\&2}$ is the drag coefficient evaluated for situation in which both phases are considered as partly continuous or partly dispersed. In this study, the linear blending method was tested, which defines the blending coefficients as follows:

$$\begin{aligned} f_1 &= \min\left(\max\left(\frac{\alpha_2-\alpha_{2,P}}{\alpha_{2,F}-\alpha_{2,P}},0\right),1\right) \\ f_2 &= \min\left(\max\left(\frac{\alpha_1-\alpha_{1,P}}{\alpha_{1,F}-\alpha_{1,P}},0\right),1\right) \end{aligned} \tag{3.9}$$

where $\alpha_{1,F}$ and $\alpha_{2,F}$ are the minimum phase fraction values for phase 1 and phase 2 to be considered as fully continuous phases, respectively; and $\alpha_{1,P}$ and $\alpha_{2,P}$ are the minimum phase fraction values for phase 1 and phase 2 to be considered as partly continuous phases, respectively. If the phase fraction value for phase 1 is less than the value of $\alpha_{1,P}$ then phase 1 will be considered as a fully dispersed phase. The same rule applies to phase 2 as well.

The linear blending method generates a continuous function for the blending factors for the whole range of the phase fraction values as shown in Figure 3.13.

To test this method for handling the phase inversion problem, test cases were constructed using equation (2.21) to evaluate the momentum transfer term due to drag force when gas phase is considered to be the dispersed phase. When liquid phase is considered to be the dispersed phase, the `SchillerNaumann` [43] model was employed for the momentum transfer evaluation. The following formula is implemented in OpenFOAM® for the `SchillerNaumann` model:

$$C_d = \begin{cases} 0.44, & \text{if } \mathbf{Re} > 1000 \\ 24\left(1.0+0.15\mathbf{Re}^{0.687}\right)/\mathbf{Re}, & \text{if } \mathbf{Re} \leq 1000 \end{cases} \tag{3.10}$$

where **Re** is obtained from equation (2.68). Contributions of these two drag models are handled via linear blending method. The parameters used for blending are summarized in Table 3.7. The same values were assigned for the pair of $\alpha_{1,F}$ and $\alpha_{1,P}$ and the pair of $\alpha_{2,F}$ and $\alpha_{2,P}$ to avoid the situation of having any phase being partly dispersed, as Figure 3.14 illustrates, hence the drag coefficient $K_{1\&2}$ does not need to be evaluated.

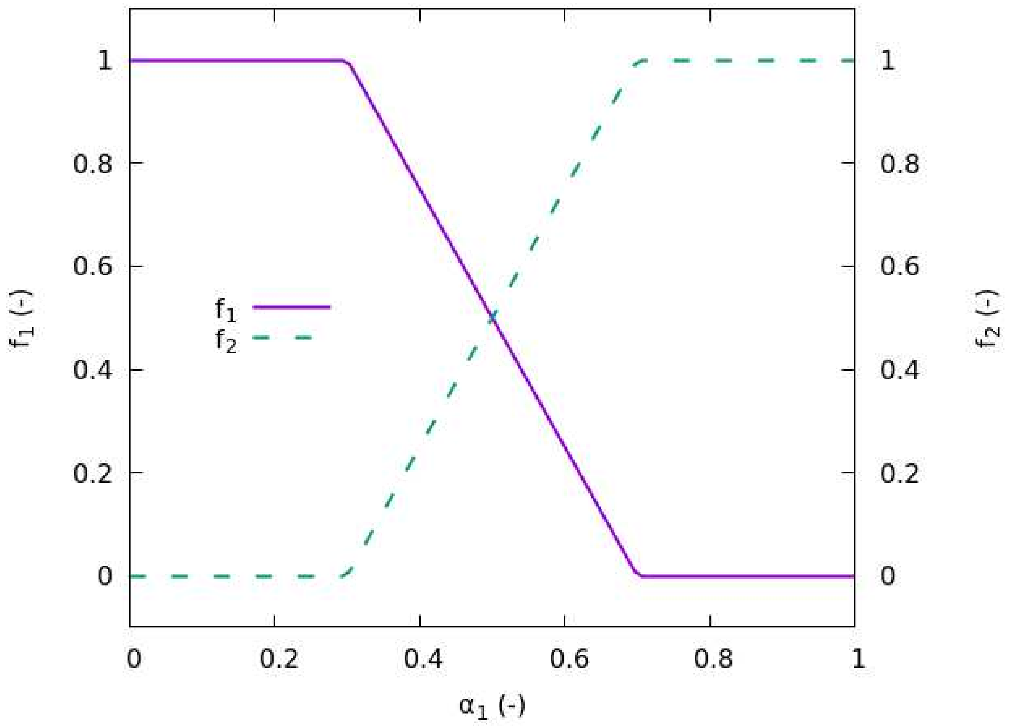


Figure 3.13: Blending factors f_1 and f_2 calculated using linear blending method. $\alpha_{1,F} = 0.7$, $\alpha_{1,P} = 0.3$, $\alpha_{2,F} = 0.7$, and $\alpha_{2,P} = 0.3$.

Table 3.7: Specifications of parameters for linear blending test cases.

Case		Air	Water
01	$\alpha_{1,F}$	0.97	N/A
	$\alpha_{1,P}$	0.97	N/A
	$\alpha_{2,F}$	N/A	0.03
	$\alpha_{2,P}$	N/A	0.03
02	$\alpha_{1,F}$	0.90	N/A
	$\alpha_{1,P}$	0.90	N/A
	$\alpha_{2,F}$	N/A	0.10
	$\alpha_{2,P}$	N/A	0.10
03	$\alpha_{1,F}$	0.70	N/A
	$\alpha_{1,P}$	0.70	N/A
	$\alpha_{2,F}$	N/A	0.30
	$\alpha_{2,P}$	N/A	0.30

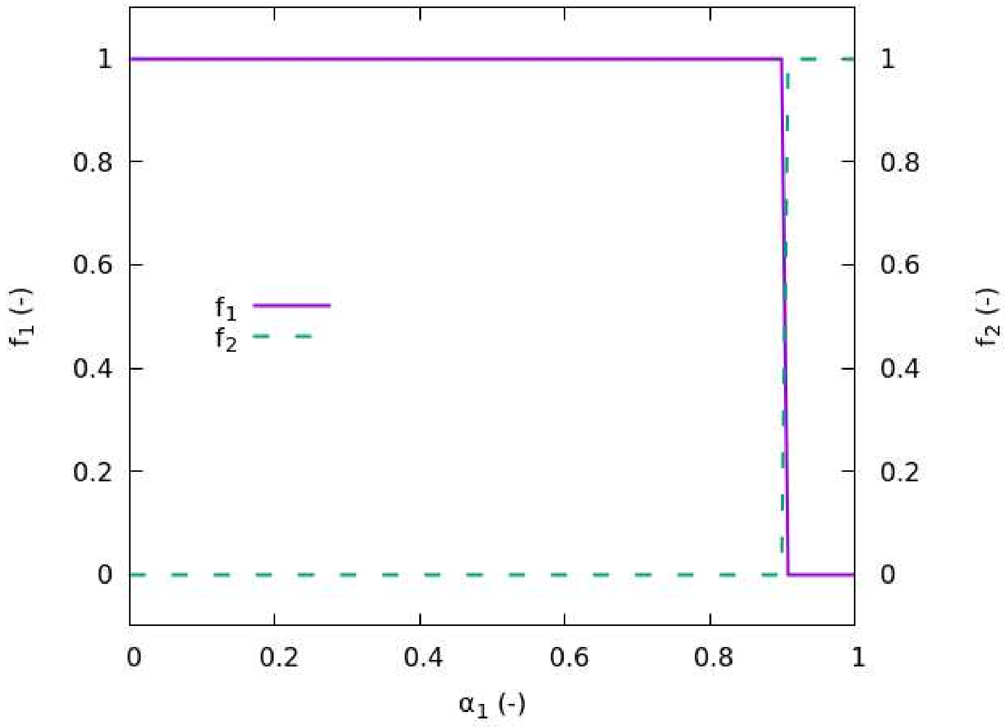


Figure 3.14: Blending factors f_1 and f_2 calculated using linear blending method for case 02 in Table 3.7.

Results of test case 01 in Table 3.7 are presented in Figure 3.15 and Figure 3.16. The results are similar to those of the base case, indicating that the blending method employed for test case 01 in Table 3.7 is a valid solution to the phase inversion problem.

The results of all test cases in Table 3.7 are compared with the base case result in terms of time-averaged overall liquid hold-up, shown in Figure 3.17. It shows that although the blending method solves the numerical instability introduced by phase inversion problem, the simulation results varies significantly with different specified values of the blending parameters. Only the the case 01 in Table 3.7 gives comparable results with the base case. This indicates that the values of blending parameters should be set according to a specific manner (see Table 3.7) such that the momentum transfer term due to drag force will be calculated mainly based on equation (2.21) and blending with the `SchillerNaumann` model only occurs at regions with very high gas phase fractions to maintain physical behaviour.

Another approach to over come the phase inversion problem is to introduce a weighting factor into the original drag model as presented in equation (2.70). Figure 3.18 illustrates the working mechanism of this method. The weighting factor magnifies the predicted drag force for regimes where dispersed phase fraction is lower than the predicted balance value at convergence and suppresses the predicted drag force for regimes where dispersed phase fraction is higher than the predicted balance value at convergence. Therefore, this weighting factor works in a similar way as the blending method does, without the necessity of introducing new drag models. It also promotes the solution stability in regions with high phase fraction gradient.

Two working approaches—i.e., Blending method and modified drag model—were explored for overcoming the phase-inversion problem. The blending method introduces a new drag model that requires input of dispersed phase diameter. This eliminates the advantage of calculating drag force without using the phase diameters by employing the drag model proposed by Krishna et al. [23] alone. Also, the parameters of blending method need optimization to maintain physical behaviour. Unless otherwise indicated, all subsequent simulations conducted in this study use the modified drag model as presented in equation (2.70) without blending.

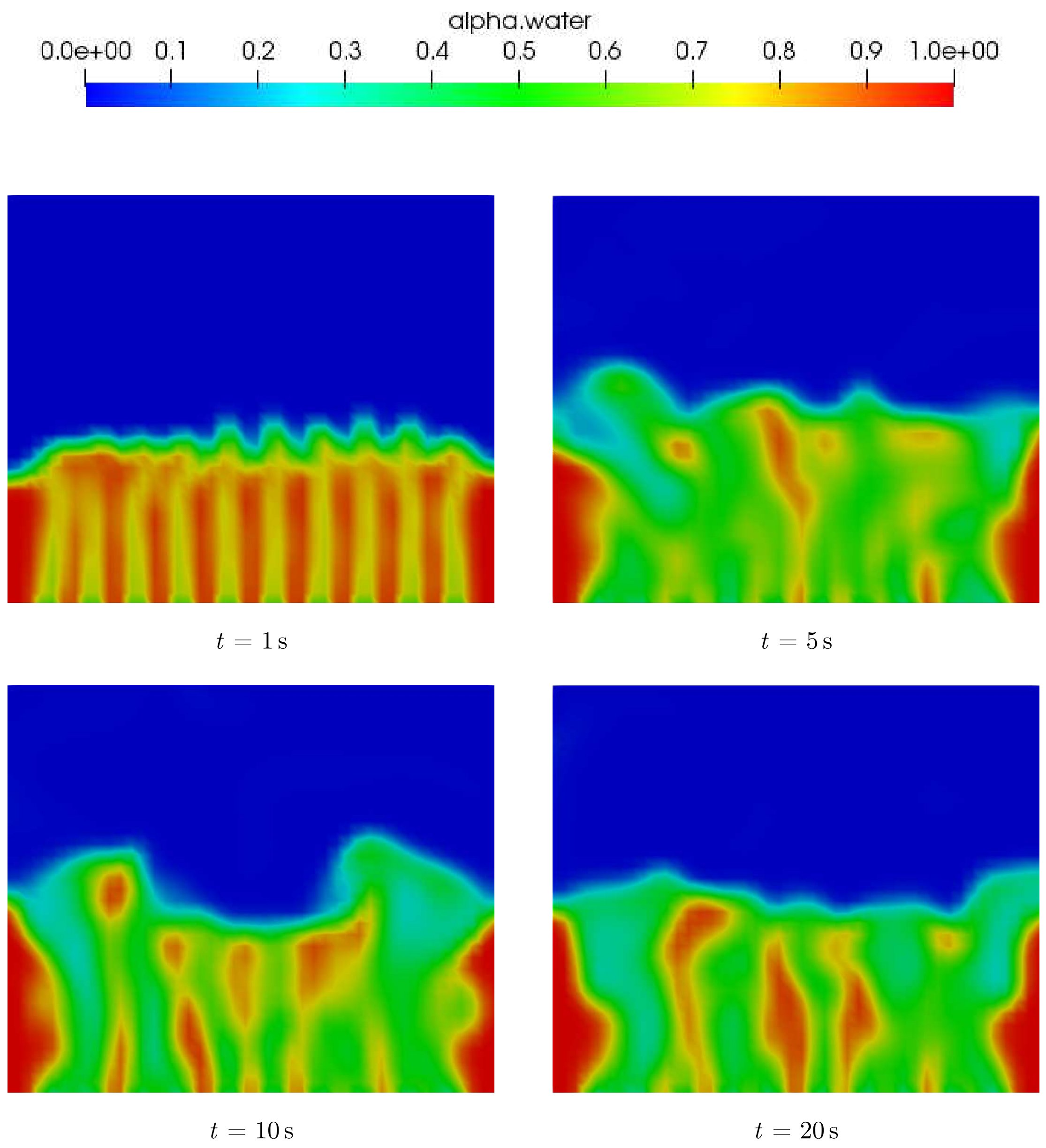
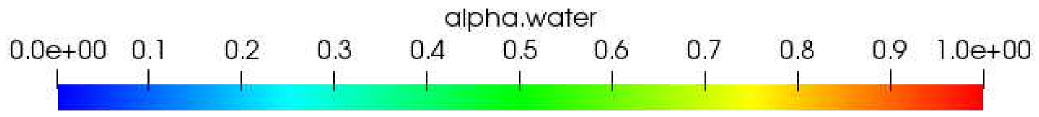
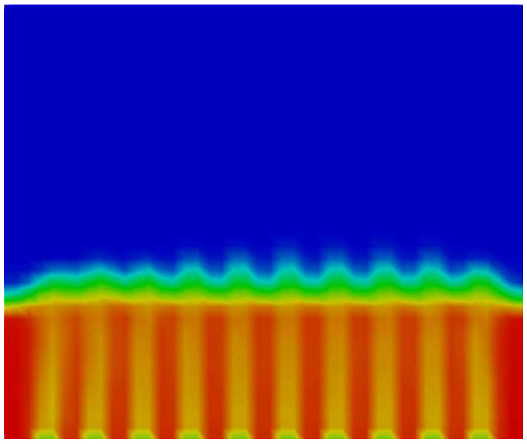
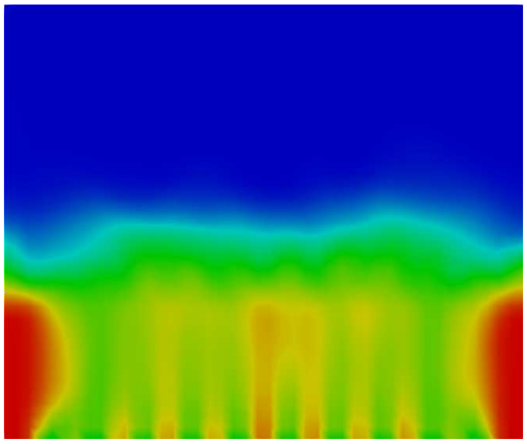
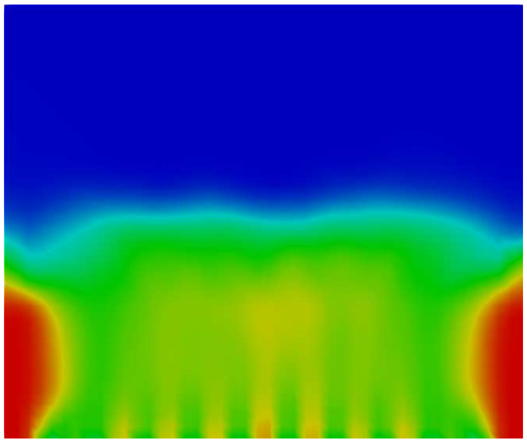


Figure 3.15: Instantaneous liquid phase fraction profiles on the XZ plane across the tray center at different simulated times for test case 01 in Table 3.7.




(a) $t = 0\,\mathrm{s}$ to $t = 1\,\mathrm{s}$


(b) $t = 0\,\mathrm{s}$ to $t = 5\,\mathrm{s}$


(c) $t = 0\,\mathrm{s}$ to $t = 10\,\mathrm{s}$

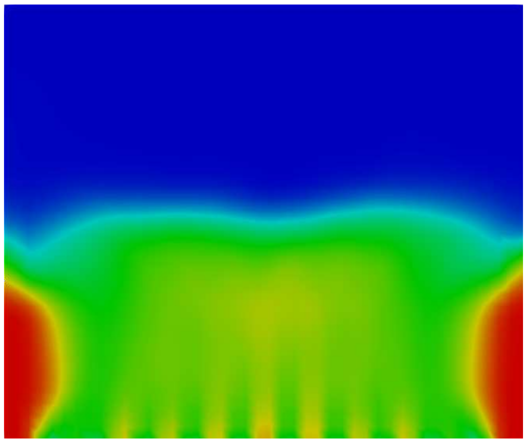

(d) $t = 0\,\mathrm{s}$ to $t = 20\,\mathrm{s}$

Figure 3.16: Time-averaged liquid phase fraction profile on the XZ plane across the tray center using different time intervals for test case 01 in Table 3.7.

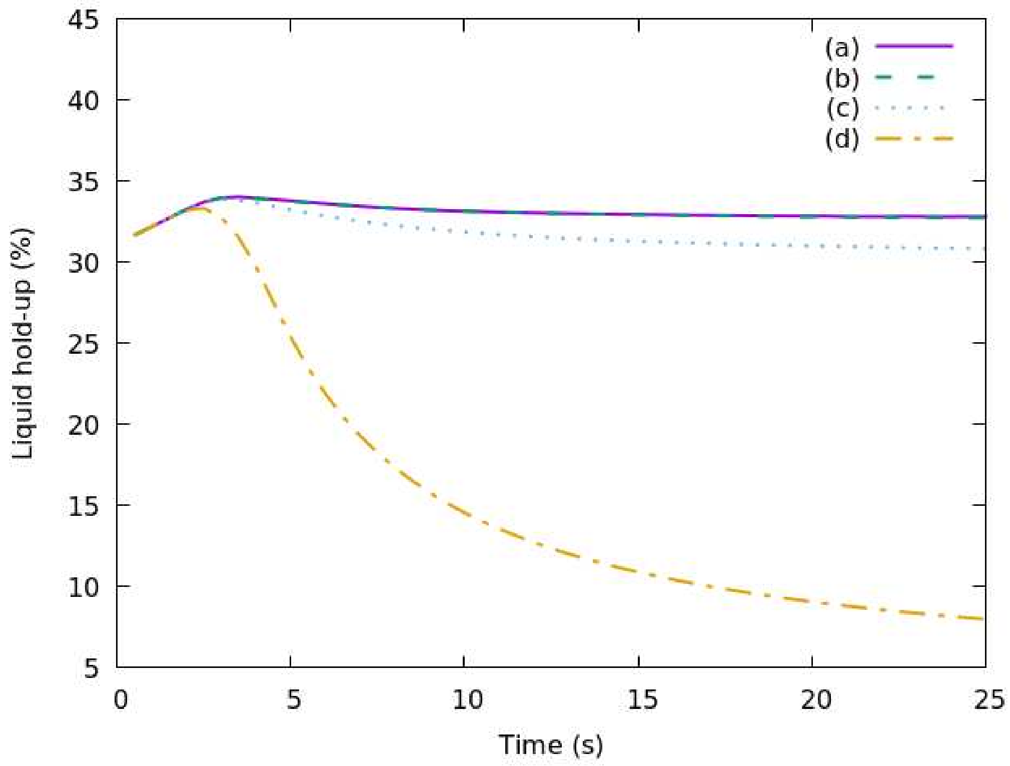


Figure 3.17: Time-averaged overall liquid hold-up over the time range from $t = 0\,\mathrm{s}$ to time t. (a) base case, (b) case 01 in Table 3.7, (c) case 02 in Table 3.7, (d) case 03 in Table 3.7.

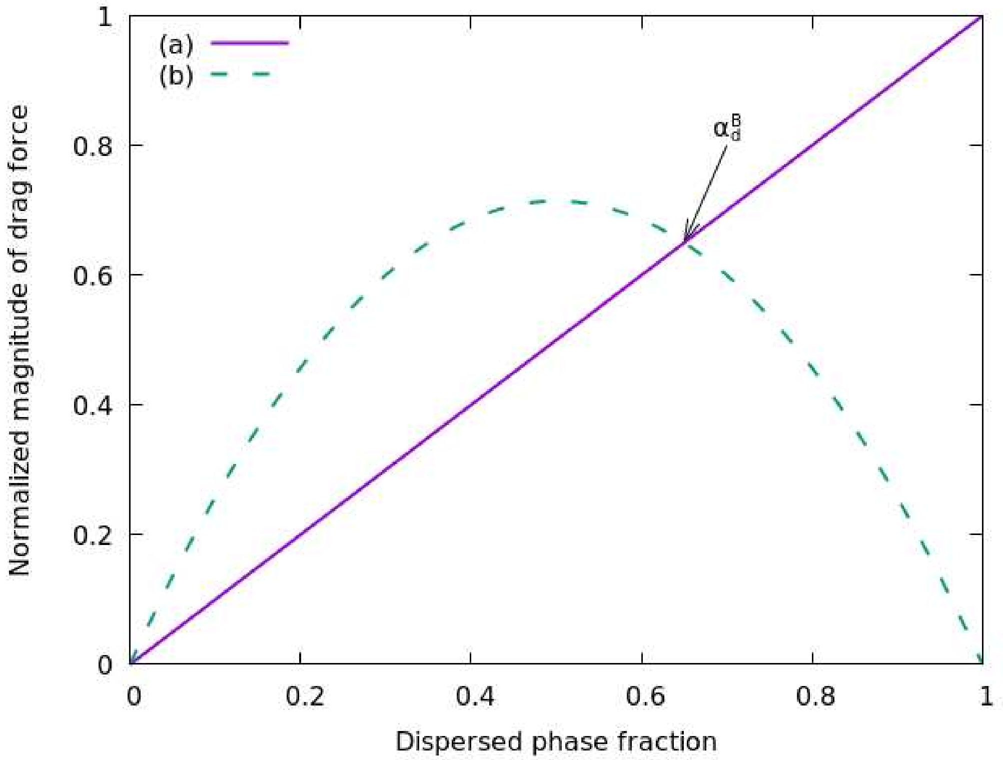


Figure 3.18: Drag force predicted using the model proposed by Krishna et al. [23]. (a) In the original form, as presented in equation (2.21), (b) In the modified form, as presented in equation (2.70). α_d^B is the average gas hold-up in the froth when system operates at a quasi-steady-state condition, calculated using the correlation proposed by Bennett et al. [5] (see equation (2.22)).

3.5 Turbulence model case study

As discussed in Section 1.2, the majority of previous studies on sieve tray simulation employed the standard $k - \epsilon$ turbulence model for the liquid phase. However, this turbulence model can introduce numerical instability when applied to mutiphase flow simulations due to the phase inversion problem. In OpenFOAM®, the governing equations of the standard $k - \epsilon$ turbulence model are formulated using the phase fraction field; therefore, the discretized transport equations face numerical problems in the computing regions where the phase fraction value tends towards zero. Holzinger [17] suggested that a preconditioned biconjugate gradient solver without preconditioner could be used as the linear solver for the discretized turbulence transport equations to overcome the numerical problem. In this study a relatively newly developed turbulence model, `mixtureKEpsilon`, was employed. The reason of choosing this turbulence model is to avoid numerical instability when phase inversion occurs as expressed in Section 2.3.

Case studies were carried out using different turbulence models and linear solver settings to test numerical stability. The results are presented in Table 3.8. The `mixtureKEpsilon` turbulence model gives stable simulations with various linear solvers. On the other hand, the `kEpsilon` model always leads to unstable simulations unless a preconditioned biconjugate gradient solver, `PBiCGStab`, is used without a preconditioner. It was also observed that, although the simulation using `PBiCGStab` solver without preconditionerisis stable, it requires significantly more iterations to solve the discretized transport equations for turbulence properties when compared to other working cases.

The simulation results were compared in terms of time-averaged overall liquid hold-up as presented in Figure 3.19. It shows that the results of both cases using different turbulence models converged to quasi-steady-states. The value of converged overall liquid hold-up given by the case—i.e., Case 06 in Table 3.8—using standard $k - \epsilon$ turbulence model for the liquid phase and laminar model for the gas phase is slightly lower than the value given by the case—i.e., Case 01 in Table 3.8—using the `mixtureKEpsilon` model. One of the potential causes of this difference is that the `mixtureKEpsilon` model took the contribution of gas phase into consideration when evaluating the turbulence, which accounts more dissipation and less interphase

Table 3.8: Results of turbulence model case studies.

Case	Turbulence model	Linear solver for turbulence	Stability
01	water: `mixtureKEpsilon` air: `mixtureKEpsilon`	solver: `smoothSolver` smoother: `symGaussSeidel`	stable
02	water: `mixtureKEpsilon` air: `mixtureKEpsilon`	solver: `PBiCGStab` preconditioner: `DILU`	stable
03	water: `mixtureKEpsilon` air: `mixtureKEpsilon`	solver: `PBiCGStab` preconditioner: `none`	stable
04	water: `kEpsilon` air: `laminar`	solver: `smoothSolver` smoother: `symGaussSeidel`	unstable
05	water: `kEpsilon` air: `laminar`	solver: `PBiCGStab` preconditioner: `DILU`	unstable
06	water: `kEpsilon` air: `laminar`	solver: `PBiCGStab` preconditioner: `none`	stable

momentum transfer and therefore less liquid phase was carried away by the gas phase.

Unless otherwise indicated, all subsequent simulations conducted in this study use the The `mixtureKEpsilon` turbulence model for turbulence evaluation.

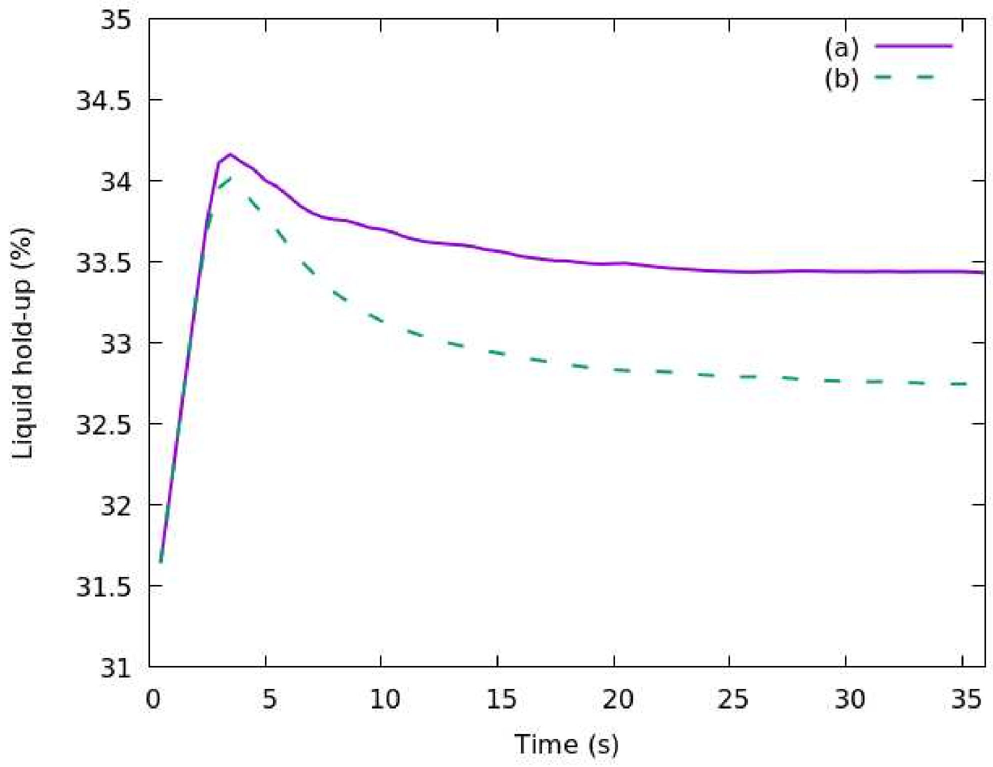


Figure 3.19: Time-averaged overall liquid hold-up over the time range from $t = 0\,\mathrm{s}$ to time t. (a) Case 06 in Table 3.8, (b) Case 01 in Table 3.8.

Chapter 4

Experimental-Scale Tray Simulation

CFD simulations of an experimental-scale tray under various operating conditions were conducted using the OpenFOAM® solver `reactingTwoPhaseEulerFoam` with the optimized settings described in Chapter 3. The results were validated with experimental data and CFD simulations from the work of Krishna et al. [22]. The simulation procedure and results are presented in this chapter.

4.1 Case setup

4.1.1 Simulation geometry and mesh

Figure 4.1 shows the configuration of the experimental-scale tray that was simulated. The design parameters of this tray presented in Table 4.1 are adapted from the simulation study of Krishna et al. [22], but the height of the simulation domain was extended from 0.12 m to 0.18 m for better numerical convergence as explained in Section 3.1.2.

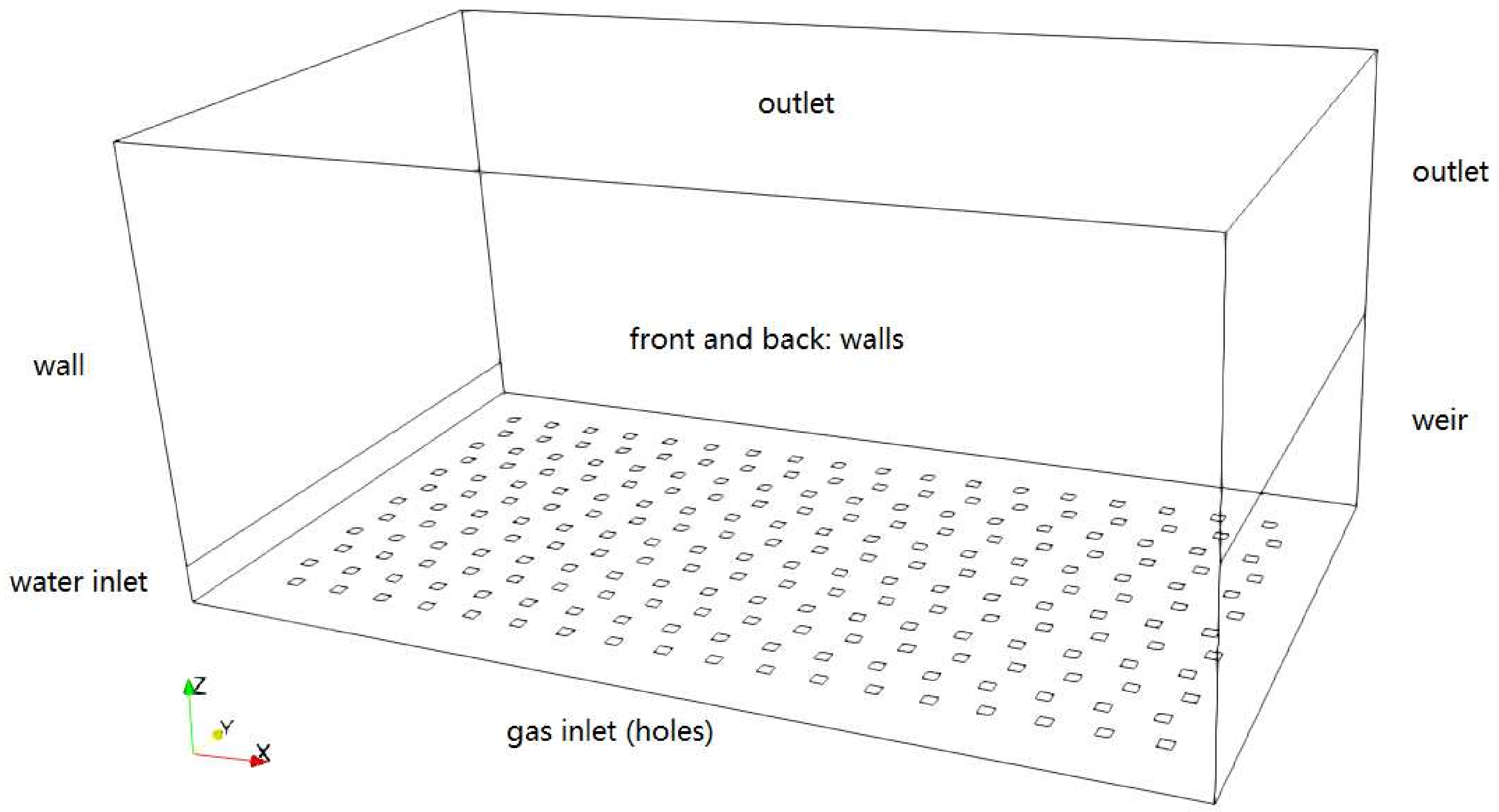


Figure 4.1: Schematic layout of the experimental-scale sieve tray.

Table 4.1: Design parameters of the experimental-scale sieve tray.

Design parameter	Value
Tray length	0.39 m
Tray height	0.18 m
Tray depth	0.22 m
Downcomer clearance	0.015 m
Weir height	0.06 m, 0.08 m, and 0.10 m
Number of holes	216
Square Hole size	0.005 m $\times$ 0.005 m

A fully orthogonal hexahedral grid was created based on this geometry. The total number of cells within the computing domain is 123 552, with a uniform grid size of 5 mm. The cell size of this grid is the same as the one used in the simulation work of Krishna et al. [22], which has passed their grid sensitivity tests.

4.1.2 Boundary and initial conditions

The mesh topology of the simulated experimental-scale tray is identical to that of the base case described in Chapter 3. Therefore, the type of boundary conditions employed in all the simulation cases in this chapter are kept the same as the ones described in Section 3.1.3, with their numerical values adjusted accordingly. Various values of superficial gas velocity, liquid load, and weir height were used in simulations. These values are summarized in Table 4.2.

In this study, the tray was initially filled with a water and air mixture from the bottom of the tray up to the weir height. The volume fraction of water in this region was set to be 0.8. The rest of the computational domain was filled with pure air phase. The velocity fields of both phases within the tray were initialized to be stationary.

Table 4.2: Specifications of operating parameters for experimental-scale tray simulations.

Operating parameter	Tested values
Superficial gas velocity	$0.4\,\mathrm{m\,s^{-1}}$ $0.7\,\mathrm{m\,s^{-1}}$ $0.9\,\mathrm{m\,s^{-1}}$
Liquid load per weir length	$4.0 \times 10^{-4}\,\mathrm{m^2\,s^{-1}}$ $8.25 \times 10^{-4}\,\mathrm{m^2\,s^{-1}}$ $1.2 \times 10^{-4}\,\mathrm{m^2\,s^{-1}}$
Weir height	$0.06\,\mathrm{m}$ $0.08\,\mathrm{m}$ $0.10\,\mathrm{m}$

4.1.3 Solver setup

The specifications for physical properties and other sub-models employed for the experimental-scale tray simulations are the same as the ones used for the base case as described in Section 3.1.4.

The discretization schemes used for all the simulation cases in this chapter are the same as the ones used for the base case, which are summarized in Table 3.4.

A set of linear solver specifications was constructed through trial and error, and the final settings are summarized in Table 4.3. These specifications provided a good balance between robustness and convergence speed for this particular simulation problem. It should be noted that two convergence criteria were used for the pressure variable, `p_rgh`. A more relaxed tolerance criterion was used for the intermediate corrector steps and a tighter tolerance was used for the last step. This procedure helped to speed up the simulations by putting most of the computational effort into the last corrector step, denoted by `p_rghFinal`.

For experimental-scale tray simulations, `nOuterCorrectors` and `nCorrectors` were set to be 3 and 2, respectively. `nNonOrthogonalCorrectors` was set to be 0. No under-relaxation was applied to the system of equations. The variable time stepping method was employed and controlled by Courant number with its maximum value set to be 0.5.

Table 4.3: Specifications of linear solvers for experimental-scale tray simulations.

Flow field	Linear solver	Tolerance
Phase fraction	solver: `PBiCGStab` smoother: `DILU`	1e-8
`p_rgh`	solver: `GAMG` smoother: `GaussSeidel`	1e-4
`p_rghFinal`	solver: `GAMG` smoother: `GaussSeidel`	1e-7
Velocity	solver: `PBiCGStab` smoother: `DILU`	1e-8
Energy	solver: `PBiCGStab` smoother: `DILU`	1e-8
Turbulence	solver: `PBiCGStab` smoother: `DILU`	1e-8

4.1.4 Parallel computing in OpenFOAM

Due to the considerably larger grid size used in the simulations in this chapter, parallel computing was employed to speed up the simulations. OpenFOAM$^{®}$ supports parallel computing through the method of domain decomposition [13], in which the grid and fields are decomposed into pieces and each of them is allocated to individual processors for computing. OpenFOAM$^{®}$ provides four methods of decomposition. In this study, the `Scotch` method was used, which requires no geometric input from the user and uses a strategy that attempts to minimize the number of boundaries between the decomposed grid on processors [13].

Simulation cases were conducted using parallel computing with different number of processors. The execution time needed to complete a simulation time period of 5 seconds for each test was recorded and the results are presented in Table 4.4.

It can be observed that, within the tested range, the simulation runs faster with an increased number of processors. However, the speed-up between simulations running on 32 processors and 16 processors is less significant than the speed-up between simulations running on 16 processors and 8 processors. This is because parallel computing

Table 4.4: Results of parallel computing case studies.

Case	Number of processors	Execution time	Relative speed-up (compared to 8 processors)
01	8	10 108 s	0%
02	16	6769.86 s	49.31%
03	32	5673.85 s	78.15%

Table 4.5: Results of `renumberMesh` utility case studies.

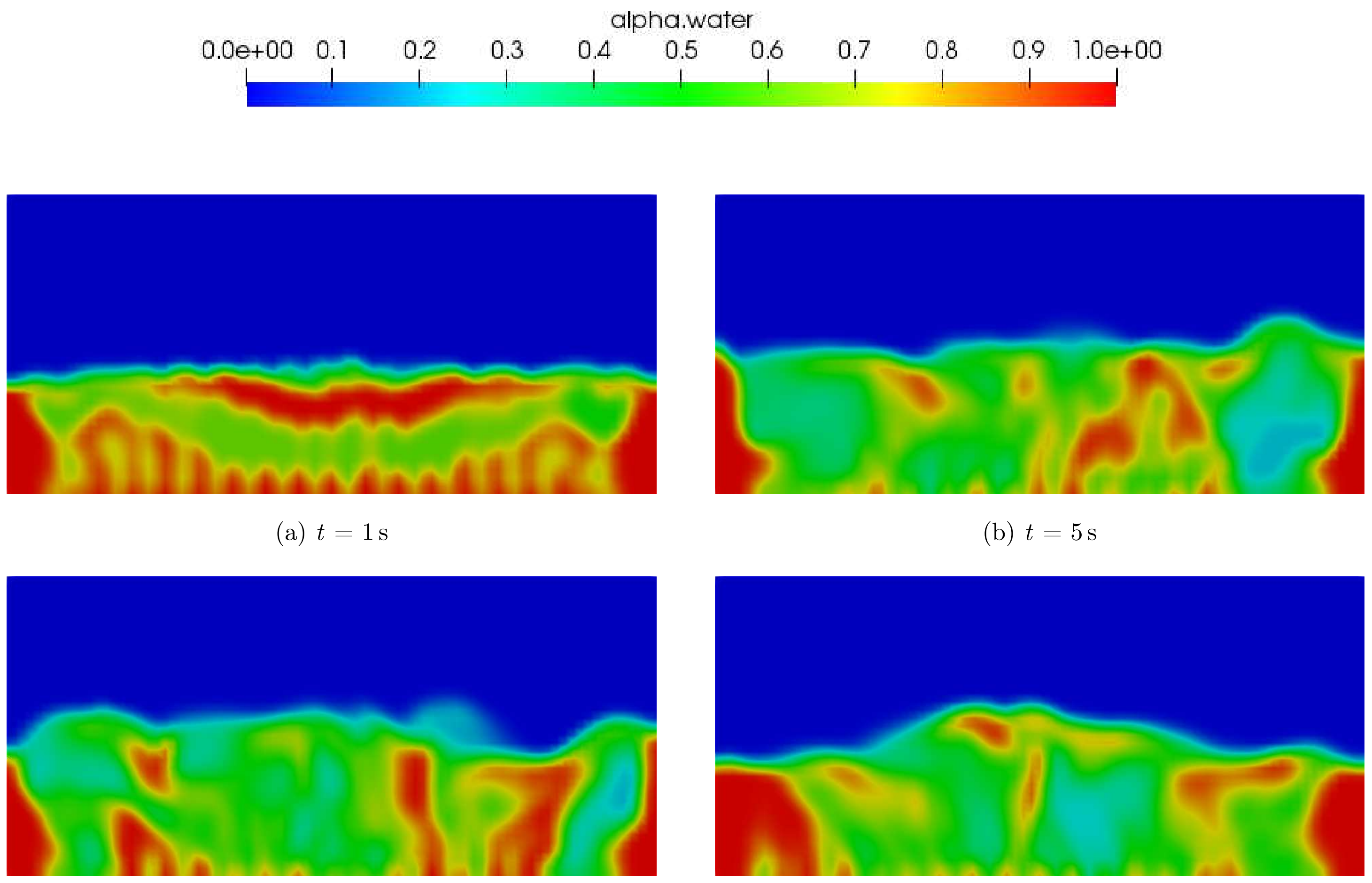
Case	# of processors	`renumberMesh`	Execution time	Relative speed-up
01	32	No	5673.85 s	0%
02	32	Yes	5122.88 s	10.76%

with more processors comes with heavier demand of data communication between the processors, which may eventually offset the advantage of using more processors.

An OpenFOAM® utility named `renumberMesh` was used to further reduce the simulation time. This utility reduces the bandwidth of the large sparse coefficient matrices produced from discretization by renumbering the cell label list [13]. Table 4.5 illustrates the improvement in speed after using `renumberMesh`. Unless otherwise indicated, all simulation cases presented in this chapter after this section were execture after running `renumberMesh` and carried out using parallel computing on 32 processors.

4.2 Results and discussions

The results of an experimental-scale tray simulation case with the superficial gas velocity of $0.5\,\mathrm{m\,s^{-1}}$, the weir height of $0.8\,\mathrm{m}$, and the liquid load per weir length of $8.25 \times 10^{-4}\,\mathrm{m^3s^{-1}m^{-1}}$ are presented in Figure 4.2 and 4.3. The overall liquid hold-up of this specific case was monitored and presented in Figure 4.4. This specific simulation is deemed to have converged to the quasi-steady-state after 40 seconds of simulated time.



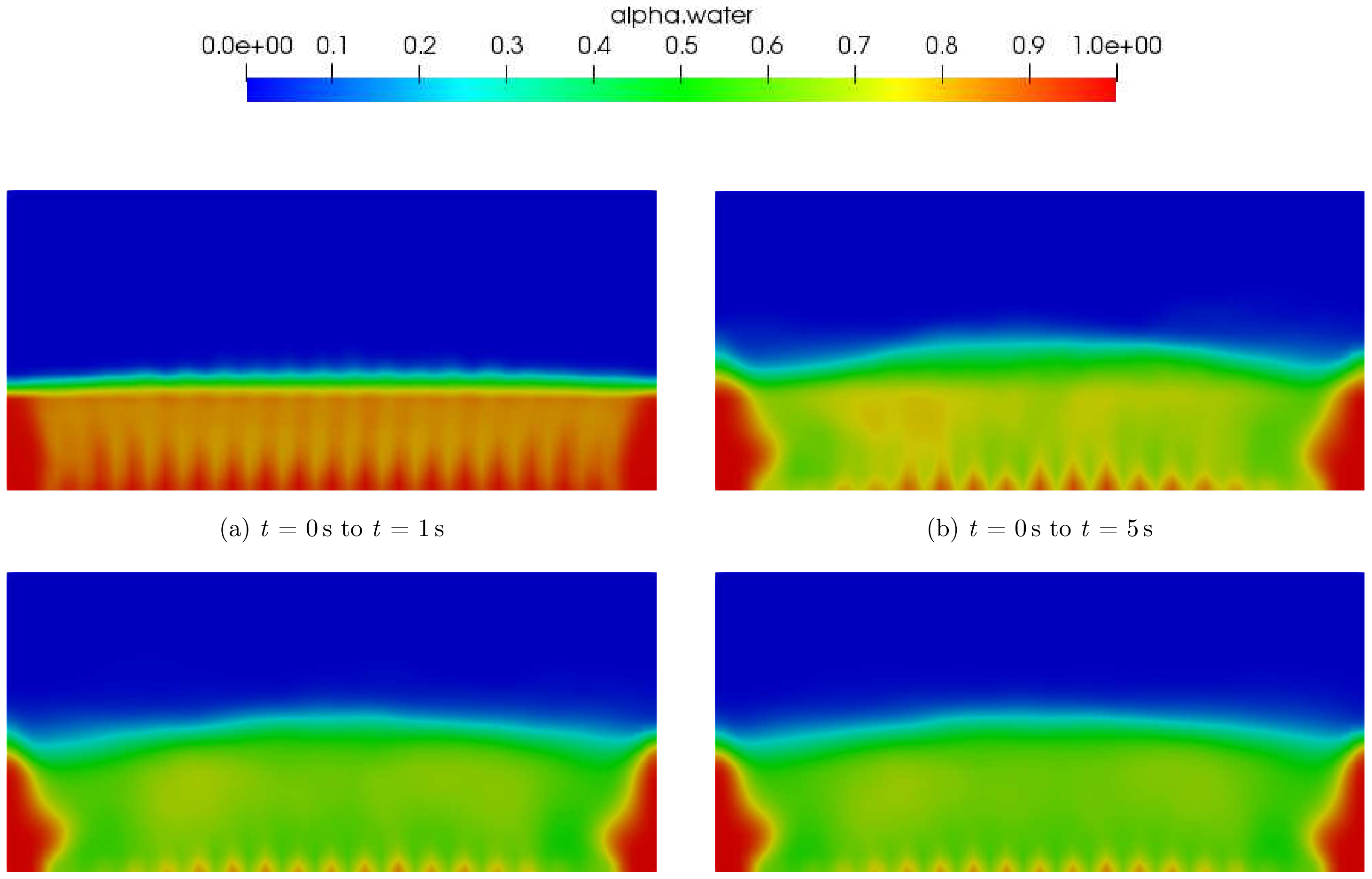
(a) $t = 1\,\mathrm{s}$

(b) $t = 5\,\mathrm{s}$

(c) $t = 10\,\mathrm{s}$

(d) $t = 40\,\mathrm{s}$

Figure 4.2: Instantaneous liquid phase fraction profiles on the XZ plane across the tray center at different simulated times for an experimental-scale sieve tray simulation with the superficial gas velocity of $0.5\,\mathrm{m\,s^{-1}}$, the weir height of $0.8\,\mathrm{m}$, and the liquid load per weir length of $8.25 \times 10^{-4}\,\mathrm{m^3 s^{-1} m^{-1}}$.



(a) $t = 0\,\mathrm{s}$ to $t = 1\,\mathrm{s}$

(b) $t = 0\,\mathrm{s}$ to $t = 5\,\mathrm{s}$

(c) $t = 0\,\mathrm{s}$ to $t = 10\,\mathrm{s}$

(d) $t = 0\,\mathrm{s}$ to $t = 40\,\mathrm{s}$

Figure 4.3: Time-averaged liquid phase fraction profiles on the XZ plane across the tray center over different time intervals for an experimental-scale sieve tray simulation with the superficial gas velocity of $0.5\,\mathrm{m\,s^{-1}}$, the weir height of $0.8\,\mathrm{m}$, and the liquid load per weir length of $8.25 \times 10^{-4}\,\mathrm{m^3 s^{-1} m^{-1}}$.

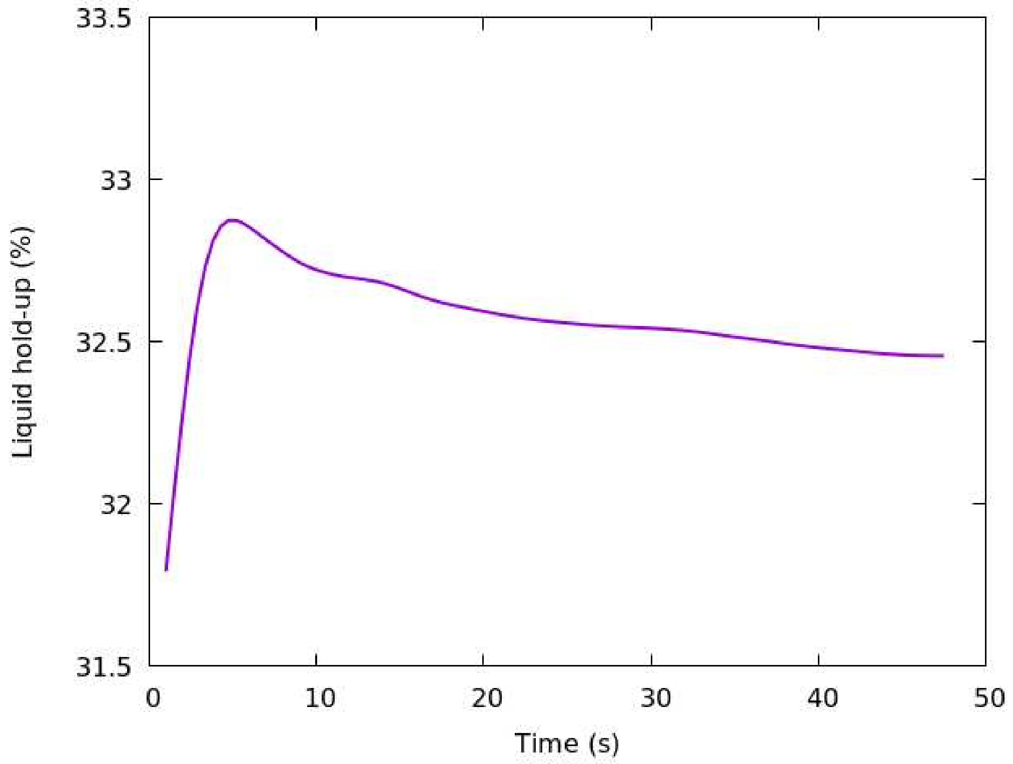


Figure 4.4: Time-averaged overall liquid hold-up over the time range from $t = 0\,\mathrm{s}$ to time t for an experimental-scale sieve tray simulation with the superficial gas velocity of $0.5\,\mathrm{m\,s^{-1}}$, the weir height of $0.8\,\mathrm{m}$, and the liquid load per weir length of $8.25 \times 10^{-4}\,\mathrm{m^3 s^{-1} m^{-1}}$.

Clear liquid height was determined by multiplying the time-averaged overall liquid hold-up at the quasi-steady-state by the height of the tray. Figures 4.5, 4.6, and 4.7 present the comparison of the simulation results of this study with the experimental and CFD simulation data from the work of Krishna et al. [22] in terms of clear liquid height under various operating conditions. The clear liquid height was predicted to decrease with increased superficial gas velocity at given weir height and liquid load, as illustrated in Figure 4.5. The clear liquid height was predicted to increase with increased weir height at given superficial gas velocity and liquid load, as shown in Figure 4.6. The clear liquid height was predicted to increase with increased liquid load at a given superficial gas velocity and weir, as shown in Figure 4.7. These trends are consistent with reference CFD simulations and experimental data. The predicted clear liquid heights from experimental-scale tray simulations were therefore in good quantitative agreement with the CFD simulation results of Krishna et al. [22].

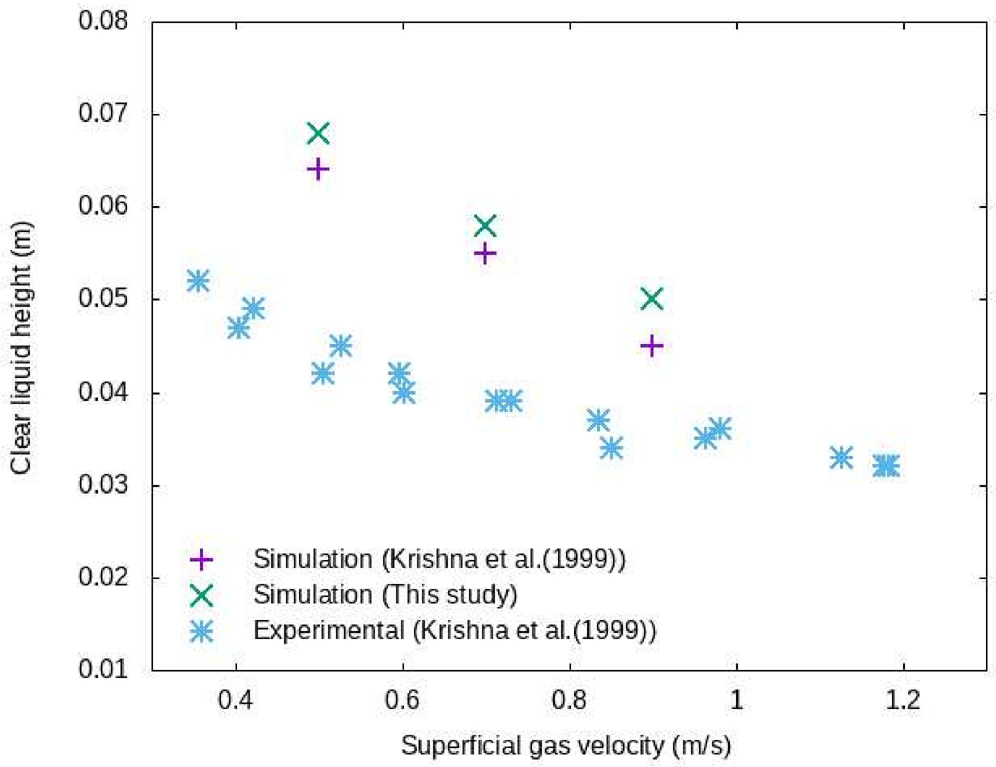


Figure 4.5: Clear liquid heights at different superficial gas velocities for experimental-scale sieve tray simulations with the weir height of 0.8 m and the liquid load per weir length of $8.25 \times 10^{-4}\,\mathrm{m^3s^{-1}m^{-1}}$.

It is noticeable that the predicted clear liquid heights from this study are slightly higher in values compared to the CFD simulation results of Krishna et al. [22] in most tested cases. The simulation setup employed in this study differs from that of Krishna's work mainly in two aspects: turbulence models and boundary conditions at the outlet boundary. It has been shown in Figure 3.19 that the turbulence model employed in this study, the mixture model, tends to slightly underestimate the clear liquid height compared to the turbulence model settings used in Krishna's CFD work. Therefore, the different boundary conditions utilized in this study and Krishna's CFD work are the major contributor to the differences in simulation results.

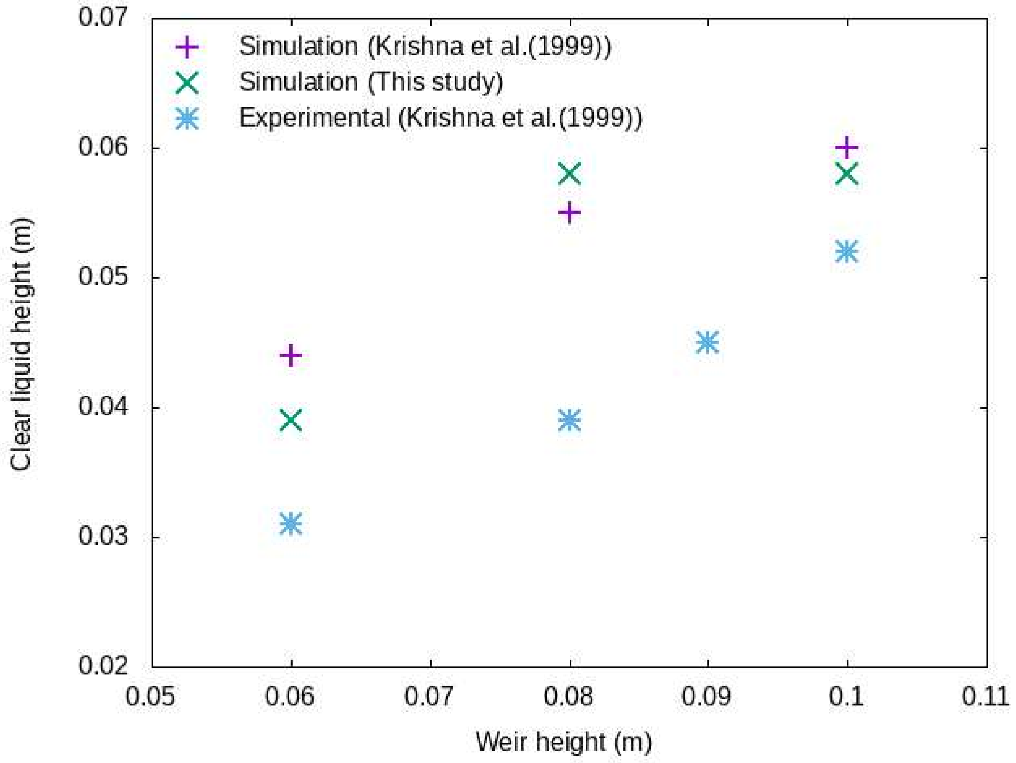


Figure 4.6: Clear liquid heights at different weir heights for experimental-scale sieve tray simulations with the superficial gas velocity of $0.7\,\mathrm{m\,s^{-1}}$ and the liquid load per weir length of $8.25 \times 10^{-4}\,\mathrm{m^3 s^{-1} m^{-1}}$.

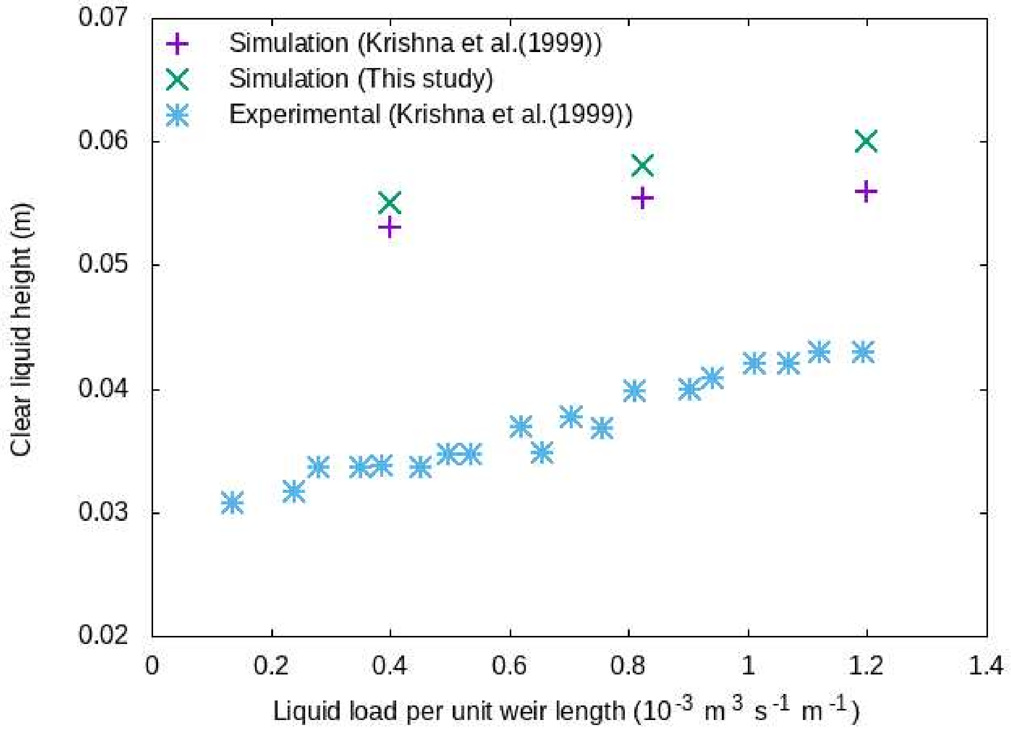


Figure 4.7: Clear liquid heights at different liquid loads for experimental-scale sieve tray simulations with the weir height of 0.8 m and the superficial gas velocity of $0.7\,\mathrm{m\,s^{-1}}$.

Chapter 5

Commercial-Scale Tray Simulation

The developed simulation technique was applied to a commercial-scale tray to test its capacity to make reasonable predictions when scaling from pilot-scale to large-scale systems. The results were validated with experimental data from the work Solari and Bell [45] and CFD simulation results from the work of Gesit et al. [10]. The simulation procedure and results are presented in this chapter.

5.1 Case setup

5.1.1 Simulation geometry and mesh

The design parameters of the simulated commercial-scale sieve tray are presented in Table 5.1. These parameters are based on the tray properties of the experimental work from Solari and Bell [45], which was again adopted in the CFD simulation work of Gesit et al. [10].

Table 5.1: Design parameters of the commercial-scale sieve tray.

Design Parameter	Value
Tray diameter	1.213 m
Tray spacing	0.61 m
Weir length	0.925 m
Weir height	0.05 m
Downcomer clearance	0.038 m
Hole diameter	0.0127 m
Fractional hole area to bubbling area	5%

In this work, one of the challenges is modeling the gas inlet holes because of their large quantity and small sizes. Instead of modelling the actual size and number of holes as used in the experiment, a reduced number of 90 holes was modelled and the fractional hole area to bubbling area was kept the same as that of the actual number of holes. Although this means that there are fewer point sources of gas on the tray, it does ensure that the gas velocity exiting each hole matches the true condition. Gesit et al. [10] reported that this simplification on gas inlet holes gives simulation results that are in good agreement with the results of simulations using the actual number of holes. Furthermore, Solari and Bell have observed symmetric flow patterns with respect to the tray center in their experimental work [45]. Thus, only half of the tray was modeled in the present simulations to save the computational cost. This method was also been utilized in the CFD simulation work of Gesit et al. [10].

Figure 5.1 shows the layout and computational grid used in the simulations of the commercial-scale tray. Water phase enters through the liquid inlet patch, which is marked yellow in the figure, and leaves through gas outlet patch, which is marked green in the figure. Air enters through the gas inlet patch located on the tray floor, which is marked blue, and leaves through the gas outlet patch located on top of the tray, which is marked green. The center plane of the tray is modelled as a symmetry plane and the rest of the patches are modelled as walls, which are marked grey. This grid contains 54 456 hexahedral cells. The size of cells are no larger than the one used in the simulation work of Gesit et al. [10], which has passed their grid sensitivity test.

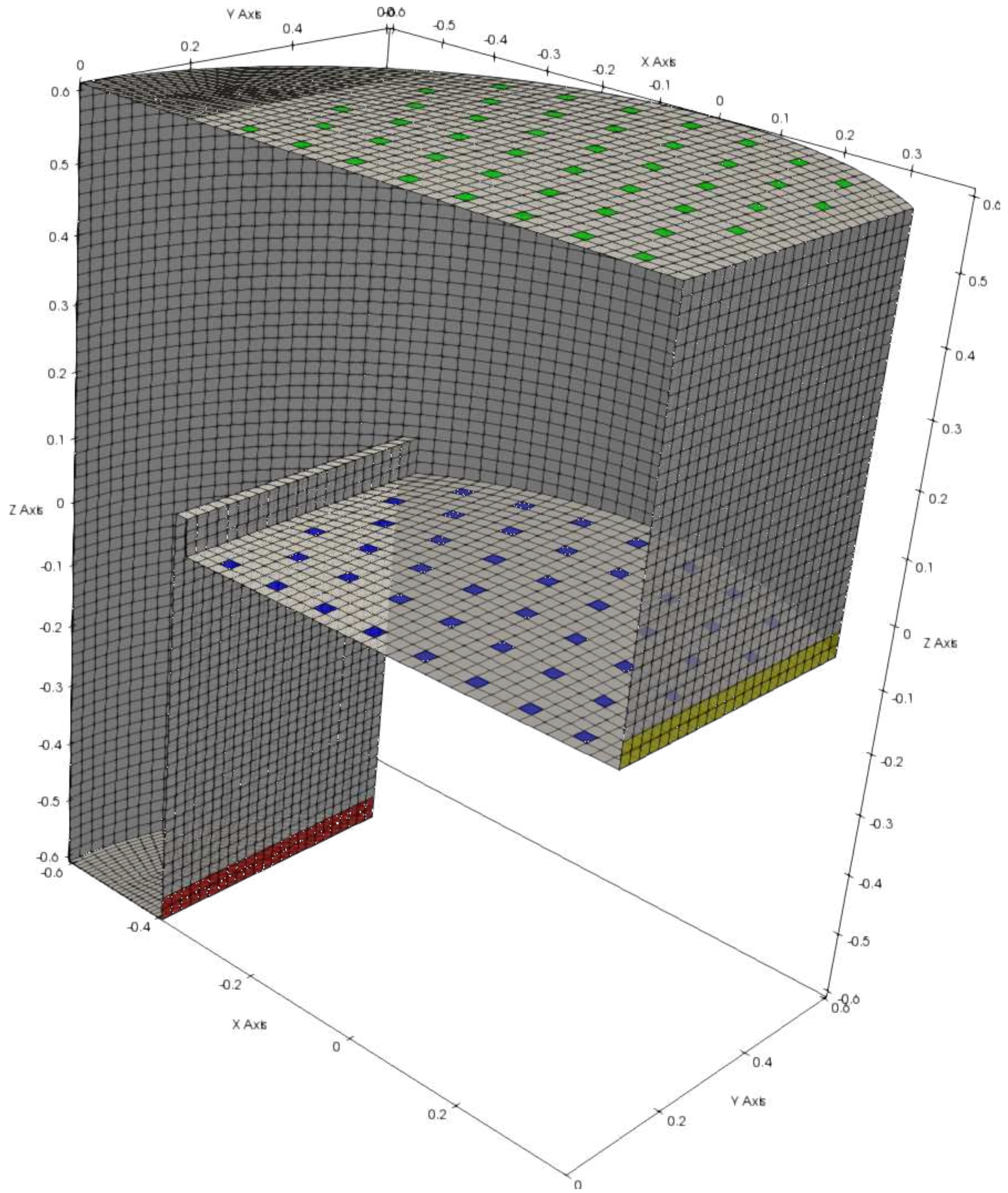


Figure 5.1: Flow geometry and computational grid for commercial-scale sieve tray simulations.

5.1.2 Boundary and initial conditions

It is noticeable that the computational grid for the commercial-scale tray simulation is similar to the one used for the base case described in Chapter 3, with the exception of two patches: the `liquid outlet`, and the `gas outlet`.

On the boundary patch of liquid outlet, the following boundary conditions were specified for each flow variable:

- `alpha.air`: `inletOutlet` boundary condition with the inlet value specified to be pure liquid.
- `p_rgh`: `fixedFluxPressure` condition.
- `U.air`: A no-slip wall boundary condition. Therefore, it is assumed that the gas phase does not penetrate the liquid outlet boundary.
- `U.water`: `matchedFlowRateOutletVelocity` condition. This boundary condition corrects the extrapolated velocity on the patch to match the flow rate of a specified patch that provides an inlet flow rate. In this case the corresponding inlet patch is the liquid inlet patch.

On the boundary patch of gas outlet, the following boundary conditions were specified for each flow variable:

- `alpha.air`: `inletOutlet` boundary condition with the inlet value specified to be pure gas.
- `p_rgh`: `prghTotalPressure` condition.
- `U.air`: `pressureInletOutletParSlipVelocity` condition.
- `U.water`: A no-slip wall boundary condition. Therefore, it is assumed that the liquid phase does not penetrate the gas outlet boundary.

Mehta et al. [33] developed a criterion for selecting the appropriate boundary condition for the liquid inlet velocity profile based on the value of a flow parameter, F_{lv}, calculated as follows:

$$F_{lv} = \frac{Q_L}{Q_G}\sqrt{\frac{\rho_L}{\rho_G}} \tag{5.1}$$

where Q_L and Q_G are the volumetric flow rates for liquid and gas, and ρ_L and ρ_G are liquid density and gas density. Mehta et al. [33] suggest that the liquid inlet velocity profile could be assumed to be uniform if the value of F_{lv} is less than 0.25. Otherwise, it should be modelled as a parabolic curve.

On the boundary patch of liquid inlet, the following boundary condition for `U.water` was thus specified:

- If $F_{lv} \geq 0.25$, the velocity is calculated as follows:

$$\mathbf{U}_{L,in} = \begin{bmatrix} \frac{1.5Q_L}{h_{ap}L_w}[1 - (\frac{2y}{L_w})^2] \\ 0 \\ 0 \end{bmatrix} \tag{5.2}$$

- If $F_{lv} < 0.25$, the velocity is calculated using the following expression:

$$\mathbf{U}_{L,in} = \begin{bmatrix} \frac{Q_L}{h_{ap}L_w} \\ 0 \\ 0 \end{bmatrix} \tag{5.3}$$

where h_{ap} is the downcomer clearance, L_w is the weir length, and y is the y-coordinate (see Figure 5.1).

The boundary condition described in equation (5.2) is not readily available in OpenFOAM®. Thus, it was implemented as a user-specified boundary condition using a utility named `codedFixedValue` that allows the user to implement complex boundary conditions without extensive C++ programming effort. The equivalent of equation (5.2) was implemented using the `codedFixedValue` utility, and the code is presented in Figure 5.2.

In the code presented in Figure 5.2 it should be noted that due to the orientation of the modelled geometry, a negative sign was added to the x component of the velocity vector to ensure that the flow flux pointed towards the inside of the computing domain.

The boundary conditions applied to other boundary patches are in consistent with the type of boundary conditions applied on the corresponding boundary patches in the base case simulation, but the numerical values were adjusted according to the operating conditions of the commercial-scale tray. The tested operating parameters are summarized in Table 5.2.

```
    inletL
    {
        type            codedFixedValue;
        value           uniform (0 0 0);

        redirectType parabolicInlet;

        code
        #{
            const fvPatch& boundaryPatch = patch();
            const vectorField& Cf = boundaryPatch.Cf();
            vectorField& field = *this;

            const scalar QL  = 0.0178;
            const scalar hap = 0.03812;
            const scalar Lw  = 0.92508;

            forAll(Cf, faceI)
            {
              const scalar y = Cf[faceI].y();
              field[faceI] = vector(-(1.5*QL)/(hap*Lw)*(1-pow((2*y)/Lw,2)), 0,
    0);
            }
        #};
    }

```

Figure 5.2: User-coded boundary condition using `codedFixedValue`.

Table 5.2: Specifications of operating parameters for commercial-scale tray simulations.

Operating parameter	Tested values
Superficial gas velocity	$0.421\,\mathrm{m\,s^{-1}}$
	$0.730\,\mathrm{m\,s^{-1}}$
	$0.925\,\mathrm{m\,s^{-1}}$
	$1.334\,\mathrm{m\,s^{-1}}$
Liquid load	$6.94 \times 10^{-3}\,\mathrm{m^3\,s^{-1}}$
	$1.24 \times 10^{-2}\,\mathrm{m^3\,s^{-1}}$
	$1.78 \times 10^{-2}\,\mathrm{m^3\,s^{-1}}$

In this study, the tray was initially filled with a water and air mixture from the bottom of the tray up to the height of 0.1 m above the tray floor. The volume fraction of air in this region was set to be a fixed value calculated using equation (2.22). The lower one third of the downcomer region was initially filled with pure water phase. The rest of the computational domain was filled with pure air phase. The velocity fields of both phases within the computing region were initialized to be stationary.

5.1.3 Solver setup

The specifications for physical properties and other sub-models employed for the commercial-scale tray simulations are the same ones used for the base case as described in Section 3.1.4.

The discretization schemes used for all of the commercial-scale tray simulations are summarized in Table 5.4. The scheme used for gradient terms is `cellMDLimited leastSquares 0.5`, which has second-order accuracy and suppresses numerical oscillation on the gradient computation. This scheme requires an explicitly specified coefficient with a value between 0 and 1, where 1 provides the strongest limiting. In this study, the coefficient was specified to be 0.5. For the divergence term `div(phi,alpha.air)`, which denotes the term $\nabla \cdot (\mathbf{U}\alpha_{air})$ in the discretized mass conservation equations, the `Gauss limitedLinear01 1` scheme was used. This scheme is specially formulated to bound the values of underlying variables—e.g, phase fraction α—between 0 and 1. This scheme requires an explicitly specified coefficient with a value between 0 and 1, where 1 provides the strongest limiting. In this study, the coefficient was specified to be 1. The `limited corrected 0.333` scheme was applied to surface normal gradient calculation and the surface normal gradient component of the Laplacian terms. This scheme maintains second-order accuracy on non-orthogonal grids by introducing an explicit non-orthogonal correction to the orthogonal component. This scheme also requires an explicitly specified coefficient with a value between 0 and 1, where 0 offers relatively best stability and 1 offers relatively best accuracy. In this study, the coefficient was specified to be 0.33.

The linear solver specifications are the same as those shown in Table 4.3.

For all commercial-scale tray simulations, `nOuterCorrectors` and `nCorrectors` of the `SIMPLE` algorithm were set to be 3 and 2, respectively. Because the mesh is not fully orthogonal, the `nNonOrthogonalCorrectors` was set to be 1. No under-relaxation was applied to the system of equations.

A fixed time step was employed for the commercial-scale tray simulations. The size of the time step was calculated using the tray operating conditions to give a Courant number that is no greater than 0.4. A fixed time step was used because, for these large-scale simulations, using the variable time stepping method caused unstable simulations even if the maximum allowed Courant number was set to be 0.4.

All simulations in this chapter were executed after running `renumberMesh` and carried out using parallel computing on 16 processors.

Table 5.3: Specifications of discretization schemes for the commercial-scale tray simulations.

Category	Syntax of related terms in the equations	Discretization scheme
Time	`ddt()`	`Euler`
Gradient	`grad()`	`cellMDLimited leastSquares 0.5`
Divergence	`div(phi,alpha.air)`	`Gauss limitedLinear01 1`
Divergence	`div(phir,alpha.air)`	`Gauss vanLeer`
Divergence*	`div(alphaRhoPhi,U)` `div(phi,U)`	`Gauss limitedLinearV 1`
Divergence*	`div(alphaRhoPhi,(h|e))` `div(alphaRhoPhi,K)` `div(alphaPhi,p)` `div(alphaRhoPhi,(k|epsilon))` `div(phim,(k|epsilon)m)`	`Gauss limitedLinear 1`
Divergence*	`div((alpha*rho*nuEff)*dev2(T(grad(U))))`	`Gauss linear`
Laplacian	`laplacian()`	`Gauss linear limited corrected 0.333`
Interpolation	N/A	`linear`
Surface normal gradient	`snGrad()`	`limited corrected 0.333`

* These schemes apply to equation terms for both phases—e.g. `div(phi.air,U.air)` and `div(phi.water,U.water)`—the suffixes are omitted for simplicity.

5.2 Results and discussions

A series of commercial-scale sieve tray simulations was conducted using the specified settings. Figure 5.3 shows the time-averaged overall liquid holdup in the system over time for a representative case of the commercial-scale tray simulations. This specific simulation is deemed to have converged to a quasi steady state after 60 seconds of simulated time. Figure 5.4 shows the liquid phase fraction profile of this specific simulation at the quasi-steady-state. Clearly, most of the liquid phase is present in the region near the tray floor. From the tray floor to the weir height, regions above the gas inlet holes have relatively higher gas phase fractions compared to the surrounding tray floor regions. The liquid phase fraction diminishes rapidly from the weir height up, leaving most of the tray space to be filled with gas phase only.

Clear liquid height was determined by multiplying the overall liquid phase fraction at the quasi-steady-state by the tray spacing. The predicted results of clear liquid height were compared against the CFD simulation results of Gesit et al. [10] and validated using the experimental data of Solari and Bell [45], as shown in Figure 5.5 and Figure 5.6. It is noticeable that, within the tested range, the results obtained from the present simulations were in excellent quantitative agreement with the CFD simulation results of Gesit et al. [10] at conditions with relatively lower superficial gas velocities. The predicted clear liquid height obtained in the present simulations showed improved accuracy compared to the CFD simulation results of Gesit et al. [10] at conditions in which the superficial gas velocities were relatively larger. Since the drag model and boundary conditions are largely the same in both CFD works, the deviations between the present simulation results and those of Gesit et al. [10] are mostly likely introduced by the different turbulence models and numerical settings employed in each work.

The local velocity profiles were compared against the reference CFD simulations [10] as well as experimental measurements [45].

In the experimental work of Solari and Bell [45], a set of probes was placed on a plane 0.038 m above the tray floor, as shown in Figure 5.7. The average linear liquid velocities were calculated by dividing the distance between probe 5 and probe 9 by the time required for the dye indicator to travel through these two probes. This procedure was repeated for probes 6 and 10, probes 7 and 11, and probes 8 and 12.

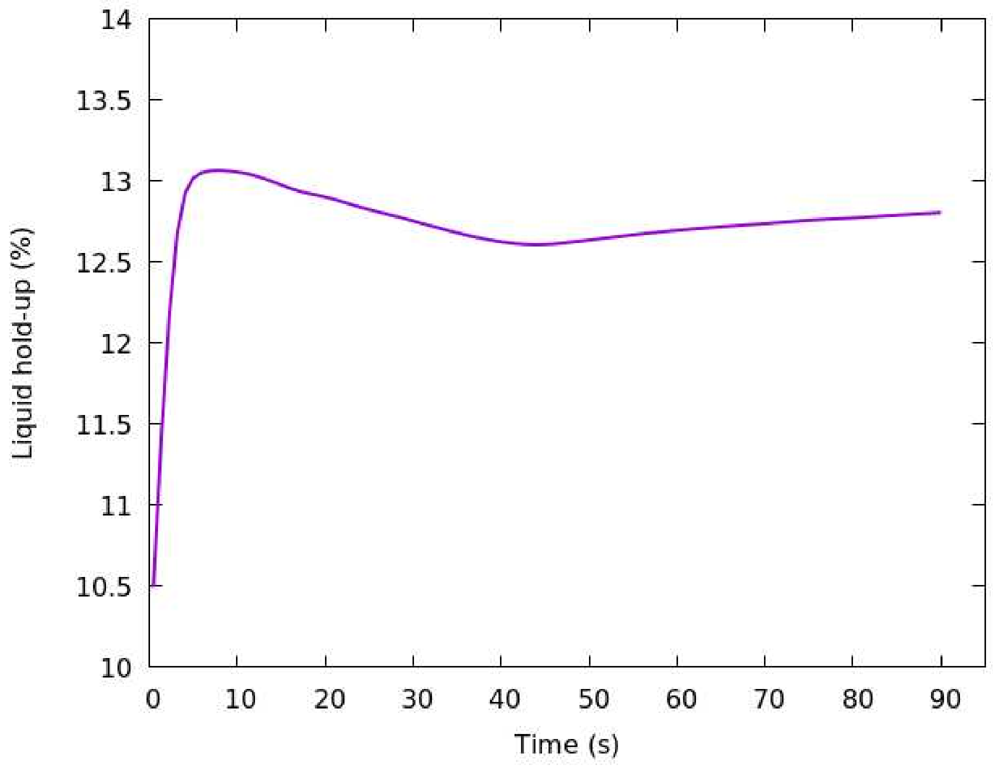


Figure 5.3: Time-averaged overall liquid hold-up over the time range from $t = 0\,\mathrm{s}$ to time t for a commercial-scale sieve tray simulation case with the superficial gas velocity of $0.421\,\mathrm{m\,s^{-1}}$ and the liquid load of $17.8 \times 10^{-3}\,\mathrm{m^3\,s^{-1}}$.

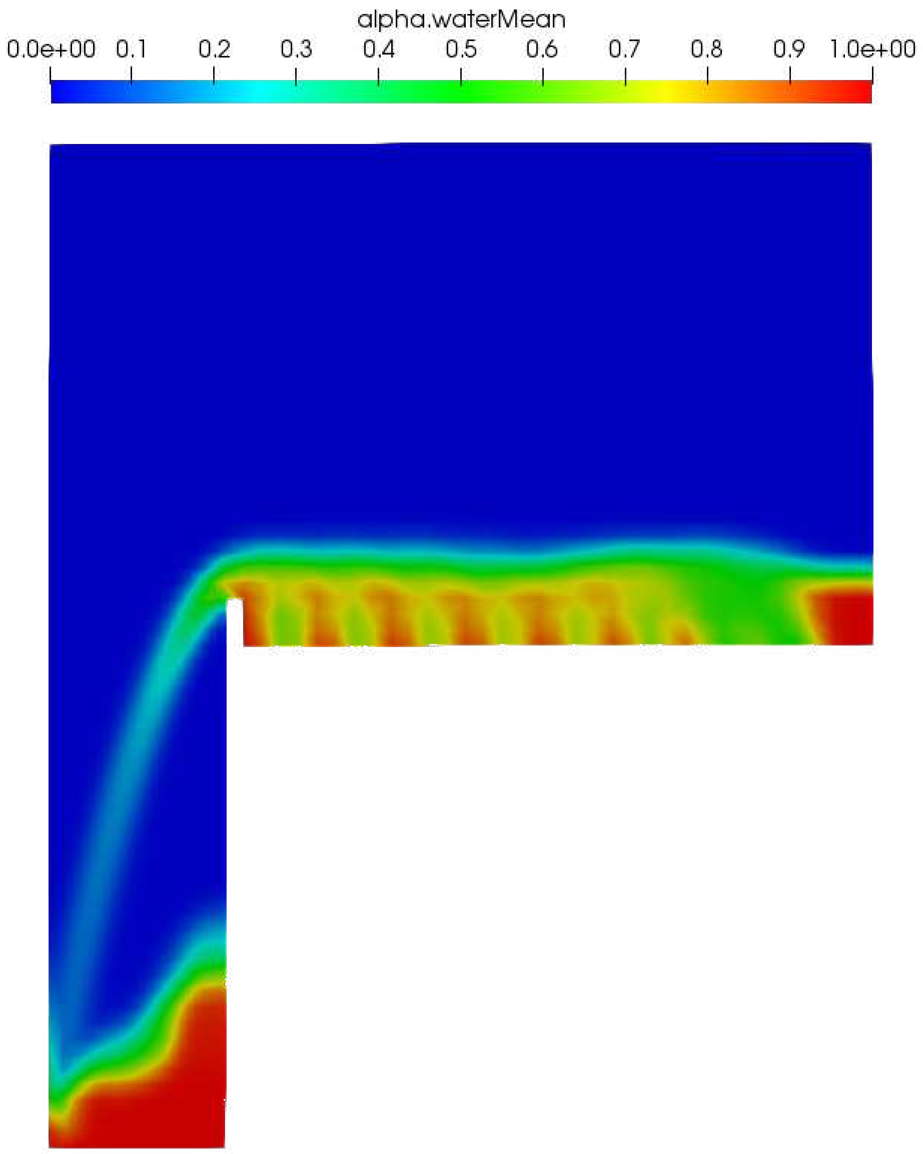


Figure 5.4: Liquid phase fraction profile on the YZ plane 0.01 m from the tray center for an commercial scale sieve tray simulation at quasi-steady-state with the superficial gas velocity of $0.421\,\mathrm{m\,s^{-1}}$, the liquid load of $17.8 \times 10^{-3}\,\mathrm{m^3\,s^{-1}}$. Averaged over time range from $t = 0\,\mathrm{s}$ to $t = 60\,\mathrm{s}$.

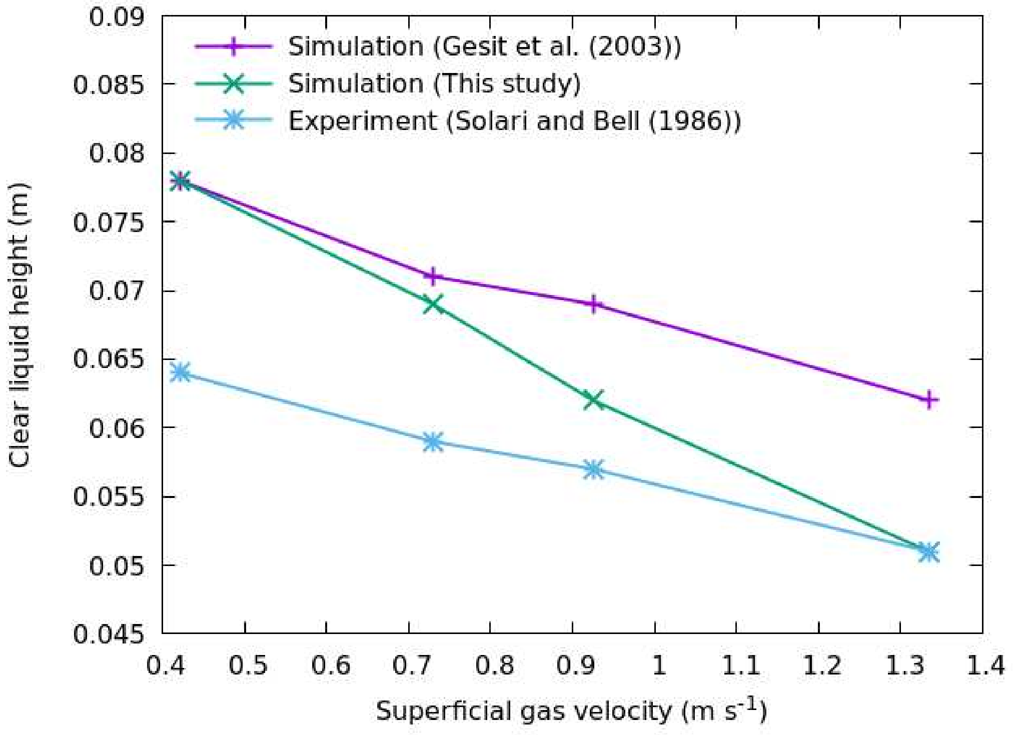


Figure 5.5: Clear liquid height as a function of superficial gas velocity for the commercial-scale sieve tray with the liquid load of $17.8 \times 10^{-3}\,\mathrm{m}^3\,\mathrm{s}^{-1}$.

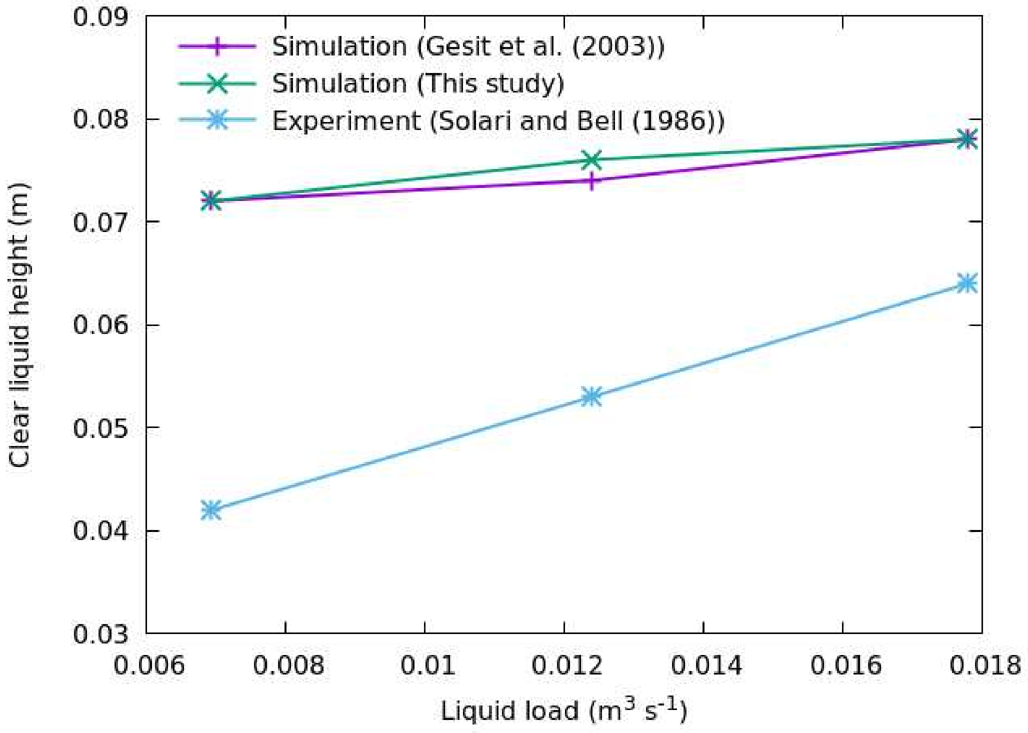


Figure 5.6: Clear liquid height as a function of liquid load for the commercial-scale sieve tray with the superficial gas velocity of $0.421\,\mathrm{m\,s^{-1}}$.

The results yielded a velocity distribution, namely the upstream profile. In the same manner, a downstream profile was determined between the line composed of probes 9 to 12 and the line composed of probes 13 to 16.

In the CFD work of Gesit et. al. [10], the upstream velocity profile was determined by integrating the liquid velocity component in the primary flow direction—i.e, the x direction in Figure 5.1—over the plane region between the line composed of probes 5 to 8 and the line composed of probes 9 to 12. The downstream velocity profile determined in the same manner over the region between the line composed of probes 9 to 12 and the line composed of probes 13 to 16.

The same method was adopted in the present study as well, the results were compared against the reference CFD simulations [10] as well as experimental measurements [45], as shown in Figure 5.8 and Figure 5.9. The predictions are in good agreement with the CFD simulation of Gesit et al. [10] and the experimental data. It was observed that at locations close to the wall of the tray, where the dimensionless coordinate in the transverse direction is approaching to 1, the predicted x-component of liquid velocity was negative, indicating the existence of reverse flow.

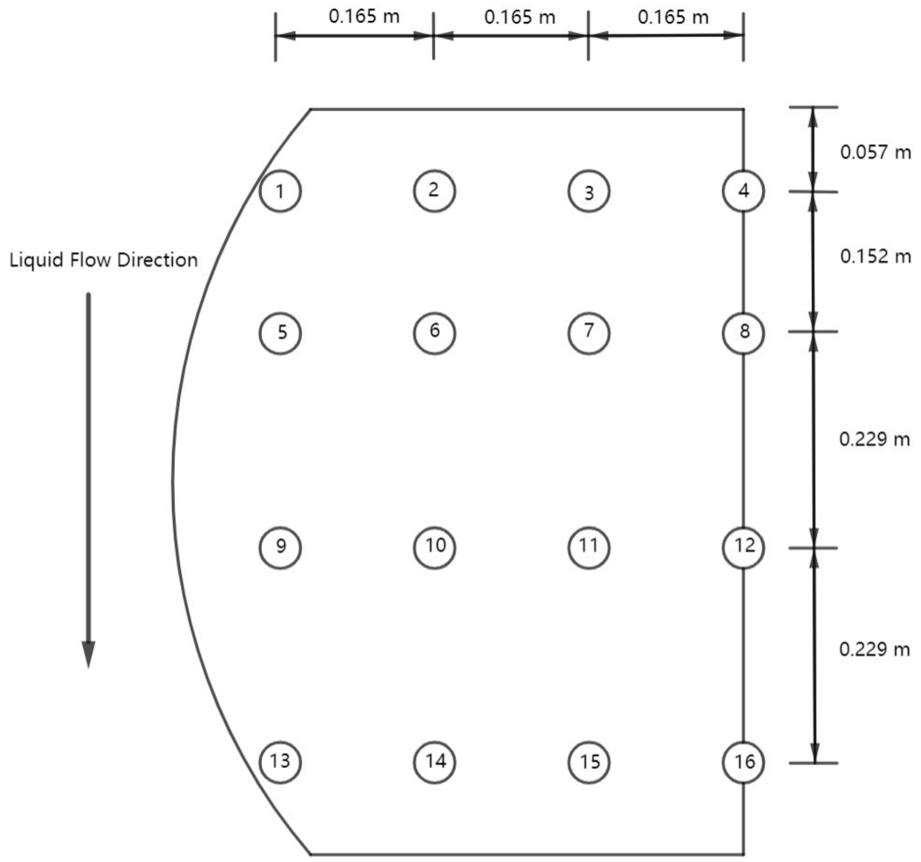


Figure 5.7: Probe positions in the experimental work of of Solari and Bell [45].

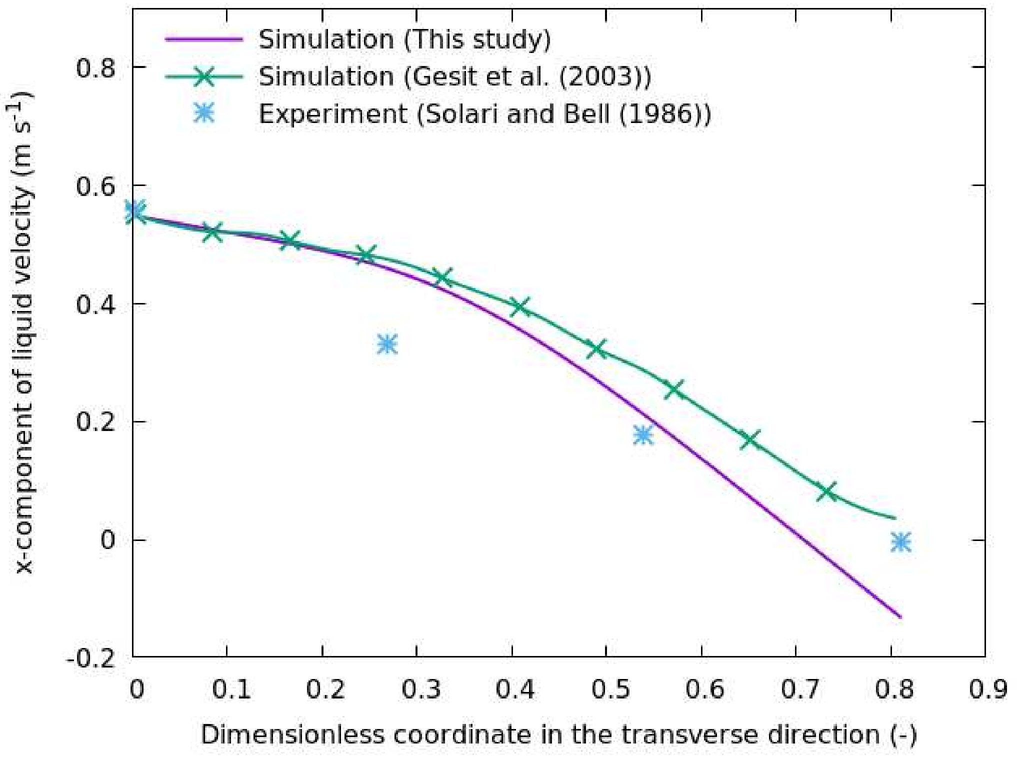


Figure 5.8: Upstream liquid velocity profile for a commercial-scale sieve tray simulation at quasi-steady-state with the superficial gas velocity of $0.421\,\mathrm{m\,s^{-1}}$, the liquid load of $17.8 \times 10^{-3}\,\mathrm{m^3\,s^{-1}}$.

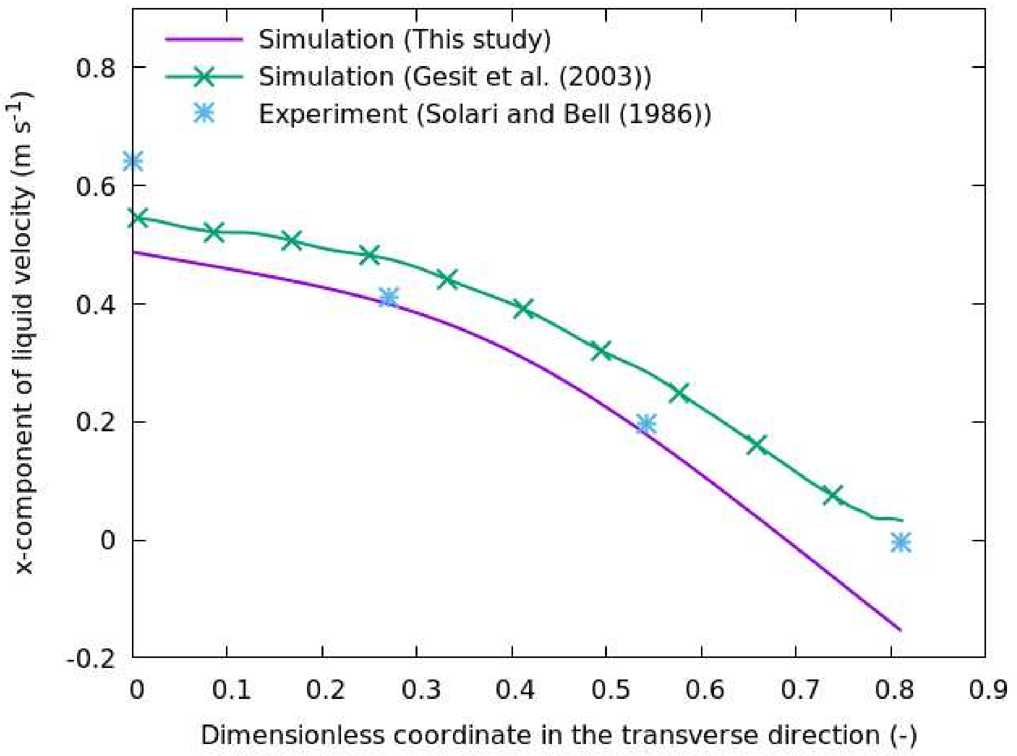


Figure 5.9: Downstream liquid velocity profile for a commercial-scale sieve tray simulation at quasi-steady-state with the superficial gas velocity of $0.421\,\mathrm{m\,s^{-1}}$, the liquid load of $17.8 \times 10^{-3}\,\mathrm{m^3\,s^{-1}}$.

Table 5.4: Specifications of discretization schemes for the commercial-scale tray simulations.

Term	Scheme
Time	`Euler`
Gradient	`cellMDLimited leastSquares 0.5`

Divergence	`Gauss limitedLinear01 1`
	`Gauss vanLeer`
	`Gauss limitedLinearV 1`
	`Gauss limitedLinear 1`
	`Gauss linear`

Laplacian	`Gauss linear limited corrected 0.333`
Interpolation	`linear`
Surface normal gradient	`limited corrected 0.333`

[*] These schemes apply to equation terms for both phases—e.g. `div(phi.air,U.air)` and `div(phi.water,U.water)`—the suffixes are omitted for simplicity.

Chapter 6

Conclusion and Future Work

In this study, a method to perform CFD simulations of sieve tray hydrodynamics using the software package OpenFOAM® was developed and systematically documented.

The work started with constructing the governing equations that describe the physics of the gas-liquid mixture on a sieve tray. An Euler-Euler two-fluid model was selected to treat both phases as interpenetrating continua sharing the same pressure field. A correlation for drag force established by Krishna et al. [23] was employed as the closure model for the interphase momentum transfer term. The mixture $k-\epsilon$ model developed by Behzadi [2] was selected as the closure model for the Reynolds stress term.

The governing equations were solved using OpenFOAM® version 6, with the interphase momentum transfer closure model implemented as a user-defined model. The method was tested and optimized for sieve tray simulations using a series of test cases with a quasi 2D sieve tray geometry to reduce the computational cost. A set of boundary conditions was selected that successfully handles the critical phase fraction-velocity-pressure coupling relationship on the outlet boundary, where the flow conditions are normally not well characterized. It has been shown that the implemented mixture $k-\epsilon$ model [2] eliminates the numerical instability induced by phase inversion. Furthermore, it was proved that the modification applied to the drag model [22] is essential because it provides better predictions on drag force calculation and improves the stability of the simulation, as claimed by Krishna et al. [22]. It has been shown that the instantaneous flow variables oscillate in a chaotic but bounded manner and time-averaged properties converge to a quasi steady state eventually.

The developed method was applied to an experimental-scale tray simulations. The predicted clear liquid heights at various operating conditions were compared with reference CFD simulations and experimental data [22], and good quantitative agreements were observed. Simulations were conducted on an industrial-scale sieve tray geometry as well. It was shown that a fixed time step, instead of varying time steps limited by the Courant number, provides more stable simulations. Proper discretization schemes were selected to ensure the boundedness of the flow variables and the overall stability of simulations. Macroscopic results (i.e., clear liquid heights) as well as localized results (i.e., velocity profiles) were computed and validated against the results of CFD simulations [10] and experimental data [45], and good quantitative agreement was observed. The results of this work show that the developed OpenFOAM® methodology is capable of predicting sieve tray hydrodynamics and can be used as a tool for analysis and design.

Future studies could be conducted on the following areas:

- Incorporate interphase mass transfer model to study the tray efficiency.
- Investigate the effect of other interfacial forces, e.g, lift force, virtual mass force.
- Implement and explore new discretization schemes and linear solvers.

Bibliography

[1] Ansys, I. *ANSYS FLUENT 12.0/12.1 Documentation.*

[2] Behzadi, A., Issa, R., and Rusche, H. "Modelling of dispersed bubble and droplet flow at high phase fractions". In: *Chemical Engineering Science* 59.4 (2004), pp. 759–770.

[3] Bell, R. L. "Experimental determination of residence time distributions on commercial scale distillation trays using a fiber optic technique". In: *AIChE Journal* 18.3 (1972), pp. 491–497.

[4] Bell, R. L. "Residence time and fluid mixing on commercial scale sieve trays". In: *AIChE Journal* 18.3 (1972), pp. 498–505.

[5] Bennett, D., Agrawal, R., and Cook, P. "New pressure drop correlation for sieve tray distillation columns". In: *AIChE Journal* 29.3 (1983), pp. 434–442.

[6] Boussinesq, J. "Essai sur la théorie des eaux courantes. Mémoires présentés par divers savants a l'Academie des Sciences de l'Institute National de France". In: *XXIII (1)* (1877).

[7] Danckwerts, P. "Significance of liquid-film coefficients in gas absorption". In: *Industrial & Engineering Chemistry* 43.6 (1951), pp. 1460–1467.

[8] Dopazo, C. "On conditioned averages for intermittent turbulent flows". In: *Journal of Fluid Mechanics* 81.3 (1977), pp. 433–438.

[9] Garnier, C., Lance, M., and Marié, J. "Measurement of local flow characteristics in buoyancy-driven bubbly flow at high void fraction". In: *Experimental Thermal and Fluid Science* 26.6-7 (2002), pp. 811–815.

[10] Gesit, G., Nandakumar, K., and Chuang, K. T. "CFD modeling of flow patterns and hydraulics of commercial-scale sieve trays". In: *AIChE journal* 49.4 (2003), pp. 910–924.

[11] Górak, A. and Sorensen, E. *Distillation: fundamentals and principles.* Academic Press, 2014.

[12] Gosman, A. et al. “Multidimensional modeling of turbulent two-phase flows in stirred vessels”. In: *AIChE Journal* 38.12 (1992), pp. 1946–1956.

[13] Greenshields, C. *OpenFOAM v6 User Guide*. The OpenFOAM Foundation, 2018. URL: `https://doc.cfd.direct/openfoam/user-guide-v6`.

[14] Hanjalić, K. and Launder, B. E. “A Reynolds stress model of turbulence and its application to thin shear flows”. In: *Journal of fluid Mechanics* 52.4 (1972), pp. 609–638.

[15] Higbie, R. “The Rate of Absorption of a Pure Gas into a Still Liquid during Short Periods of Exposure”. In: *Trans. Am. Inst. Chem. Eng.* 31 (1935), pp. 365–377. URL: `https://cir.nii.ac.jp/crid/1573387450495969280`.

[16] Hill, D. P. “The computer simulation of dispersed two-phase flow”. PhD thesis. Imperial College London (University of London), 1998.

[17] Holzinger, G. *OpenFOAM A little User-Manual.* `https://www.researchgate.net/profile/Gerhard-Holzinger/publication/340174689_OpenFoam_-_a_little_user_manual/links/5e7c75aea6fdcc139c046bcb/OpenFoam-a-little-user-manual.pdf`, Last accessed on 2020-11-25.

[18] Issa, R. *A simple model for C_t*. private Communication. 1992.

[19] Issa, R. I. “Solution of the implicitly discretised fluid flow equations by operator-splitting”. In: *Journal of computational physics* 62.1 (1986), pp. 40–65.

[20] Jasak, H. “Finite volume discretisation with polyhedral cell support”. In: *Predavanja, NUMAPFOAM Summer School, Fakultet strojarstva i brodogradnje, Sveučilište u Zagrebu, Rujan* (2009).

[21] Jiang, B. et al. “Hydrodynamics and mass-transfer analysis of a distillation ripple tray by computational fluid dynamics simulation”. In: *Industrial & Engineering Chemistry Research* 52.49 (2013), pp. 17618–17626.

[22] Krishna, R. et al. “CFD simulations of sieve tray hydrodynamics”. In: *Chemical Engineering Research and Design* 77.7 (1999), pp. 639–646.

[23] Krishna, R. et al. “Rise velocity of a swarm of large gas bubbles in liquids”. In: *Chemical Engineering Science* 54.2 (1999), pp. 171–183.

[24] Krishna, R. and Van Baten, J. "Modelling sieve tray hydraulics using computational fluid dynamics". In: *Chemical Engineering Research and Design* 81.1 (2003), pp. 27–38.

[25] Lahey Jr, R. T. "The simulation of multidimensional multiphase flows". In: *Nuclear Engineering and Design* 235.10-12 (2005), pp. 1043–1060.

[26] Launder, B. E. and Spalding, D. B. "The numerical computation of turbulent flows". In: *Numerical prediction of flow, heat transfer, turbulence and combustion*. Elsevier, 1983, pp. 96–116.

[27] Lewis, W. K. and Whitman, W. G. "Principles of gas absorption." In: *Industrial & Engineering Chemistry* 16.12 (1924), pp. 1215–1220.

[28] Li, D. and Christian, H. "Simulation of bubbly flows with special numerical treatments of the semi-conservative and fully conservative two-fluid model". In: *Chemical Engineering Science* 174 (2017), pp. 25–39.

[29] Li, X. et al. "Computational fluid dynamics modeling of hydrodynamics of a new type of fixed valve tray". In: *Industrial & Engineering Chemistry Research* 53.1 (2014), pp. 379–389.

[30] Liu, C. et al. "A fluid–dynamic model for flow pattern on a distillation tray". In: *Chemical Engineering Science* 55.12 (2000), pp. 2287–2294.

[31] Lucas, D., Krepper, E., and Prasser, H.-M. "Use of models for lift, wall and turbulent dispersion forces acting on bubbles for poly-disperse flows". In: *Chemical Engineering Science* 62.15 (2007), pp. 4146–4157.

[32] Magnaudet, J. "The forces acting on bubbles and rigid particles". In: *ASME Fluids Engineering Division Summer Meeting, FEDSM*. Vol. 97. 1997, pp. 22–26.

[33] Mehta, B., Chuang, K., and Nandakumar, K. "Model for liquid phase flow on sieve trays". In: *Chemical Engineering Research and Design* 76.7 (1998), pp. 843–848.

[34] Menter, F. R. "Two-equation eddy-viscosity turbulence models for engineering applications". In: *AIAA journal* 32.8 (1994), pp. 1598–1605.

[35] Montoya, G. et al. “A review on mechanisms and models for the churn-turbulent flow regime”. In: *Chemical Engineering Science* 141 (2016), pp. 86–103.

[36] Moukalled, F., Mangani, L., and Darwish, M. “The Finite Volume Method”. In: *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab*. Cham: Springer International Publishing, 2016, pp. 103–135.

[37] Nernst, W. “Theorie der Reaktionsgeschwindigkeit in heterogenen Systemen”. In: *Zeitschrift für physikalische Chemie* 47.1 (1904), pp. 52–55.

[38] Norouzi, H. *Mastering twoPhaseEulerFoam Two: Extending the solver.* `https://www.cemf.ir/PDFs/OpenFOAM/extending-twoPhaseEulerFoam.pdf`, Last accessed on 2019-11-25.

[39] Paladino, E. E. and Maliska, C. R. “Virtual mass in accelerated bubbly flows”. In: *Proceedings of 4th European Themal Sciences, 29th-31st March, National Exhibition Centre, Birmingham, UK* (2004).

[40] Patankar, S. V. and Spalding, D. B. “A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows”. In: *Numerical prediction of flow, heat transfer, turbulence and combustion*. Elsevier, 1983, pp. 54–73.

[41] Rusche, H. “Computational fluid dynamics of dispersed two-phase flows at high phase fractions”. PhD thesis. University of London, 2002.

[42] Rzehak, R. and Krepper, E. “Closure models for turbulent bubbly flows: a CFD study”. In: *Nuclear Engineering and Design* 265 (2013), pp. 701–711.

[43] Schiller L. Naumann, Z. “A drag coefficient correlation”. In: *Zeitschrift des Vereins Deutscher Ingenieure* 77 (1935), pp. 318–320.

[44] Schubert, M. et al. “An imaging technique for characterization of fluid flow pattern on industrial-scale column sieve trays”. In: *Chemical Engineering Research and Design* 111 (2016), pp. 138–146.

[45] Solari, R. and Bell, R. “Fluid flow patterns and velocity distribution on commercial-scale sieve trays”. In: *AIChE journal* 32.4 (1986), pp. 640–649.

[46] Stichlmair, J. and Ulbrich, S. “Liquid channelling on trays and its effect on plate efficiency”. In: *Chemical engineering & technology* 10.1 (1987), pp. 33–37.

[47] Sun, J. et al. “Computational Fluid Dynamics Hydrodynamic Analysis of a Cross-Orthogonal Fixed-Valve Tray”. In: *Chemical Engineering & Technology* 37.3 (2014), pp. 383–391.

[48] Tomiyama, A. et al. “Drag coefficients of bubbles. 2nd Report. Drag coefficient for a swarm of bubbles and its applicability to transient flow; Kiho no koryoku keisu ni kansuru kenkyu. 2. Kihogun no koryoku keisu to hiteijoryu eno tekiyo-sei”. In: (1995).

[49] Van Baten, J., Ellenberger, J., and Krishna, R. “Hydrodynamics of reactive distillation tray column with catalyst containing envelopes: experiments vs. CFD simulations”. In: *Catalysis Today* 66.2-4 (2001), pp. 233–240.

[50] Wilcox, D. C. et al. *Turbulence modeling for CFD*. Vol. 2. DCW industries La Canada, CA, 1998.

[51] Zarei, T., Rahimi, R., and Zivdar, M. “Computational fluid dynamic simulation of MVG tray hydraulics”. In: *Korean Journal of Chemical Engineering* 26.5 (2009), pp. 1213–1219.

[52] Zuiderweg, F. “Sieve trays: a view on the state of the art”. In: *Chemical Engineering Science* 37.10 (1982), pp. 1441–1464.

Appendix A

Description of Finite Volume Discretization Schemes in OpenFOAM

In OpenFOAM®, the governing partial differential equations are discretized in space using a finite volume method (FVM) for collocated arbitrary polyhedral grids [20]. A typical control volume V_Q with cell center Q, as represented in Figure A.1, has an arbitrary number of faces that are shared with neighbouring control volumes. For example, the face f is the common face shared with a neighbour control volume with cell center N. $\mathbf{s}_f$ represents the normal vector of surface f, which has a magnitude equal to the face surface area, and $\mathbf{d}_f$ represents the distance vector from cell center Q to N.

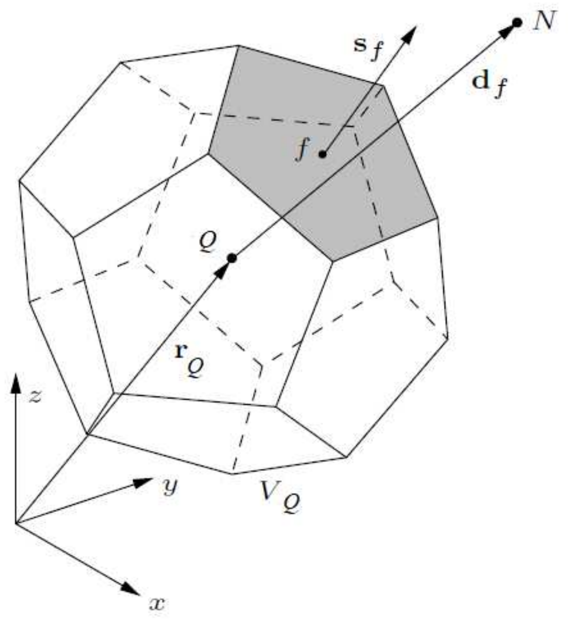


Figure A.1: A polyhedral control volume. Source: [20].

The definition of the cell center is given by the following equation:

$$\int_{\mathbf{V_Q}} (\mathbf{x} - \mathbf{x}_Q)\, \mathrm{d}V = 0 \tag{A.1}$$

where $\mathbf{x}$ denotes a location inside the control volume, and $\mathbf{x}_Q$ is the location of the control volume center. The value of an arbitrary quantity ϕ at any given location inside the control volume can be obtained using the following expression:

$$\phi(\mathbf{x}) = \phi_Q + (\mathbf{x} - \mathbf{x}_Q) \cdot (\nabla\phi)_Q \tag{A.2}$$

Integrating $\phi(\mathbf{x})$ expressed in equation (A.2) over the volume of V_Q yields the following equation:

$$\begin{aligned} \int_{V_Q} \phi(\mathbf{x})\mathrm{d}V &= \int_{V_Q} [\phi_Q + (\mathbf{x} - \mathbf{x}_Q) \cdot (\nabla\phi)_Q]\, \mathrm{d}V \\ &= \int_{V_Q} \phi_Q \mathrm{d}V + (\nabla\phi)_Q \cdot \int_{V_Q} (\mathbf{x} - \mathbf{x}_Q)\mathrm{d}V \\ &= \phi_Q V_Q \end{aligned} \tag{A.3}$$

Equation (A.3) indicates that the space dependent values of a variable $\phi(\mathbf{x})$ can be stored at the center of the control volume V_Q as a constant value ϕ_Q. Therefore, any properties of interest inside a given control volume can be stored as constant profiles at the center of the control volume.

Neighbouring control volumes exchange information through the common faces between each other. To obtain information of properties at the surface, the Gauss theorem is employed to convert the volume integral of the divergence of a vector field into the surface integral of that vector field over the closed surface of a control volume. This conversion relationship is given by the following expression:

$$\int_{V_Q} (\nabla \cdot \phi)\mathrm{d}V = \sum_f [\mathbf{s}_f \cdot (\phi)_f] \tag{A.4}$$

Recall the general conservation equation of an arbitrary quantity ϕ:

$$\underbrace{\frac{\partial(\rho\phi)}{\partial t}}_{\text{accumulation term}} = - \underbrace{\nabla \cdot (\rho\mathbf{v}\phi)}_{\text{convection term}} + \underbrace{\nabla \cdot (\Gamma\nabla\phi)}_{\text{diffusion term}} + \underbrace{S_\phi}_{\text{source term}} \tag{2.1 revisited}$$

This equation can be integrated over a control volume as follows:

$$\int_{V_Q} \frac{\partial(\rho\phi)}{\partial t}\mathrm{d}V = -\int_{V_Q} \nabla \cdot (\rho\mathbf{v}\phi)\mathrm{d}V + \int_{V_Q} \nabla \cdot (\Gamma\nabla\phi)\mathrm{d}V + \int_{V_Q} S_\phi(\phi)\mathrm{d}V \tag{A.5}$$

The Gauss theorem—i.e., equation (A.4)—can be applied to the convection and diffusion terms:

$$\int_{V_Q} \frac{\partial(\rho\phi)}{\partial t}\mathrm{d}V = -\sum_f [\mathbf{s}_f \cdot (\rho\mathbf{v}\phi)_f] + \sum_f [\mathbf{s}_f \cdot (\Gamma\nabla\phi)_f] + \int_{V_Q} S_\phi(\phi)\mathrm{d}V \tag{A.6}$$

The next step is to convert the integral form PDE into a set of linear algebraic equations by using suitable linearization schemes on each terms. Using equation (A.6) as an example, the schemes used to treat each term and the results are demonstrated below:

- Accumulation term: Using the Euler scheme gives the following expression.

  $$\int_{V_Q} \frac{\partial(\rho\phi)}{\partial t}\mathrm{d}V = \frac{(\rho_Q\phi_Q)^n - (\rho_Q\phi_Q)^o}{\Delta t}V_Q \tag{A.7}$$

  where superscript ^{o} and ^{n} denote the variable values at the old time and new time steps, respectively. For the rest of the terms of equation (A.6) the linearization is applied to the new time step value only (i.e., using an implicit time integration), so superscripts will be omitted for simplicity.

- Convection term: Using the central differencing scheme gives the following equation.

  $$\sum_f [(\rho\mathbf{v})_f \mathbf{s}_f \cdot \phi_f] = \sum_f [f_x \mathbf{S}_f \cdot (\rho\mathbf{v})_Q]\phi_Q + \sum_f [(1 - f_x)(\mathbf{S}_f \cdot (\rho\mathbf{v})_N \phi_N] \tag{A.8}$$

  where $f_x = \overline{fN}/\overline{QN}$ represents the ratio of the distance between the face center f and cell center N to the distance between cell center Q and cell center N. It should be noted that equation (A.8) needs to be summed over all control volumes that share a common face with cell V_Q, and the value of f_x also needs to be evaluated based on the individual distance between cell centers.

- Diffusion term: When the grid is orthogonal, $\mathbf{d}_f$ is parallel to $\mathbf{s}_f$ and the gradient term can be discretized to give the following expression:

  $$\sum_f [\Gamma_f \mathbf{s}_f \cdot (\nabla\phi)_f] = \sum_f \Gamma_f |\mathbf{s}_f| \frac{\phi_N - \phi_Q}{|\mathbf{d}_f|} \tag{A.9}$$

  In case of non-orthogonal grid, an explicit correction term needs to be introduced [20].

- Source Term: When the source term is a non-linear function of the unknown ϕ, it is not possible to obtain discretized linear algebraic equations directly. To solve this problem, a widely adopted method is to locally linearize the source term first [20]:
$$S_\phi(\phi) = Su(\phi) + Sp(\phi)\phi \tag{A.10}$$
It should be noted that $Su()$ and $Sp()$ may or may not be ϕ dependent [20]. Therefore, the discretized source term can be written as follows:
$$\int_{\mathbf{V_P}} S_\phi(\phi)\mathrm{d}V = SuV_Q + SpV_Q\phi_Q \tag{A.11}$$

Combining like terms in the discretized linear algebraic equations, (A.7) to (A.11), gives the following expression:

$$a_Q\phi_Q + \sum_f a_N\phi_N = SuV_Q + \frac{(\rho_Q\phi_Q)^o V_Q}{\Delta t} \tag{A.12}$$

where a_Q and a_N are simply the summation of coefficients belonging to variables ϕ_Q and ϕ_N, respectively.

Equation (A.12) is the linearized result of equation (A.6) and can be solved algebraically. If this discretization process is performed on governing PDEs over each control volume in the grid, the result is a system of linear algebraic equations that can be represented as a large-scale sparse matrix:

$$[\mathbf{A}][\boldsymbol{\phi}] = [\mathbf{b}] \tag{A.13}$$

where $[\boldsymbol{\phi}]$ and $[\mathbf{b}]$ are the column vectors of unknowns and sources, respectively, and $[\mathbf{A}]$ is the square coefficient matrix.

During the process of discretization in OpenFOAM® each term of the PDEs is individually evaluated through functions grouped into two namespaces [13]:

- `fvm` (Finite Volume Method): Contains functions that evaluate terms of PDEs implicitly based on unknown values and return an `fvMatrix<Type>` (i.e., modification of $[\mathbf{A}]$ in equation (A.13))

- `fvc` (Finite Volume Calculus): Contains functions that evaluate terms of PDEs explicitly based on known values and return a `geomertricField<Type>` (i.e., modification of $[\mathbf{b}]$ in equation (A.13)).

```
    (
        fvm::ddt(rho,phi)
      + fvm::div(rho,U,phi)
      - fvm::laplacian(Gamma,phi)
     ==
      + fvm::SuSp(S,phi)
    );

```

Figure A.2: A source code that represents (2.6) in OpenFOAM® syntax.

Some of the most commonly used functions for PDE discretization in OpenFOAM® are summarized in Table A.1.

The syntax in OpenFOAM® is designed to resemble its corresponding mathematical form. This provides a convenient way to read, modify, or create new source code. For example, an OpenFOAM® style source code that represents equation (2.6) is presented in Figure A.2

Table A.1: Common functions in namespace `fvm` and `fvc` in OpenFOAM®.

PDE term description	Implicit/Explicit	Math expression	`fvm::/fvc::` functions
Time derivative	Implicit/Explicit	$\partial(\rho\phi)/\partial t$	`ddt(rho,phi)`
Laplacian	Implicit/Explicit	$\nabla\cdot(\Gamma\nabla\phi)$	`laplacian(Gamma,phi)`
Convection	Implicit/Explicit	$\nabla\cdot(\psi\phi)$	`div(psi,phi)`
Divergence	Explicit	$\nabla\cdot\chi$	`div(chi)`
Gradient	Explicit	$\nabla\cdot\chi$	`grad(chi)`
Source	Implicit	$\rho\phi$	`Sp(rho,phi)`
	Explicit		`Su(rho,phi)`
	Implicit/Explicit		`SuSp(rho,phi)`

[1] ρ: scalar, volScalarField; ϕ: vol<Type>Field; Γ: scalar, surfaceScalarField, volTensorField, surfaceTensorField; ψ: surfaceScalarField; χ: surface<Type>Field, vol<Type>Field.

[2] `fvm::SuSp` performs implicit discretization if the sign of ρ is negative, otherwise the source is discretized explicitly.

[3] Source: [13]

Appendix B

Description of Mixture $k - \epsilon$ Model in OpenFOAM

In OpenFOAM® version 6, a built-in turbulence model named `mixtureKEpsilon` is provided. The source code can be found in the `OpenFOAM-6/src/TurbulenceModels/phaseCompressible/RAS/mixtureKEpsilon` directory. The source code is not exactly the same as the model provided by Behzadi [2] in Section 2.3. Therefore, the differences and important modifications are discussed in this section.

First, the density of the dispersed phase is correlated with virtual mass taken into account. The effective density `rhogEff` of the dispersed phase is calculated using the method shown in line 16 of Figure B.1. This correlation is usually employed to improve numerical stability in simulations where there is significant density difference and relative acceleration between phases [39]. Based on the definition of virtual mass introduced in Section 2.2, this modification does not need to be applied to the continuous phase, as shown on line 6 of Figure B.1, because the returned continuous phase effective density `rholEff` is unchanged from the regular definition. In this study, the only interphase force being modeled is the drag force only and no virtual mass model has been employed.

With C_{vm} being set to be zero, the effective densities in the `mixtureKEpsilon` model are essentially the same as the regular densities, and the other mixture properties are calculated in the same manner as displayed in equations (2.36) to (2.41).

```
template<class BasicTurbulenceModel>
tmp<volScalarField> mixtureKEpsilon<BasicTurbulenceModel>::rholEff() const
{
const transportModel& gas = this->transport();
const twoPhaseSystem& fluid = refCast<const twoPhaseSystem>(gas.fluid());
return fluid.otherPhase(gas).rho();
}

template<class BasicTurbulenceModel>
tmp<volScalarField> mixtureKEpsilon<BasicTurbulenceModel>::rhogEff() const
{
    const transportModel& gas = this->transport();
    const twoPhaseSystem& fluid = refCast<const twoPhaseSystem>(gas.fluid())
;
    const virtualMassModel& virtualMass =
        fluid.lookupSubModel<virtualMassModel>(gas, fluid.otherPhase(gas));
    return gas.rho() + virtualMass.Cvm()*fluid.otherPhase(gas).rho();
}

```

Figure B.1: The definition of effective density in `mixtureKEpsilon.C`.

The source code used to evaluate the turbulence response coefficient C_t is displayed in Figure B.2.

In Figure B.2, line 21 corresponds to equation (2.46), line 22 corresponds to equation (2.45), and the code between line 15 to 20 correspond to equations (2.47) to (2.50) with the modification that equation (2.48) has been replaced with equation (2.67) to prevent singularity problems and to account the swarm correlation as well. It should be mentioned that ${C_t}^2$ is returned by the code instead of C_t simply because ${C_t}^2$ is the only form used when the value of C_t is called in (2.38) to (2.42).

The source code for the turbulent kinetic energy transport equation and related codes of the `mixtureKEpsilion` model are shown in Figure B.3.

The code related to turbulent kinetic energy equation in Figure B.3 can be translated into the following equation:

$$\begin{aligned}\frac{\partial k_m}{\partial t} &+ \nabla \cdot (\Phi_m k_m) - k_m \nabla \cdot (\Phi_m) - \nabla^2 \left(D_{k,eff}(\nu_{t,m}) k_m\right) \\ &= G_m - \left(\frac{2}{3} k_m \nabla \cdot \mathbf{U}_m\right) - \left(\frac{\epsilon_m}{k_m} k_m\right) + S_{km} + S_{\mathrm{fvOptions},k_m} \quad \text{(B.1)}\end{aligned}$$

The differences between equation (B.1) and the original formula presented in equation (2.34) are summarized as follows:

```
template<class BasicTurbulenceModel>
tmp<volScalarField> mixtureKEpsilon<BasicTurbulenceModel>::Ct2() const
{
    const mixtureKEpsilon<BasicTurbulenceModel>& liquidTurbulence =
        this->liquidTurbulence();

    const transportModel& gas = this->transport();
    const twoPhaseSystem& fluid = refCast<const twoPhaseSystem>(gas.fluid())
;
    const transportModel& liquid = fluid.otherPhase(gas);

    const volScalarField& alphag = this->alpha_;

    volScalarField magUr(mag(liquidTurbulence.U() - this->U()));

    volScalarField beta
    (
        (6*this->Cmu_/(4*sqrt(3.0/2.0)))
       *fluid.Kd()/liquid.rho()
       *(liquidTurbulence.k_/liquidTurbulence.epsilon_)
    );
    volScalarField Ct0((3 + beta)/(1 + beta + 2*gas.rho()/liquid.rho()));
    volScalarField fAlphad((180 + (-4.71e3 + 4.26e4*alphag)*alphag)*alphag);

    return sqr(1 + (Ct0 - 1)*exp(-fAlphad));
}

```

Figure B.2: The definition of the turbulence response coefficient in `mixtureKEpsilon.C`.

```
    // Lahey model
    tmp<volScalarField> bubbleG
    (
        Cp_
       *liquid*liquid.rho()
       *(
            pow3(magUr)
          + pow(drag.CdRe()*liquid.nu()/gas.d(), 4.0/3.0)
           *pow(magUr, 5.0/3.0)
        )
       *gas
       /gas.d()
    );
    return bubbleG;
    }

    ...

    template<class BasicTurbulenceModel>
    tmp<fvScalarMatrix> mixtureKEpsilon<BasicTurbulenceModel>::kSource() const
    {
        return fvm::Su(bubbleG()/rhom_(), km_());
    }

    ...

    // Mixture flux
    surfaceScalarField phim("phim", mixFlux(phil, phig));

    ...

    // Turbulent kinetic energy equation
    tmp<fvScalarMatrix> kmEqn
    (
        fvm::ddt(km)
      + fvm::div(phim, km)
      - fvm::Sp(fvc::div(phim), km)
      - fvm::laplacian(DkEff(nutm), km)
     ==
        Gm
      - fvm::SuSp((2.0/3.0)*divUm, km)
      - fvm::Sp(epsilonm/km, km)
      + kSource()
      + fvOptions(km)
    );

    ...

    volScalarField Cc2(rhom/(alphal*rholEff() + alphag*rhogEff()*Ct2_()));
    kl = Cc2*km;
    kl.correctBoundaryConditions();
    epsilonl = Cc2*epsilonm;
    epsilonl.correctBoundaryConditions();
    liquidTurbulence.correctNut();

    Ct2_() = Ct2();
    kg = Ct2_()*kl;
    kg.correctBoundaryConditions();
```

- `rhom`: This term represents the effective density ρ_m (definition is given in Figure B.1) of the mixture. ρ_m is extracted from the coded turbulence transport equations for the mixture and reintroduced into the calculation during the evaluation of turbulence properties for individual phases. The relevant code is shown at lines 49 and 50 in Figure B.3. The likely reason for this treatment is because the significant density difference between the phases would introduce numerical difficulty if the equation were solved in terms of $\rho_m k_m$ instead of k_m.

- `phim`: This term represents volumetric mixture flux Φ_m and the definition is given at line 28 in Figure B.3. This term is equivalent to $\mathbf{U}_m$.

- `DkEff()`: This term represents a function $D_{k,eff}()$, which returns the value of effective diffusivity for k. The value of $D_{k,eff}(\nu_{t,m})$ is $((\mu_{t,m}/\mathbf{Sc}_{\epsilon,m})/\rho_m)$ with the compressibility correction part removed.

- `fvm::SuSp((2.0/3.0)*divUm, km)`: This term represents the compressibility correction term $((2/3)k_m\nabla\cdot\mathbf{U}_m)$, which is separated from the effective diffusivity term.

- `fvm::Sp(fvc::div(phim),km)`: This term, $k_m(\nabla\cdot\Phi_m)$, is introduced into the equation through the following process:

$$\mathbf{U}_m\cdot\nabla k_m = \nabla\cdot(\mathbf{U}_m k_m) - k_m\nabla\cdot\mathbf{U}_m \tag{B.2}$$

  This numerical treatment helps maintain boundedness of the solution of k_m and promotes convergence for steady-state simulation. For transient simulation using the `PIMPLE` algorithm, this treatment is also beneficial for intermediate iterations in the `SIMPLE` loop.

- `kSource()`: This term accounts for additional sources of k_m. More specifically, bubble-induced turbulence can be included, which is modeled using the model of Lahey [25]. The definition of the Lahey model is given at lines 2 to 15 in Figure B.3. In this study, this term is omitted by setting the coefficient Cp to be zero.

- `fvOptions(km)`: This term accounts for other sources for k_m that can be included by the user at run time. No additional sources were considered in this study.

The treatments for the turbulence kinetic energy dissipation rate equation are similar and not reported here for simplicity.