Numerical Simulation of Corium Jet Fragmentation

Yunus Eren Akin

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science

in

Nuclear Engineering

Yang Liu, Chair
Mark Pierson
Juliana Pacheco Duarte

April 11, 2022
Blacksburg, Virginia

Keywords: Jet breakup, MULES, Two-phase flow
Copyright 2022, Yunus Eren Akin
Numerical Simulation of Corium Jet Fragmentation

Yunus Eren Akin

(ABSTRACT)

Severe accident studies have become important for understanding the environmental impact of nuclear power following the Three Mile Island (TMI) accident. Severe accident phenomena are interdisciplinary since they can include the interaction of molten corium with reactor structures and water, and the transport and release of fission products that carry vapors and aerosols. During an accident, molten corium may fall in the form of a jet into a water pool. The fragmentation of the corium jet is a critical process concerning Fuel-Coolant Interactions (FCI). A steam explosion could occur depending on jet fragmentation and dispersion in the water. A 2-D in-house CFD MATLAB script has been developed by employing the Volume of Fluid Method (VOF) to simulate jet fragmentation in a water pool. This research is the first study to use the Multidimensional Universal Limiter for Explicit Solution (MULES) algorithm to capture the interface behavior in the jet breakup process. The effect of the compression constant, the mesh sensitivity, and the face correction loop on the MULES are investigated in this thesis. Moreover, the effects of the jet inlet velocity and jet diameter on the jet breakup length are numerically studied using the newly developed code. The simulation results compared well with the experimental results and another numerical study found in the literature.
Numerical Simulation of Corium Jet Fragmentation

Yunus Eren Akin

(GENERAL AUDIENCE ABSTRACT)

After the TMI accident, many countries have started experimental and numerical studies to investigate and predict the accident’s possible outcomes. During a severe accident, the molten corium may drop into a water pool in the form of a jet. The jet contacts the water, fragments into droplets and disperses in the pool, which could then cause a steam explosion. A possible molten corium-water interaction is simulated using a 2-D in-house CFD code written in MATLAB in this study. Interface tracking is one of the most critical processes in multi-phase flow simulation, and the MULES algorithm is used to capture the interface in this study. It is the first study to use the MULES algorithm to investigate jet breakup behavior. The simulation results compared well with experimental data and other numerical studies.
Dedication

To my lovely family and fiancee
Acknowledgments

Firstly, I would like to thank the Ministry of National Education of the Turkish Government for providing me with this scholarship I was granted. Thanks to this opportunity, I could have a chance to expand my ideas and views, not only my academic career. I also would like to thank my advisor Dr. Yang Liu for his continuous guidance and patience during my study. I have always felt him watching over me for academic research during my study. Also, I would like to thank my precious family and my dear fiancee for their limitless support, even from a long distance. Without their encouragement, I could not be in this position. Lastly, I would like to thank Virginia Tech for the welcoming environment that served me, which enabled me to focus on my study.
# Contents

List of Figures ix

List of Tables xii

1 Introduction 1

1.1 Severe Accident Progression 2

1.2 Fuel-Coolant Interaction 2

2 Review of Literature 6

2.1 Previous Studies 6

2.1.1 Experimental Studies 6

2.1.2 Numerical Studies 11

3 Governing Equations and Discretization 19

3.1 Single-Phase Flow 19

3.1.1 Continuity Equation 19

3.1.2 Momentum Equation 22

3.2 Multi-Phase Flow Equations 25

3.2.1 Surface Tension Force 26
3.2.2 Curvature Calculation ........................................... 28
3.3 The Volume of Fluid Method ...................................... 28
3.4 The Finite Volume Method ........................................ 31
   3.4.1 Discretization Process ....................................... 32
   3.4.2 Discretization of Continuity Equation ..................... 32
   3.4.3 Discretization of Momentum Equation ...................... 34
   3.4.4 Discretization of Surface Tension .......................... 37
   3.4.5 MULES Algorithm ........................................... 39
3.5 Rhie-Chow Interpolation ......................................... 44
3.6 The SIMPLE Algorithm .......................................... 46

4 Results and Discussion ............................................ 52
   4.1 Single-Phase Flow Results .................................... 52
      4.1.1 Steady-State Single-Phase Flow ....................... 52
      4.1.2 Unsteady Single-Phase Flow ............................ 55
      4.1.3 Effect of Initial Velocity ............................... 57
   4.2 MULES Algorithm Verification ................................. 59
      4.2.1 Effect of $C_\alpha$ on the MULES .................... 65
      4.2.2 Effect of the Mesh Size ............................... 67
      4.2.3 Effect of $\lambda$ on the MULES ....................... 69
4.3 Jet Fragmentation Simulation ........................................... 70

4.3.1 Effect of Jet Diameter ............................................ 72

4.3.2 Effect of Inlet Velocity ........................................... 76

5 Conclusion and Future Work ........................................... 80

Bibliography ................................................................. 83

Appendix A The MULES Algorithm MATLAB Code .............. 89

Appendix B The SIMPLE Algorithm MATLAB Code .............. 97
# List of Figures

1.1 Sketch of Severe Accident Scenario ................................................. 4  
2.1 MISTEE Facility Schematic ............................................................. 7  
2.2 Molten Wood’s metal melt jet fragmentation in a water pool ................. 8  
2.3 The jet breakup behaviour observed by Iwasawa et al. .......................... 10  
2.4 Jet fragmentation behavior observed by Matsuo et al. .......................... 11  
2.5 Jet breakup and fragmentation in a water pool .................................. 13  
2.6 Jet breakup and fragmentation in a water pool .................................. 15  
2.7 The jet breakup length with different We numbers ............................... 16  
2.8 Jet breakup process simulated by Zhou et al. ................................... 17  
3.1 Finite control volume fixed in space ................................................. 20  
3.2 Surface forces performing on the volume of control for $u$ velocity component ............................................... 23  
3.3 True interface .................................................................................. 29  
3.4 2D Control volumes and grid structure ............................................. 32  
3.5 Control volume and grid structure ..................................................... 33  
3.6 One-dimensional domain ................................................................... 46  
3.7 Flow chart of the developed code ..................................................... 51
4.1 Domain geometry and boundary conditions ....................... 53
4.2 Velocity distribution the developing velocity and analytical solution .......................... 54
4.3 Horizontal velocity distribution ........................................ 54
4.4 Pressure distribution ..................................................... 55
4.5 Different mesh sizes simulation’s velocity distribution .................. 56
4.6 Illustration of the channel ............................................... 57
4.7 Specific cell velocity changes with time .............................. 58
4.8 Specific cell velocity changes with time .............................. 58
4.9 (a) Domain geometry and the initial conditions (b) void fraction distribution and void fraction color bar ................................. 59
4.10 The flow of a bubble in the channel .................................... 60
4.11 Void fraction distribution ................................................. 61
4.12 The flow of a bubble in the channel .................................... 62
4.13 Void fraction distribution ................................................. 62
4.14 Void fraction distribution ................................................. 63
4.15 Void fraction distribution zoom-in viewed .............................. 63
4.16 A turning notched disk in a vortex pool .............................. 64
4.17 Void fraction distribution of the notched disk .......................... 65
4.18 Void fraction distribution of the notched disk .......................... 65
4.19 Comparison of compression constant ($C_\alpha$) ......................... 66
4.20 Total void fraction values’ variation with time .......................... 67
4.21 Mesh sensitivity ........................................................................ 68
4.22 Void fraction distribution ............................................................. 69
4.23 Total void fraction values’ variation with time ............................ 70
4.24 Domain geometry and boundary conditions ................................. 71
4.25 Jet breakup and fragmentation in the water pool (500x75 mm²) ...... 73
4.26 Velocity fields in the water pool (500x75 mm²) .............................. 74
4.27 Jet breakup length comparison ..................................................... 75
4.28 Jet breakup with different Weber numbers for $D_{jet} = 5$ mm ........ 77
4.29 Breakup length variation with respect to Weber number ............... 78
List of Tables

2.1 Summary of numerical and experimental studies in the literature . . . . . . . 18

3.1 Delta functions used in the literature . . . . . . . . . . . . . . . . . . . . . . 27

4.1 Mesh size comparison . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57

4.2 Comparison of the current study and numerical and experimental studies
found in the literature . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79
Nomenclature

Greek Letters

\( \alpha \)  Void Fraction

\( \delta_S \)  Delta Function

\( \kappa \)  Curvature

\( \mu \)  Dynamic Viscosity

\( \nu \)  Kinematic Viscosity

\( \rho \)  Density

\( \sigma \)  Normal Stress

\( \tau \)  Tangential Stress

\( \tilde{\alpha} \)  Smoothed Void Fraction

Symbols

\( \dot{m} \)  Mass Flux

\( \lambda \)  Weighting Factor

\( \vec{n} \)  Unit Normal

\( \vec{v} \)  Velocity

\( A \)  Anti-diffusive Flux
$a_x$ Horizontal Acceleration

$C_\alpha$ Compression Constant

$F$ Face Fluxes

$f_{\sigma}, F_{\text{surf}}$ Surface Tension Force

$g$ Gravity

$m$ Mass

$P$ Pressure

$Q$ Net Flux

$S$ Surface

$t$ Time

$u$ Horizontal Velocity

$u_r$ Compression Velocity

$V$ Volume

$v$ Vertical Velocity
Acronyms

BDBA  Beyond-the-Design-Basis-Accident.

CFD  Computational Fluid Dynamics.

CSF  Continuum Surface Force.

DCH  Direct Containment Heating.

DEA  Design Extension Accident.

FCI  Fuel-Coolant Interaction.

FCT  Flux Corrected Transport.

KTH  Royal Institute of Technology.

LS  Level-Set.

LWR  Light Water Reactor.

MCCI  Molten Corium Concrete Interaction.

MPS  Moving Particle Semi-implicit.

MULES  Multidimensional Universal Limiter for Explicit Solution.

PLIC  Piecewise Linear Interface Calculation.

RPV  Reactor Pressure Vessel.
SA  Severe Accident.

SIMPLE  Semi-Implicit Method for Pressure Linked Equations.

SLIC  Simple Line Interface Calculation.

TMI  Three Mile Island.

VOF  Volume Of Fluid.
Chapter 1

Introduction

Nuclear power plants play an important role in generating carbon-free electricity in the world. Many countries are currently building or are planning to build more nuclear power plants. Even though it is one of the best energy sources, it could bring some safety issues. Especially after the Three Mile Island (TMI) accident, nuclear authorities and industry took precautions, and they have started research about the accident’s causes and potential safety impact extensively.

The Three Mile Island (TMI-2) accident was a serious alarm to the nuclear regulatory authorities and the nuclear industry. Although there had been earnest efforts to avoid a core-melt accident, it is possible that this kind of accident may happen to reactors, as shown in the Fukushima Daiichi disaster. Another point is that a core-melt accident has a hazardous effect on the public and the Fukushima Daiichi and Chernobyl accidents demonstrated that clearly. For this reason, such accidents must be avoided at the design phase and be mitigated should they occur. A deep knowledge base on these kinds of accidents needs to be acquired for better design and safe operation of the nuclear power plants. After the TMI-2 accident, experimental and analytical research programs started in the United States and European countries. Initially, the core-melt accidents were named degraded core accidents. Now they are referred to as Severe Accidents (SA). Frequently they are called Beyond-the-Design-Basis-Accidents (BDBA) or just Design Extension Accidents (DEA). The most commonly accepted classification is a severe accident [1].
1.1 Severe Accident Progression

The phenomena associated with a severe accident are remarkably complicated since its main characteristics include the core melts’ interaction with the reactor structures and water, the deposition, transport, and release of the fission products that carry aerosols and vapor. The interaction of core melt may give rise to ablation of structures, steam explosion, concrete melting and gas generation, vessel failure, and spreading/dispersion of heat-generating core melt (debris). These phenomena are related to the disciplines of thermal hydraulics, high-temperature material interactions, high-temperature chemistry, and aerosol physics. Besides the integrity of the containment has an important role in severe accidents research. Containment failures can be divided into two groups: (1) those that may fail the containment early, and (2) those that may fail the containment much later. There are some energetic processes that may fail the containment early, which are hydrogen combustion-detonation, direct containment heating (DCH), steam explosion, and melt attack. Also, molten corium concrete interaction (MCCI) and the lack of melt coolability were determined as the processes that may fail the containment later [1].

1.2 Fuel-Coolant Interaction

In case of a severe accident in Light Water Reactors (LWR), a molten corium in the fuel core region could be in touch with the water pool. There are two possibilities: (1) the corium from the initial core region may fall into the water in the lower plenum (in-vessel case), (2) the corium may fall into the containment’s lower cavity (ex-vessel case). After melt-coolant interactions, a steam explosion may occur. In an in-vessel case, the core melt could move to the bottom of the reactor pressure vessel (RPV) in the form of small-scale jets, where
they will contact with water and may lead to a steam explosion. Moreover, the melt will accumulate at the bottom of the RPV and damage the vessel if it is not coolable [2]. The ex-vessel accident progression focuses on the process that could damage the containment. Hydrogen detonation, steam explosion, Direct Containment Heating (DCH), and melt attack on the BWR containment was determined as the loads which can lead to early containment failure [1]. In case of a severe accident of the LWRs, it is possible to have interaction between the reactor coolant and reactor core melt. This is called Fuel-Coolant Interactions and it may result in a steam explosion (FCI) [3].

Two main factors can decide the severity of the steam explosion. The first one is the quantity of the melt flowing into a water pool. The second one is the limited breakup of the molten corium poured through the water pool before the steam explosion [4]. After a jet enters the water pool, there is constantly stripping from the surface of the jet that gives rise to the thinning of the jet, and eventually, the coarse breakup of a coherent core could happen. The coarse breakup generates relatively larger droplets which could be pursued by a secondary breakup, namely, the bigger droplets break into smaller fragments [5]. Molten corium jet fragmentation, breakup modes, and jet breakup length have been studied by many investigators numerically and experimentally [2, 3, 6, 7]. A possible ex-vessel severe accident scenario is schematically shown in Figure 1.1.happening near the leading edge. Although many numerical and experimental studies were carried out for liquid-liquid systems, the phenomenon is not well understood because of its complexity. Intermixing of the jet and water makes it difficult to predict because a number of mechanisms could exist. The jet breakup/fragmentation mainly involves two mechanisms. (1) Sideways stripping gives rise to the thinning of the jet, and it causes a coarse breakup next to the leading edge, where the jet breaks into numerous small segments. (2) Coarse breakup could occur earlier for large diameters, even before enough the jet thinning, because of stripping [2].
In the present study, a two-dimensional in-house CFD code has been developed to simulate jet fragmentation behavior. The two liquid phases, namely, molten corium and water, are treated as immiscible fluids. Since the problem of interest is a multi-phase flow problem, the void fraction equation has to be solved in addition to the continuity and momentum equations. The energy equations have not been considered in this study. The Multidimensional Universal Limiter for Explicit Solution (MULES) algorithm, which has been used in other two-phase flow simulations, will be implemented to capture the interface behavior in the jet breakup process. The Finite Volume Method (FVM), one of the most commonly used numerical methods, is used to discretize the equations. Continuum Surface Force (CSF) model is used for the calculation of the surface tension force, which is another important factor in multi-phase flow. The semi-Implicit Method for Pressure Linked Equations (SIM-
1.2. Fuel-Coolant Interaction

PLE) is used to solve the Navier-Stokes equations. Since the collocated grid is used due to its flexibility, the pressure is corrected using Rhie-Chow interpolation. All these discretization processes and algorithms are implemented in a code written in MATLAB. The effect of the compression constant, the mesh sensitivity, and the face correction loop on the MULES algorithm are studied. Moreover, the effects of the inlet jet velocity and jet diameter on the jet breakup length are investigated using the MULES algorithm to capture the interface. The simulation is run with similar conditions in experimental and numerical studies found in the literature. Finally, the results are compared to studies found in the literature.
Chapter 2

Review of Literature

2.1 Previous Studies

2.1.1 Experimental Studies

There are many experimental facilities to investigate steam explosion phenomena [3, 6, 9, 10]. One of them is built by KTH (Royal Institute of Technology) in Sweden, called the MISTEE (Micro interactions in steam explosion experiments) facility. It is intensively used to investigate steam explosions by using different molten materials [11]. The MISTEE facility was also modified to perform experiments by using small jets in different inlet diameters from 5 to 10 mm. The MISTEE facility’s schematic is shown below in Figure 2.1. The stainless steel crucible has a hole in the bottom, and a stopper covers the hole during molten melt preparation. The hole size can be changed according to the initial jet diameter. When the melt reaches the desired temperatures, the stopper is raised by using a pneumatic system; in that way, the crucible’s bottom hole is revealed in order to inject the melt. The melt is poured into a rectangular plexiglass tank (500 mm × 75 mm × 75 mm) filled with water. The experiments and measurements performed at the small-scale MISTEE facility are discussed here. For convenience, it is assumed that the inlet jet diameter is equal to the bottom hole size of the crucible in MISTEE. The experiments were performed in two different inlet jet velocities, which are 1.5 m/s and 3 m/s. The corresponding Weber numbers range from 10 to
2.1 Previous Studies

320, depending on both jet velocity and diameter. The leading edge slows down, followed by fragments’ stripping from the lateral surfaces of the jet, and these stripplings lead to thinning of the melt jet and then breakup [3].

![Figure 2.1: MISTEE Facility Schematic](image)

The experiment was performed with several different materials to investigate what thermo-physical properties affect jet breakup and debris formation. Since Wood’s metal melting temperature (≈ 70°C) is below the boiling temperature of the water, the heat transfer between molten material and water can be negligible. For this reason, the material can be used to investigate the hydrodynamics of the jet breakup process. Eutectic mixtures of WO3-ZrO2 and WO3-Bi2O3 were used in the study as simulant corium material [3].

The melt jet progressing in the water pool breaks into droplets. The distance at which the jet breaks is defined as jet breakup length. According to the study, jet diameter is one of the properties affecting jet breakup length, as can be seen in Figure 2.2. In other words, as the jet diameter increases, jet breakup length increases.
(a) $D_j = 5\text{mm}, U_j = 1.3\text{m/s}$

(b) $D_j = 10, U_j = 1.7\text{m/s}$

(c) $D_j = 20, U_j = 3\text{m/s}$

(d) $D_j = 30, U_j = 3.2\text{m/s}$

Figure 2.2: Molten Wood’s metal melt jet fragmentation in a water pool [3]
2.1. Previous Studies

Iwasawa et al. [10] carried out an experiment in which material was injected into the water that was used as the coolant. Bi-Sn alloy with a low melting temperature was used to simulate molten corium. Jet breakup behavior was observed by changing the initial temperature of the coolant and molten material and jet velocity.

The experiment facility comprises a water tank, injection equipment, and measurement apparatus, including a high-speed video camera and thermocouples. To inject the molten material into the water tank (0.2 m x 0.2 m x 1.5 m), the injection equipment is used. The injection diameter is 20 mm. Using a high-speed video camera, the jet breakup behavior and the fragmentation behavior were observed. In the case of low initial temperature, it is assumed that solidification effect on the surface of the molten material becomes comparatively strong. Because of the solidification, fragmentation is suppressed. In the case of low initial velocity, it is assumed that jet instability was not likely to happen since the relative velocity between the coolant and molten material becomes low. Jet fragmentation is suppressed as well, and the molten material falls into the water pool without breakup. It is found that there is a strong relation between jet breakup behavior and the inlet velocity and initial temperature. Also, they found that jet breakup length is independent of the inlet velocity and the initial interfacial temperature when the molten material breaks into the fragments. They found that the competition between the instability growth rate on the surface of the jet and the solidification speed of the molten jet is a significant factor in fragmentation and breakup behavior [10]. Figure 2.3 shows the jet breakup behavior in 20 mm inlet diameter.
Matsuo et al. [12] have carried out an experimental study to investigate jet breakup behavior. The experimental equipment is comprised of four parts. (1) Injection apparatus is used to heat up, melt and insert the material. (2) Molten jet is injected at a specified velocity using a nitrogen pressurizing device. (3) To observe the jet fragmentation and breakup in a coolant, and a water tank is used. It is comprised of three glass plates, 1.5 m in height and 0.2 m in width. (4) To adjust the water temperature in the tank, a heater is used. Firstly, the water is heated to the desired temperature using the heater. The molten material is placed in the injection apparatus and is heated up to the desired temperature. After the molten material and the water reached at the desired temperatures, the injection apparatus is pressurized by nitrogen gas. Then, the stopper is removed to inject the molten material jet into the tank. U-alloy78 is used as a molten material in this study. The breakup behavior of the material jet is observed by means of the high-speed camera. Figure 2.4 shows the observation results of the jet behavior. The fragmentation and instabilities can be seen at the leading edge of the jet and the surface of the jet. After the breakup, the speed of the jet’s leading edge decreases and becomes steady. It is found that the jet breakup length is associated with the jet front velocity [12]. The jet diameter is 20 mm and the jet inlet velocity is 0.69 m/s in Fig. 2.4.
2.1. Previous Studies

2.1.2 Numerical Studies

Thakre and Ma [2] have investigated two-dimensional melt jet fragmentation/breakup behavior using a commercial CFD code, ANSYS FLUENT 14.0. The results were compared with the experimental results, which were obtained by KTH [3]. Since the phases are assumed immiscible, the Volume Of Fluid (VOF) method was used to simulate the two-phase flow phenomenon. This study includes the jet fragmentation pattern and the effect of jet velocity and jet diameter on the jet breakup length. It used adaptive mesh refinement to resolve the interfaces in two-phase flow, which was used successfully in their previous works [13, 14]. This method can capture the deformation of interfaces during jet breakup/fragmentation. Their simulation results demonstrate that the jet breakup pattern is axisymmetric throughout the initial deformation of the jet, which deteriorates at a later stage because of the growth of...
the instability. While the jet injection velocity is low, the leading edge breakup is dominant. On the other hand, when the jet velocity is increased, sideways stripping was observed—the jet breakup length changes without following an axisymmetric pattern. The instability on the jet surface could be a reason for the variation in the jet breakup length. Since the simulation is run by changing the jet diameter, they observed the effect of the diameter on the jet breakup length. The study shows that the growth of the jet diameter gives rise to an increased jet breakup length. This result is consistent with the experiment carried out at KTH. Also, they ran the simulation with different Weber numbers and obtained a trend matching the jet behavior investigated in the previous studies. It is realized that, although the jet breakup length is slightly greater for the less viscous jet, the effect is less significant compared with the inlet diameter [2]. Their simulation results are shown in Figure 2.5.
2.1. Previous Studies

(a) $We = 1.25, U_j = 0.5m/s, D_j = 5mm$

(b) $We = 125, U_j = 5m/s, D_j = 5mm$

(c) $We = 1280, U_j = 16m/s, D_j = 5mm$

Figure 2.5: Jet breakup and fragmentation in a water pool

[2]
Mehrdad et al. [15] have performed a similar numerical study of the jet breakup phenomenon as Thakre’s et al. [2]. They have used the Level Set Method to track the interface. The molten melt was injected into the water pool, and when the melt gets contact with the inlet, the interface was initiated. The simulation is 2D, and the domain dimensions are 100 mm×500 mm. It was solved with different grid sizes (≈ 0.66, 1.42, and 1.80 mm). The continuity equation and Navier-Stokes equations are solved. By using different jet velocities, jet breakup length and fragmentation were observed. Since Burger et al. [7] shows that inlet jet diameter has a strong effect on the jet breakup regime, they have investigated it as well. According to Mehrdad and co-authors’ [15] results, jet breakup length increased with the jet diameter, and the results are consistent with the results of the study using the VOF method. When they looked into the effect of surface tension on the jet breakup, they realized that lower surface tension causes an early jet breakup. However, around 0.8 N/m surface tension is a critical magnitude since it gives rise to the highest jet breakup length [16]. The Level Set method was used to capture the interface in this study. Dimensionless jet breakup length was compared with the VOF method result. It is seen that dimensionless jet breakup length in the LS method is smaller than in the VOF method at lower surface tensions (especially less than 1 N/m). The effects of surface tension, jet diameter, and jet inlet velocity on the jet breakup length were studied, and it was observed that there are good agreements with the previous studies [15].
2.1. Previous Studies

(a) $U_j = 1m/s, D_{jet} = 10mm$

(b) $U_j = 5m/s, D_{jet} = 10mm$

(c) $U_j = 10m/s, D_{jet} = 10mm$

Figure 2.6: Jet breakup and fragmentation in a water pool [15]
Shibata et al. [17] have numerically studied 2-D the jet breakup behavior using the MPS method (Moving Particle Semi-implicit Method). This method is a particle method for incompressible flows. The surrounding fluid of the jet is neglected. The effect of the Weber number on the jet breakup behavior is investigated. They have also studied the size of droplets distribution in different Weber numbers. It is found that the results are comparable with the experimental results. Figure 2.7 shows the jet breakup length in different Weber numbers when the inlet diameter equals 13 mm [17].

![Figure 2.7: The jet breakup length with different We numbers](image)

Zhou et al. [18] have investigated molten jet behavior, the leading edge of the jet, and the jet breakup length numerically. They have considered the heat transfer in the study as well. They used the VOF method to capture the interface. To reconstruct the interface, they used the Piecewise Linear Interface Calculation (PLIC) method. 2-D Navier-Stokes equations are
solved using the SIMPLE algorithm. They developed a code to understand and analyze the jet breakup pattern. The CSF model was used to calculate surface tension force. The simulation domain is 0.2 m x 1 m, and also, three different grid sizes (0.2 mm x 0.2 mm, 0.4 mm x 0.4 mm and 0.8 mm x 0.8 mm) were used. The bottom and two side boundaries are treated as walls. The simulation is performed with different inlet jet velocities in 20 mm inlet diameter. The simulation results are in good agreement with the experimental result [10]. Figure 2.8 shows the jet breakup behavior for a 20 mm inlet diameter.

![Figure 2.8: Jet breakup process simulated by Zhou et al.][18]

Table 2.1 summarizes the experimental and numerical studies mentioned above. The dimensionless jet breakup length, Weber number, methods [2, 15, 17, 18] and other experimental conditions [3, 10, 12] are compared in similar conditions. Even though many studies have been made, the phenomenon is not entirely understood since it includes complex mechanisms such as stripping. The coarse breakup and certain instabilities exist at the interface. For these reasons, the effects of jet velocity and jet diameter on jet behavior still need to be investigated.
### Table 2.1: Summary of numerical and experimental studies in the literature

<table>
<thead>
<tr>
<th>Research Team</th>
<th>Method</th>
<th>Jet Velocity (m/s)</th>
<th>Jet Diameter (mm)</th>
<th>We (Weber Number)</th>
<th>Dimensionless Jet Breakup Length(L/D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thakre and Ma</td>
<td>VOF</td>
<td>2.5</td>
<td>20</td>
<td>125</td>
<td>15.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td>180</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.5</td>
<td></td>
<td>245</td>
<td>13.5</td>
</tr>
<tr>
<td>Shibata and co-authors</td>
<td>MPS</td>
<td>2.773</td>
<td>13</td>
<td>100</td>
<td>10.5</td>
</tr>
<tr>
<td>Iwasawa and co-authors</td>
<td>-</td>
<td>3</td>
<td>20</td>
<td>450</td>
<td>8</td>
</tr>
<tr>
<td>(Experiment)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Matsuo and co-authors</td>
<td>-</td>
<td>2</td>
<td>10</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>(Experiment)</td>
<td></td>
<td></td>
<td></td>
<td>20</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>17.5</td>
</tr>
<tr>
<td>Zhou and co-authors</td>
<td>VOF</td>
<td>3</td>
<td>20</td>
<td>450</td>
<td>8.4</td>
</tr>
<tr>
<td>Mehrdad and co-authors</td>
<td>LS</td>
<td>2.5</td>
<td>20</td>
<td>125</td>
<td>13.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td>180</td>
<td>11.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.5</td>
<td></td>
<td>245</td>
<td>10</td>
</tr>
<tr>
<td>Manickam and co-authors</td>
<td>-</td>
<td>2.5</td>
<td>20</td>
<td>125</td>
<td>10.5</td>
</tr>
<tr>
<td>(Experiment in KTH)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td>180</td>
<td>13</td>
</tr>
</tbody>
</table>

Table 2.1: Summary of numerical and experimental studies in the literature
Chapter 3

Governing Equations and Discretization

Computational fluid dynamics (CFD) is based on fluid dynamics equations, which refer to mathematical expressions of the conservation of physics laws. The derivation of the governing equation is introduced for the single-phase flow first. Then it is discussed how to arrange the equations to adapt to the multi-phase flow. After obtaining the governing equations, the discretization process is introduced.

3.1 Single-Phase Flow

3.1.1 Continuity Equation

One of the conservation laws for fluid flow says that the matter can neither be destroyed nor created. It is assumed an arbitrary control volume $V$ is fixed in time and space as shown in Figure 3.1. According to the mass conservation law, the rate of the mass change within the control volume has to equal the net mass flux crossing the control volume’s surface $S$ [19].

The mass conservation is shown below

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0$$

(3.1)
If Eq. 3.1 is integrated over the volume $V$, the integral form of the mass conservation equation can be obtained as shown below

$$\int_V \left[ \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) \right] dV = 0$$ \hspace{1cm} (3.2)

Eq. 3.1 is valid for an arbitrary point in the control volume. If Eq. 3.1 is expanded, it can be expressed as Eq. 3.3 in the 2-D Cartesian coordinate system.

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} + \frac{\partial (\rho v)}{\partial y} = 0$$ \hspace{1cm} (3.3)

Eq. 3.3 can be further expanded as below,

$$\frac{\partial \rho}{\partial t} + u \frac{\partial \rho}{\partial x} + v \frac{\partial \rho}{\partial y} + \rho \frac{\partial u}{\partial x} + \rho \frac{\partial v}{\partial y} = 0$$ \hspace{1cm} (3.4)
3.1. Single-Phase Flow

The velocity vector of the fluid \( \vec{v} \) at any point in the flow domain is expressed by the local velocity components \( u, v \) in 2D. They are functions of location \( (x, y) \) and time \( (t) \). The above equation (Eq. 3.4) is the partial differential form of the continuity equation.

It is known that the material derivative at a function \( \phi \) in 2-D cartesian coordinate can be written as the equation below,

\[
\frac{D \phi}{Dt} = \frac{\partial \phi}{\partial t} + u \frac{\partial \phi}{\partial x} + v \frac{\partial \phi}{\partial y}
\]  

Therefore, Eq. 3.4 can be rearranged by using the material derivative property.

\[
\frac{D \rho}{Dt} + \rho \nabla \cdot \vec{v} = 0
\]  

The density of fluid does not vary with time nor spatially for incompressible flow, so the material derivative equals zero. Since the density is a non-zero scalar, the continuity equation can be simplified as Eq. 3.7.

\[
\nabla \cdot \vec{v} = 0
\]  

The final version of the continuity equation for 2D incompressible flow can be written as below,

\[
\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0
\]
3.1.2 Momentum Equation

Considering the general property of variable per unit mass indicated as \( \phi \), the material derivative of it with respect to time can be written as Eq. 3.5.

Eq. 3.5 shows the time rate of change property \( \phi \) per unit mass. If Eq. 3.5 is multiplied by the density \( \rho \), the change rate of the property of variable per unit volume can be given as below,

\[
\rho \frac{D\phi}{Dt} = \rho \frac{\partial \phi}{\partial t} + \rho u \frac{\partial \phi}{\partial x} + \rho v \frac{\partial \phi}{\partial y} = 0
\]  

Eq. 10 can be rewritten as,

\[
\rho \frac{D\phi}{Dt} = \frac{\partial (\rho \phi)}{\partial t} + \frac{\partial (\rho u \phi)}{\partial x} + \frac{\partial (\rho v \phi)}{\partial y}
\]  

Newton’s second law of motion denotes that the sum of forces that acts on the fluid elements equals the product between the acceleration and mass of the element. This is shown by considering the \( x \) component of the law,

\[
\sum F_x = ma_x
\]  

where \( a_x \) and \( F_x \) are the acceleration and force in the \( x \) direction. The acceleration is the
3.1 SINGLE-PHASE FLOW

Figure 3.2: Surface forces performing on the volume of control for $u$ velocity component [19]

time rate of change of $u$, which is shown by the material derivative. Hence,

$$a_x = \frac{Du}{Dt}$$  \hspace{1cm} (3.13)

The right-hand side of Eq. 3.12 can be expressed as,

$$ma_x = \rho \frac{Du}{Dt} \Delta x \Delta y$$  \hspace{1cm} (3.14)

There are two sources of the force acting on the fluid element on the left-hand side term of Eq. 3.12. They are surface forces and body forces. The kind of body forces that could influence the rate of change of the momentum of the fluid are gravity, centrifugal, electromagnetic forces, and Coriolis. The influence of the surface forces can be shown in Figure 3.2 for $u$ velocity component. These forces are the normal stress $\sigma_{xx}$ and the tangential stresses $\tau_{yx}$ and $\tau_{zx}$ acting on the fluid element’s surface [19]. If the sum of surface forces and the time rate change of the $u$ component is combined, the x-momentum equation can
be written as,
\[
\rho \frac{Du}{Dt} = \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \sum F_{x}^{bodyforces}
\] (3.15)

The normal stress and the tangential stress components for 2D are expressed by,
\[
\sigma_{xx} = -p + \tau_{xx}
\] (3.16)

where \(\tau_{xx}\) and \(\tau_{yx}\)
\[
\tau_{xx} = 2\mu \frac{\partial u}{\partial x} + \lambda \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right)
\] (3.17)
\[
\tau_{yx} = \mu \left( \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right)
\] (3.18)

where \(p\) and \(\mu\) are the pressure and dynamic viscosity. According to Stokes hypothesis, for Newtonian fluids \(\lambda = -\frac{2}{3}\mu\) is often used [19]. For the two-dimensional flow, the x-momentum equation can be written as below after the stress-strain terms included,
\[
\frac{Du}{Dt} = -\frac{1}{\rho} \frac{\partial P}{\partial x} + \nu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + \frac{\partial}{\partial x} \left[ \lambda \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) \right] + \frac{\partial}{\partial x} \left[ \nu \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) \right] + \frac{\partial v}{\partial x} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \frac{\partial u}{\partial y} = 0
\] (3.19)

After simplifications, the x-momentum equation is given by,
\[
\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} = -\frac{1}{\rho} \frac{\partial P}{\partial x} + \nu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + \frac{g}{\rho} \frac{\partial u}{\partial y}
\] (3.20)
the y-momentum equation is similar to the x-momentum equation, and it is given by,

\[
\begin{align*}
\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} = & \quad - \frac{1}{\rho} \frac{\partial P}{\partial y} + \nu \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) + g
\end{align*}
\]  

(3.21)

where \( \nu \) is kinematic viscosity \( (\nu = \mu/\rho) \).

After all derivations, the governing equations except the energy equation for two-dimensional single-phase flow can be summarized as below,

\[
\begin{align*}
\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = & \quad 0 \\
\frac{\partial \rho u}{\partial t} + \rho u \frac{\partial u}{\partial x} + \rho v \frac{\partial u}{\partial y} = & \quad -\frac{\partial P}{\partial x} + \mu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + \rho g \\
\frac{\partial \rho v}{\partial t} + \rho u \frac{\partial v}{\partial x} + \rho v \frac{\partial v}{\partial y} = & \quad -\frac{\partial P}{\partial y} + \mu \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) + \rho g
\end{align*}
\]

3.2 Multi-Phase Flow Equations

In the incompressible single-phase flow, there is only one type of fluid, so it can be solved by using the equations above. If there are more than one immiscible fluid, those equations are not enough to capture the multi-phase flow behaviors. It is possible to use one set of governing equations for the entire flow domain filled by the different phases. The different phases are assumed as one fluid with variable properties of the material that vary abruptly at the boundary of the phase. It is needed to add extra forces (via \( \delta \)-functions), which exist at the interface, to the equations. This form of the equation is called the “one-fluid” approach. The surface tension force should be added as a body force to the momentum equations [20].

The two-fluid model can be expressed using two sets of conservation equations. The
continuity and momentum equations must be applied in each phase. The two-fluid model has different phase interaction terms which represent the mass and momentum transfer from one phase to second phase [21]. On the other hand, in the one-fluid model, the multi-phase flow problem can be solved using one set of equations.

Besides the surface tension force, a void fraction transport equation is needed to solve the multi-phase flow problems in order to capture the interface by using one set of governing equations.

### 3.2.1 Surface Tension Force

When the properties of fluid vary discontinuously because abrupt changes happen in molecular forces, surface tension, which is a characteristic of the material, plays an important role at the interface. Surface tension results in a local surface force that exerts itself on elements of fluid at interfaces in both the tangential and normal directions. The motion of fluid interface caused by surface tension plays a significant role in many industrial and natural phenomena. Cavitation [22], hydrodynamic stability [23], and fuel sprays in internal combustion engines [24] are some examples. The surface tension force is written by \( f_\sigma \) integrated over \( V \) for a control volume that, includes an interface.

\[
\text{Surface tension force} = \int_{CV} f_\sigma \delta S dV \tag{3.22}
\]

The surface tension force can be written as Eq.3.23.

\[
f_\sigma \delta S = \sigma \kappa n \delta S \tag{3.23}
\]

where, \( \sigma, \kappa, n, \delta S \) are surface tension constant, curvature, interface normal vector, and delta
function, respectively. Even if the viscosity \( \mu \) field and the density \( \rho \) field vary discontinuously at the interface, Eq. 3.23 are satisfied for the whole flow domain \([20]\).

There are some different models to calculate the surface tension force in a numerical simulation. One of those is the Continuum Surface Force (CSF) model, which is used commonly in multi-phase flow simulations. It was proposed by Brackbill \textit{et al.} and has been applied to model incompressible fluid flows \([25]\).

Since the delta function affects the surface tension term, the choice of delta function in a numerical simulation plays an important role. For this reason, researchers have developed different calculations of delta function \([25, 26, 27]\). Some of the approximations for the delta function used in studies are shown in Table 3.1. Brackbill’s delta function \( |\nabla \tilde{\alpha}| \) is used in this study, where \( \alpha \) represents the void fraction, and it will be introduced in the next sections. " \( \sim \) " represents a smooth factor and it is used to smooth interface in numerical simulations.

<table>
<thead>
<tr>
<th>Authors</th>
<th>Delta function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brackbill \textit{et al.}</td>
<td>(</td>
</tr>
<tr>
<td>M. ten Caat \textit{et al.}</td>
<td>( 4\alpha(1 - \alpha) )</td>
</tr>
<tr>
<td>Meier \textit{et al.}</td>
<td>( \delta_S n = \frac{\nabla p(x) \cdot 2 \rho(x)}{\rho_1 - \rho_2 \rho_1 + \rho_2} )</td>
</tr>
</tbody>
</table>

Table 3.1: Delta functions used in the literature
3.2.2 Curvature Calculation

Curvature defining the change in unit tangent vector to a curve with respect to the surface length of the arc along the curve is calculated using Eq. 3.24 in the CSF model [25].

\[ \kappa = - (\nabla \cdot \vec{n}) \]  

(3.24)

where \( \vec{n} \) is the surface unit normal. To calculate the surface unit normal at the interface, the smoothed void fraction is used in the CSF model [25]. The unit normal is given by,

\[ \vec{n} = \frac{\nabla \tilde{\alpha}}{|\nabla \tilde{\alpha}|} \]  

(3.25)

The surface tension force can be written as below,

\[ f_{\sigma} \delta_s = -\sigma (\nabla \cdot \vec{n}) \frac{\nabla \tilde{\alpha}}{|\nabla \tilde{\alpha}|} |\nabla \tilde{\alpha}| \]  

(3.26)

After simplification, the surface tension term can be obtained as below

\[ f_{\sigma} \delta_s = -\sigma (\nabla \cdot \vec{n}) \nabla \tilde{\alpha} \]  

(3.27)

3.3 The Volume of Fluid Method

In computational fluid dynamics, the 2-D single-phase problem has three primary unknown variables (velocity components and pressure). The system is complete with the continuity equation, momentum equations and proper initial and boundary condition. However,
3.3. The Volume Of Fluid Method

if the problem is multiphase flow, an additional equation is needed to track the interface. There are some different methods used in the studies, such as the Level Set method (LS) and the Moving Particle Semi-implicit method (MPS). The Volume Of Fluid method (VOF) is used in this study to capture the interface.

The VOF method, a numerical method of free surface approximation, is commonly used to solve immiscible two or more fluids flow in numerical simulations. It is developed by Hirt and Nichols [28]. A variable called the volume fraction of the phase is introduced in the computational cell for each additional phase. The sum of the volume fractions of all phases is unity. All variables and properties defined in each cell represent the volume-averaged values. For this reason, the properties and variables in the cells represent either one phase or a mixture of the phases, based on the volume fraction values. In other words, if the xth fluid’s volume fraction in the cell is shown as alpha x, then there are three possible conditions [28]. These conditions and the time-dependent void fraction equation are shown below,

![Figure 3.3: True interface](image)
• \( \alpha_x = 0 \) : the cell is occupied by other fluid or fluids

• \( \alpha_x = 1 \) : the cell occupied by xth fluid only

• \( 0 < \alpha_x < 1 \) : an interface between xth fluid and other fluid or fluids exists in the cell

\[
\frac{\partial \alpha}{\partial t} + \nabla . (\alpha \vec{v}) = 0
\]  

(3.28)

Considering the continuity equation, the above void fraction equation can be expressed as below,

\[
\frac{\partial \alpha}{\partial t} + u \frac{\partial \alpha}{\partial x} + v \frac{\partial \alpha}{\partial y} = 0
\]  

(3.29)

There are different types of methods used for the reconstruction of the interface in the literature. The common methods include Simple Line Interface Calculation (SLIC) [30] and Piecewise Linear Interface Calculation (PLIC)[31].

The Multidimensional Universal Limiter for Explicit Solution (MULES) algorithm is used for an implicit interface reconstruction in this study. The algorithm will be explained after the surface tension discretization in later sections.

Since the field equations have been obtained, the next step is to discretize the equations. The discretization process is described in the next section.
3.4 The Finite Volume Method

The finite volume method is one of the most popular methods used in Computational Fluid Dynamics (CFD). It enables flexible discretization. It is widely used, especially for fluid simulation problems and related transport phenomena. It was first used by McDonald [32] for 2-D inviscid flow simulation. The computational domain is subdivided into a finite number of grid cells, hexahedra in 3D and quadrilaterals in 2D. All control volumes have to satisfy the conservation equations. Each of the control volumes can be depicted through an algebraic equation in which a series of the neighboring nodal values appear [19].

The accuracy of the spatial discretization depends on the particular schemes with which fluxes are assessed. There are some different possibilities for defining the position and shape of the control volume with regard to the grid cells. Two main approaches can be distinguished: Cell-centered scheme: fluids’ properties (flux, pressure, density, viscosity, etc.) are stored at the center of the grid cells. Cell-vertex scheme: the variables of the flow are stored at the grid points. One of the most significant advantages of the finite volume method is that spatial discretization can be performed directly in the physical space. Also, it can be implemented easily in complex geometries [33].

The domain is divided into small discrete control volumes as in the first step. Since control volumes’ boundaries are placed between adjacent nodes, a control volume or cell is surrounded by each node. 'P' is defined as a general nodal point surrounded by its neighbors in 2D geometry; nodes on the east, west, south, and north are called as follows: E, W, S, N, respectively. Control volume faces are defined by e, w, s, n, respectively [34]. A schematic of the control volume and grid structure is shown in Figure 3.4.
3.4.1 Discretization Process

The geometric discretization of the physical domain results in a mesh on which the equations are solved eventually. The numerical solution of a partial differential equation comprises of finding the dependent values $\phi$ at specific points from which its distribution over the domain of interest could be constructed, or grid nodes, a process called meshing. The obtained nodes are generally located at vertices or centroids of cells based on the discretization procedure. The focus in all discretization methods is on replacing the continuous solution of the partial differential equation by discrete values \[36\]. Discretization of the partial differential equation is conducted term by term in the next section.

3.4.2 Discretization of Continuity Equation

The continuity equation is shown in Eq. 3.30 for the two-dimensional cases.
3.4. The Finite Volume Method

\[ \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \] (3.30)

By using Eq. 3.30, the discretized continuity equation can be obtained as below.

\[ \frac{u_e - u_w}{\Delta x} + \frac{v_n - v_s}{\Delta y} = 0 \] (3.31)

If the central differencing scheme is considered, face velocities are equal to the average of their neighbor’s velocities \([19]\).

\[ u_e = \frac{u_E + u_P}{2}, \quad u_w = \frac{u_P + u_W}{2}, \quad v_n = \frac{v_N + v_P}{2}, \quad v_s = \frac{v_P + v_S}{2} \]

The final form of the continuity equation can be obtained as Eq. 3.32

\[ \frac{u_E - u_W}{2\Delta x} + \frac{v_N - v_S}{2\Delta y} = 0 \] (3.32)
### 3.4.3 Discretization of Momentum Equation

\[ \frac{\partial \rho u}{\partial t} + \nabla \cdot (\rho \vec{v}\phi) = \nabla \cdot (\mu \nabla \phi) - \frac{\partial P}{\partial x} + \rho g + F_{surf} \]  \hspace{1cm} (3.33)

1 : Transient Term  \hspace{1cm} 4 : Pressure Term

2 : Advection Term  \hspace{1cm} 5 : Gravity Term

3 : Diffusion Term  \hspace{1cm} 6 : Surface Tension Term

The equation is discretized term by term.

Firstly, the transient term is integrated over the control volume. Then the discretized form of it can be obtained as Eq 3.34.

\[ \int_{CV} \frac{\partial \rho u}{\partial t} dV = \int_{x} \int_{y} \frac{\rho_{p}^{t+1}u_{p}^{t+1} - \rho_{p}^{t}u_{p}^{t}}{\Delta t} dxdy = \frac{\rho_{p}^{t+1}u_{p}^{t+1} - \rho_{p}^{t}u_{p}^{t}}{\Delta t} \Delta x \Delta y \]  \hspace{1cm} (3.34)

Here, subscript \( t \) refers to the previous time step, \( t + 1 \) refers to the current time step, and \( \Delta t \) represents the time interval.

For the advection or convection term discretization, it is integrated over control volume again. Then by using the Gauss Divergence theorem, it is converted to a surface integral (Eq 3.36) [37].

where \( \vec{n} \) is the unit normal vector. Gauss’s divergence theorem can be expressed as Eq. 3.35.

\[ \int_{V} \nabla \cdot \rho \vec{v} dV = \int_{S} \rho \vec{v} \cdot \vec{n} dS \]  \hspace{1cm} (3.35)
Then the equation can be written as below,

\[
\int_S (\rho \vec{V} \phi) \cdot \vec{n} dS = \left[ \rho_x u_x u_e^{t+1} (\Delta y) + \rho_y v_y v_e^{t+1} (\Delta x) + \rho_z v_z v_s^{t+1} (\Delta x) \right] (3.37)
\]

It is known that \( \rho u \Delta x \) or \( \rho v \Delta y \) means mass flux, so the discretized form of the term can be obtained as Eq. 3.38

\[
\int_S (\rho \vec{V} \phi) \cdot \vec{n} dS = \dot{m}_e u_e^{t+1} - \dot{m}_w u_w^{t+1} + \dot{m}_n u_n^{t+1} - \dot{m}_s u_s^{t+1} (3.38)
\]

Even though the central differencing is more accurate since it is of second-order accuracy, the upwind scheme is used in the convection term since it is more stable than central differencing. The upwind scheme calculation is shown below.

\[
u_e^{t+1} = \begin{cases} u_P & \text{if } \dot{m}_e \geq 0 \\ u_E & \text{if } \dot{m}_e < 0 \end{cases} (3.39)
\]

\[
u_w^{t+1} = \begin{cases} u_W & \text{if } \dot{m}_w \geq 0 \\ u_P & \text{if } \dot{m}_w < 0 \end{cases} (3.40)
\]

\[
u_n^{t+1} = \begin{cases} u_P & \text{if } \dot{m}_n \geq 0 \\ u_N & \text{if } \dot{m}_n < 0 \end{cases} (3.41)
\]
\[ u_s^{t+1} = \begin{cases} 
    u_S & \text{if } \dot{m}_s \geq 0 \\
    u_P & \text{if } \dot{m}_s < 0 
\end{cases} \tag{3.42} \]

The vertical velocity component \( v \) is calculated for the convection term using the same upwind scheme.

The diffusion term is integrated over the control volume, and it is converted to surface integral by using the Gauss Divergence theorem.

\[ \int_{CV} \nabla \cdot (\mu \nabla \phi) dV = \int_S \mu \nabla \phi \cdot ndS \tag{3.43} \]

Eq. 3.43 can be expanded as below, and the discretized form of the diffusion term can be obtained.

\[
\int_S \mu \nabla \phi \cdot ndS = \left[ \mu_e \left( \frac{\partial \phi}{\partial x} \right)_e \Delta y + \mu_w \left( \frac{\partial \phi}{\partial x} \right)_w (-\Delta y) + \mu_n \left( \frac{\partial \phi}{\partial y} \right)_n \Delta x + \mu_s \left( \frac{\partial \phi}{\partial y} \right)_s (-\Delta x) \right] \tag{3.44}
\]

\[
\int_S \mu \nabla \phi \cdot ndS = \left[ \mu_e \left( \frac{\partial \phi}{\partial x} \right)_e \Delta y + \mu_w \left( \frac{\partial \phi}{\partial x} \right)_w (-\Delta y) + \mu_n \left( \frac{\partial \phi}{\partial y} \right)_n \Delta x + \mu_s \left( \frac{\partial \phi}{\partial y} \right)_s (-\Delta x) \right] \\
= \mu_e \left( \frac{\phi_E - \phi_P}{\Delta x} \right)^{t+1} \Delta y + \mu_w \left( \frac{\phi_W - \phi_P}{-\Delta x} \right)^{t+1} (-\Delta y) + \mu_n \left( \frac{\phi_N - \phi_P}{\Delta y} \right)^{t+1} \Delta x + \mu_s \left( \frac{\phi_S - \phi_P}{-\Delta x} \right)^{t+1} (-\Delta y) \tag{3.45}
\]

The pressure term and gravity term can be discretized in the same way, and their discretized form is shown in Eq. 3.46 and Eq. 3.47, respectively.
3.4. **The Finite Volume Method**

\[
- \int_{CV} \frac{\partial P}{\partial x} dV = -(P_e^{t+1} - P_w^{t+1}) \Delta y
\]  
(3.46)

\[
P_e = \frac{P_e + P_0}{2}, P_w = \frac{P_w + P_0}{2}
\]

\[
\int_{CV} \rho g dV = \rho g \Delta x \Delta y
\]  
(3.47)

### 3.4.4 Discretization of Surface Tension

Surface tension force has been given in a previous section (Eq. 3.27). The first factor is the surface tension constant. The second one is the curvature which is the divergence of the unit normal, and the last one is the gradient of void fraction which approximates the delta function.

\[
f_{\sigma_S} \delta_S = -\sigma(\nabla \cdot \vec{n}) \nabla \alpha
\]

The gradient alpha is calculated for the nodes of the cells by using FVM.

\[
\nabla \alpha_e = \frac{\alpha_E - \alpha_P}{\Delta x}
\]  
(3.48)

\[
\nabla \alpha_w = \frac{\alpha_P - \alpha_W}{\Delta x}
\]  
(3.49)

\[
\nabla \alpha_n = \frac{\alpha_N - \alpha_P}{\Delta y}
\]  
(3.50)

\[
\nabla \alpha_s = \frac{\alpha_P - \alpha_S}{\Delta y}
\]  
(3.51)

The curvature is equal to the divergence of the unit normal, and it can be expressed as,

\[
\nabla \vec{n} = \frac{\partial n}{\partial x} + \frac{\partial n}{\partial y}
\]  
(3.52)
As the alpha gradient, the unit normal for x and y-direction can be obtained using the equations below.

\[ \vec{n} = \frac{\nabla \alpha}{|\nabla \alpha|} \] (3.53)

The magnitude of the gradient of \( \alpha \) can be calculated as Eq. 3.54

\[ |\nabla \alpha| = \left[ \left( \frac{\alpha_E - \alpha_P}{\Delta x} \right)^2 + \left( \frac{\alpha_N - \alpha_P}{\Delta y} \right)^2 \right]^{1/2} \] (3.54)

Thus, the curvature is,

\[ \kappa = \frac{\partial n_x}{\partial x} + \frac{\partial n_y}{\partial y} \] (3.55)

and the curvature at the cell centroid "P" can be obtained as below,

\[ \kappa_P = \frac{n_{x:e} - n_{x:w}}{\Delta x} + \frac{n_{y:n} - n_{y:s}}{\Delta y} \] (3.56)

All momentum equation terms are discretized. In summary, the set of governing equations for multi-phase flow can be written below. The continuity equation is the same as the incompressible single-phase flow equations. Surface tension force is an additional term in the multi-phase flow momentum equations. \( \rho_{mix} \) and \( \mu_{mix} \) are used if the cell contains an interface and it is calculated doing interpolation. The last equation is the void fraction equation to capture the interface.
3.4. The Finite Volume Method

\[
\begin{align*}
\frac{\partial \rho_{\text{mix}} u}{\partial t} + \rho_{\text{mix}} u \frac{\partial u}{\partial x} + \rho_{\text{mix}} v \frac{\partial u}{\partial y} &= -\frac{\partial P}{\partial x} + \mu_{\text{mix}} \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + \rho_{\text{mix}} g + f_{\sigma_x} \delta S \\
\frac{\partial \rho_{\text{mix}} v}{\partial t} + \rho_{\text{mix}} u \frac{\partial v}{\partial x} + \rho_{\text{mix}} v \frac{\partial v}{\partial y} &= -\frac{\partial P}{\partial y} + \mu_{\text{mix}} \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) + \rho_{\text{mix}} g + f_{\sigma_y} \delta S \\
\frac{\partial \alpha}{\partial t} + u \frac{\partial \alpha}{\partial x} + v \frac{\partial \alpha}{\partial y} &= 0
\end{align*}
\]

3.4.5 MULES Algorithm

The Multidimensional Universal Limiter with Explicit Solution (MULES) algorithm is used to solve the VOF equation in this study. MULES, which is based on Flux Corrected Transport (FCT), is one of the methods implemented in OpenFOAM. FCT technique is developed by Boris and Book \[38\], and it was improved by Zalesak \[39\] as a method to warrant boundedness in the hyperbolic problems’ solution \[40\].

Face fluxes are computed in the discretized domain, and the transport equation is solved explicitly. The desired boundedness is obtained by limiting the computed face fluxes. It must be sure that there is no large-scale smoothing of the gradient at the free surface by the discretization method. The gradient must be kept between 0 and 1. For this reason, the VOF equation’s extended version is used \[41\].

\[
\frac{\partial \alpha}{\partial t} + \nabla \cdot (\vec{v} \alpha) + \nabla(u_r \alpha(1 - \alpha)) = 0
\]

(3.57)

where \(u_r\) is the compression velocity, it is equal to the equation below.

\[
u_r = u_\alpha - u_{1-\alpha}
\]

(3.58)
The additional term is called the artificial compression term. It can be seen that the term is merely active at the interface because all $\alpha$ values are 0 or 1 except at the interface. So the artificial compression term does not affect the original equation (3.28).

To discretize Eq. 3.57, it is integrated over a control volume as below.

$$
\int_{CV} \left[ \frac{\partial \alpha}{\partial t} + \nabla \cdot (\vec{v} \alpha) + \nabla (u_r \alpha (1 - \alpha)) \right] dV = 0
$$

By using the Gauss Divergence Theorem, a semi-discretized form can be obtained as below,

$$
\left( \frac{\partial \alpha}{\partial t} \right) V + \sum_f \alpha_f \vec{v}_f \cdot S_f + \sum_f \alpha_f (1 - \alpha)_f (u_r)_f \cdot S_f = 0
$$

By using the face fluxes, $u_f \cdot S_f = F_f$, $v_f \cdot S_f = F_f$, and $(u_r)_f \cdot S_f = F_r^f$, the equation for a computational cell becomes

$$
\frac{\alpha_{P}^{t+1} - \alpha_{P}^{t}}{\Delta t} V_P + \sum_f \alpha_f^n (F_f)^n + \sum_f \alpha_f^n (1 - \alpha)_f^n (F_r)^n = 0
$$

The compressive flux is calculated as

$$
F_r^f = \hat{n}_f C_\alpha \left| \frac{F_f}{S_f} \right|
$$

where $\hat{n}_f$ is the face unit normal flux, and $C_\alpha$ is the compression constant. The face unit normal flux is computed as

$$
\frac{(\nabla \alpha)_f}{|(\nabla \alpha)_f + \delta n|} \cdot S_f
$$

where $\delta n$, a small stabilization factor, is used to prevent division by zero. It is computed as
3.4. The Finite Volume Method

below.

\[
\delta n = \frac{\varepsilon}{\left[ \frac{1}{N} \sum_{i=1}^{N} |V_i| \right]^{1/3}} \tag{3.64}
\]

\(\varepsilon\) is set to \(10^{-8}\) in OpenFOAM.

All these definitions, computations are followed by the calculation of a high order flux \((F_f^{\alpha,H})\) and a low order flux \((F_f^{\alpha,L})\)

\[
F_f^{\alpha,H} = \alpha_f F_f + \alpha_f (1 - \alpha_f) F^r_f \tag{3.65}
\]

Low order flux is calculated by using Upwind Differencing Scheme. This approach keeps the solution bounded. On the other hand, High order flux is computed by using Central Differencing Scheme, and it keeps the gradient sharpened at the expense of possible unboundedness [41].

By subtracting the low order flux from high order flux, the anti-diffusive flux can obtain as below.

\[
A_f = F_f^{\alpha,H} - F_f^{\alpha,L} \tag{3.66}
\]

Anti-diffusive flux is used in order to obtain the corrected flux

\[
F_f^{\alpha,C} = F_f^{\alpha,L} + \lambda_f A_f \tag{3.67}
\]

where \(\lambda_f\) is a weighting factor, and it is used to prevent the creation of new local extrema and to ensure that global limits are between 0 and 1 [41].

After these definitions, Eq. 3.61 can be rewritten as below
\[
\alpha_{P}^{t+1} = \alpha_{P}^{t} - \frac{\Delta t}{|V|} \sum_{f} F_{f}^{\alpha,C}
\]  

Eq. 3.68 is an explicit equation, and it is used to obtain \( \alpha \) values for the new time step.

One of the most significant steps in the MULES is the computation of the weighting factors \( \lambda_{f} \). It is calculated by doing iterations [40]. And it initiates with the computation of the local extrema as below

\[
\alpha_{max,L,t} = \max \left\{ \alpha_{P}^{t}, \alpha_{NB}^{t} \right\}
\]

\[
\alpha_{min,L,t} = \min \left\{ \alpha_{P}^{t}, \alpha_{NB}^{t} \right\}
\]

where \( NB \) represents neighbors of "P" cell. Then total outflow \( F_{P}^{-} \) and inflow \( F_{P}^{+} \) of the anti-diffusive flux \( A \) are calculated.

\[
F_{P}^{-} = \sum_{f} A_{f}^{+}
\]

\[
F_{P}^{+} = -\sum_{f} A_{f}^{-}
\]

where \( A_{f}^{+} \) and \( A_{f}^{-} \) are the outflows and the inflows, respectively, per face. After that local minimum and maximum, which are calculated, are needed to be corrected by using the global maximum and minimum.

\[
\alpha_{max,t} = \min \left\{ \alpha_{max}^{G}, \alpha_{P}^{max,L,t} \right\}
\]

\[
\alpha_{min,t} = \max \left\{ \alpha_{min}^{G}, \alpha_{P}^{min,L,t} \right\}
\]
The values obtained in that step are used to calculate the net flux per cell

\[
Q_P^- = \frac{V}{\Delta t} (\alpha_P^t - \alpha_{P}^{\text{min},t}) - \sum_f F_{f}^{\alpha,L}
\]

\[
Q_P^+ = \frac{V}{\Delta t} (\alpha_P^{\text{max},t} - \alpha_P^t) + \sum_f F_{f}^{\alpha,L}
\]

The weighting factors iteration begins with the initial guess \( \lambda_f = 1 \) for all faces. And the next step values are computed by using previous \( \lambda_f \) values. Next step values can be computed as below

\[
\lambda_{P}^{+,n+1} = max \left\{ \min \left( -\sum_f \lambda_f^n A_f^+ + Q_P^-, \frac{1}{F_P^-} \right), 0 \right\}
\]

\[
\lambda_{P}^{-,n+1} = max \left\{ \min \left( \sum_f \lambda_f^n A_f^- + Q_P^+, \frac{1}{F_P^+} \right), 0 \right\}
\]

After the calculation of lambda cell values \((P)\), the new face lambda values are computed.

\[
\lambda_f^{n+1} = \begin{cases} 
\min \left( \lambda_{P}^{+,n+1}, \lambda_{P}^{-,n+1} \right), & \text{if } A_f \geq 0 \\
\min \left( \lambda_{P}^{-,n+1}, \lambda_{P}^{+,n+1} \right), & \text{if } A_f < 0
\end{cases}
\]

(3.69)

where \( n \) and \( n + 1 \) represent previous and current step iteration, respectively.

After the user-specified number of iterations is completed, Eq. 3.68 is solved explicitly\([41]\). The current time step void fraction values are obtained. After obtaining void fraction values at the current time step, velocity and pressure are computed. The process is repeated until the desired number of time steps has been reached. The calculation of velocity and pressure is explained in the next section.
3.5 Rhie-Chow Interpolation

Navier-Stokes equation is already discretized in previous sections. The discretized x-momentum equation can be written as

\[
\rho_P \frac{u_{P}^{t+1} - u_{P}^{t}}{\Delta t} \Delta x \Delta y + \dot{m}_e u_e^{t+1} - \dot{m}_w u_w^{t+1} + \dot{m}_n u_n^{t+1} - \dot{m}_s u_s^{t+1} = \\
\mu_e \left( \frac{u_E - u_P}{\Delta x} \right)^{t+1} \Delta y + \mu_w \left( \frac{u_W - u_P}{-\Delta x} \right)^{t+1} (-\Delta y) + \mu_n \left( \frac{u_N - v_P}{\Delta y} \right)^{t+1} \Delta x + \\
\mu_s \left( \frac{v_S - v_P}{-\Delta x} \right)^{t+1} (-\Delta y) - (P_{e}^{t+1} - P_{w}^{t+1}) \Delta y + \rho_P g \Delta x \Delta y
\]

Eq. 3.70 can be written as a linear system equation as below

\[
a_P^u u_P + \sum_{NB(P)} a_{NB}^u u_{NB} = b_P^u
\]

where \(b_P^u\) is the source term, and the pressure term can be taken out from the source term

\[
b_P^u = -|V_P| (\nabla P)_P + \hat{b}_P^u
\]

Both sides in Eq. 3.70 are divided by \(a_P^u\)

\[
u_P + \sum_{NB(P)} \frac{a_{NB}^u}{a_P^u} u_{NB} = - \frac{|V_P|}{a_P^u} (\nabla P)_P + \frac{b_P^u}{a_P^u}
\]

This can be rearranged as

\[
u_P + H_P[u_{NB}] = -D_P^u \nabla P + B_P^u
\]
Here, $D^u_P$ is an unit matrices, $B^u_P$ is a vector, and $H_P$ is the matrix placed the $'P'$ cell’s neighbors’ coefficients.

$$H_P[u] = \sum_{NB(P)} \frac{a^u_{NB}}{a^u_P}$$

$$D^u_P = \frac{|V_P|}{a^u_P}$$

$$B^u_P = \frac{\hat{b}^u_P}{a^u_P}$$

The Rhie-Chow interpolation is developed by Rhie and Chow [42] while the Navier-Stokes equation is solved for the $'P'$ cell, only the pressure of its neighbor cells are used in the equation. It is acted as if it is independent of its own. It gives rise to the checkerboard effect. By means of interpolation, the possible checkerboard effect can be avoided. If a staggered grid arrangement is used, there is no need to use the Rhie-Chow interpolation. However, for a collocated mesh, the Rhie-Chow interpolation must be used to avoid the checkerboard effect.

The face values are calculated by using linear interpolation, which is expressed with an over-bar in the derivations. To obtain $'P'$ cells’ values and its neighbors, the geometrically corrected factor ($f_x$) is used [41]. The face values can be given as.

$$\bar{\phi}_f = f_x\phi_P + (1 - f_x)\phi_{NB} \quad (3.75)$$

By adding a dissipation term including two variants of the pressure gradient’s face values, the interpolation can be used for face velocity values as below

$$u_f = \bar{u}_f - D^u_f(\nabla P_f - \nabla \bar{P}_f) \quad (3.76)$$
Figure 3.6 shows ‘P’, ‘E’, ‘W’ cells and ‘w’, ‘e’ faces in 1-D domain. Over-bar term can be calculated using Eq. 3.77.

\[ \nabla P_f = \frac{\nabla P_e + \nabla P_w}{2} \quad (3.77) \]

3.6 The SIMPLE Algorithm

The Semi-Implicit Method for Pressure Linked Equations (SIMPLE) is used to solve incompressible Navier-Stokes equations, and it is an iterative method [37]. It was first developed by Prof. Brian Spalding and his student S. Patankar for a staggered grid solution [43, 44, 45]. Also, It is possible to use the algorithm for collocated grids. However, the Rhie-Chow interpolation must be used to avoid the checkerboard effect [41].

The algorithm begins with initial guess values for \( u^t, \dot{m}^t, P^t \); after that, the equation is solved for the velocity field.

\[ u^*_P + H_P[u^*_{NB}] = -D_P^{\nu}\nabla P_P + B_P^{\nu} \quad (3.78) \]

This solution satisfies the momentum equations, not the mass conservation. Then some corrections fields are needed (\( u', \dot{m}', P' \)) in order to get both continuity and momentum equations satisfied. The desired solution is shown as
\[ u = u^* + u' \]
\[ P = P^t + P' \]
\[ \dot{m} = \dot{m}^* + \dot{m}' \]

It gives
\[ \sum_{f \in P} \dot{m}'_f = - \sum_{f \in P} \dot{m}^*_f \tag{3.79} \]
for the equation of mass conservation, \( \dot{m}^*_f \) is calculated as
\[ \dot{m}^*_f = \rho_f u^*_f \cdot S_f \tag{3.80} \]
where \( u^*_f \) is the estimated face velocity, and it is computed by using Rhie-Chow interpolation as
\[ u^*_f = \overline{u^*}_f - \overline{D^*_f}(\nabla P^t_f - \nabla P'_f) \tag{3.81} \]

The difference between Eq. 3.74 and Eq. 3.78 gives
\[ u'_P + H_P[u'] = -D^u_P(\nabla P')_P \tag{3.82} \]
also mass flow correction can be written as
\[ \dot{m}'_f = \rho_f u'_f \cdot S_f \tag{3.83} \]
By subtracting Eq. 3.76 from Eq. 3.81, \( u'_f \) can be calculated as

\[
\begin{align*}
u'_f &= \overline{u}_f - D_f^u (\nabla P'_f - \nabla \overline{P}'_f) \\
&= (3.84)
\end{align*}
\]

Eq. 3.80 and Eq. 3.84 are placed in Eq. 3.79 as below

\[
\sum_f (\rho_f \overline{u}_f \cdot S_f) + \sum_f (\rho_f D_f^u \nabla P'_f \cdot S_f) - \sum_f (\rho_f D_f^u \nabla P'_f \cdot S_f) = - \sum_f \dot{m}_f^s \\
&= (3.85)
\]

where \( f \in F(P) \)

Considering the respective neighboring cell \( NB \) related with a face \( f \), using Eq. 3.82 can be written for the face values as

\[
\begin{align*}
\overline{u}'_f + H_f [u'] &= - D_f^u (\nabla P'_f) \\
&= (3.86)
\end{align*}
\]

that can be rewritten as

\[
\begin{align*}
\overline{u}'_f + D_f^u (\nabla P'_f) &= - H_f [u'] \\
&= (3.87)
\end{align*}
\]

If the equation above is inserted in Eq. 3.85, the pressure correction equation can be written as

\[
\begin{align*}
- \sum_f (\rho_f D_f^u \nabla P'_f \cdot S_f) &= - \sum_f \dot{m}_f^s + \sum_f (\rho_f H_f [u'] \cdot S_f) \\
&= (3.88)
\end{align*}
\]

The underlined term, which is the unknown correction field, can be neglected since it does not change the final solution. Because it is a correction component in an iterative procedure that becomes zero for the converged result. Nevertheless, it affects the behavior of convergence and results in bigger pressure correction. For this reason, an under-relaxation factor generally is utilized for the pressure correction [41].
The pressure correction equation can be written in algebraic form as

\[ a_P P' P + \sum_{NB} a_{NB} P'_{NB} = b_P' \]  

(3.89)

and the term on the right-hand side given below

\[ b_P' = -\sum_f \dot{m}_f^* + \sum_f (\rho_f H_f [u'] \cdot S_f) \]  

(3.90)

By using Rhie-Chow interpolation as in Eq. 3.81, the mass flow rate \( \dot{m}_f^* \) can be calculated. By using an under-relaxation (\( \lambda^P \)) the corrected values are obtained as below [41].

\[ u^{**}_P = u'_P + u'_P \quad \text{with} \quad u'_P = -D^u_P \nabla P' \]  

(3.91)

\[ \dot{m}_f^{**} = \dot{m}_f^* + \dot{m}_f' \quad \text{with} \quad \dot{m}_f' = -\rho_f D^u_P \nabla P' \cdot S_f \]  

(3.92)

\[ P^{**}_P = P'_P + \lambda^P P' \]  

(3.93)

The steps for the collocated SIMPLE algorithm can be summarized below:

1. Begin with the initial solution \((u'^t, \dot{m}_f^t, P'^t)\) or guess at time \(t\).

2. Solve the momentum equation by using Eq. 3.78 to obtain momentum conserving velocity \(u^*\).

3. Use the velocity field calculated in the previous step \(u^*\), calculate an updated mass flow rate \(\dot{m}^*\) using Rhie-Chow interpolation (Eq. 3.80 and Eq. 3.81).
4. Create the pressure correction equation (Eq. 3.88) with the mass flow rate that was computed at the previous step, then solve it to obtain a pressure correction field $P'$. 

5. Update the pressure and velocity fields by using this pressure correction with Eq. 3.91 to obtain $u^{**}, \dot{m}^{**}, P^*$. 

6. Use the results from the previous step as a new initial guess $(u^t, \dot{m}^t, P^t)$ and repeat the procedure from step 2 until convergence is achieved. 

7. The converged result is the exact solution at time $t + \Delta t$ and one continues to the next iteration. 

8. Repeat all steps until the last time step is done.
Figure 3.7 shows the flow chart of the developed in-house CFD code. The two-phase flow code begins with the MULES algorithm to determine the current step void fraction in every cell. Then the velocity and pressure are computed for the same time step using the SIMPLE algorithm. After the converged velocity and pressure results are found, it is turned back to the MULES algorithm to calculate next time step values. It continues until all time steps are completed.

Figure 3.7: Flow chart of the developed code
Chapter 4

Results and Discussion

The numerical algorithms presented in the previous chapter are implemented in a CFD code written in MATLAB, and the simulation results obtained by the code are shown in this chapter. The flow is incompressible laminar flow, and heat transfer is not considered. First, the single-phase flow results in a horizontal channel are shown, and the results are compared with the analytical solution. Then, the MULES algorithm is verified under different conditions. Finally, the simulation of the molten corium poured in a water pool is performed with different inlet diameters and inlet velocities. The results were compared with the previous simulation and experimental results found in the literature.

4.1 Single-Phase Flow Results

The single-phase flow results are obtained for steady-state and transient conditions in a horizontal channel as a preliminary verification of the written code.

4.1.1 Steady-State Single-Phase Flow

The single-phase laminar flow is simulated to obtain the velocity profile and pressure change in a 2-D horizontal channel whose length is 500 mm and width is 3.75 mm. The mesh size is 1x0.25 mm². Water enters the horizontal channel with a uniform velocity. Since
no-slip conditions are applied to the walls, velocity is zero at the bottom and top sides of the channel. On the other hand, the velocity reaches maximum values at the middle height of the channel. When the flow is fully developed, the velocity profile becomes parabolic based on analytical solution.

Figure 4.1: Domain geometry and boundary conditions

Figure 4.2 shows how the velocity profile develops when the fluid goes through the channel. The line being solid black represents the analytical solution when the flow reaches the fully developed flow. The water velocity is 1 m/s at the entrance of the channel. The centerline velocity develops and reaches around 1.5 m/s, which is the analytical solution at the middle height of the channel.

The analytical solution shows the velocity distribution from the bottom wall to the top wall. The code results show the velocity of the cell centers.

To prevent vacuum effect, on the bottom and top wall, the pressure gradient is assumed to equal to the cell center located at the bottom and top wall, respectively.
The change of horizontal velocity through the entire channel can be visualized in Fig 4.3. On the inlet side of the channel, the velocity is around 1 m/s; however, when the fluid progresses, the velocity approaches zero at the top and bottom of the channel because of the no-slip boundary condition. On the other hand, due to the continuity equation, the velocity approaches 1.5 m/s in the middle of the channel. The flow has a fully developed profile around four-fifths of the channel’s length.

Figure 4.3: Horizontal velocity distribution

Figure 4.4 demonstrates how the pressure changes in the channel. At the left inlet pressure is given as 1 atm or 101325 N/m$^2$. As is expected, the pressure decreases due to wall friction
4.1. Single-Phase Flow Results

when the fluid progresses in the channel.

![Pressure Distribution](image)

Figure 4.4: Pressure distribution

4.1.2 Unsteady Single-Phase Flow

In this case study, the unsteady equations for single-phase flow are solved, and the time-dependent results are obtained. Parametric studies are also carried out by changing properties in the channel, such as mesh size, the number of mesh, initial velocities, and time step size.

The channel dimension is 300x5 mm$^2$. Two different mesh sizes are used to study how the mesh size affects velocity distribution.
Figure 4.5: Different mesh sizes simulation’s velocity distribution

Fig. 4.5a shows that the velocity changes in different locations in the channel. The fluid entering with 1 m/s in the channel accelerates till it is fully developed; however, when the flow reaches the fully developed state, the velocity of fluid should be 1.5 m/s based on the analytical solution. As can be seen, the maximum velocity is about 1.3 m/s. If the channel were longer, the flow would reach the fully developed flow. This means the flow is still developing. On the other hand, Fig. 4.5b shows the result of the same channel; the only difference is the size of the mesh. The flow has shown a fully developed velocity profile, and the maximum velocity is around 1.5 m/s, which agrees with the analytical solution.
4.1. SINGLE-PHASE FLOW RESULTS

<table>
<thead>
<tr>
<th></th>
<th>Mesh Size</th>
<th>Initial Velocity</th>
<th>Time Step</th>
<th>Mesh Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig 4.5a</td>
<td>3 x 0.5 mm²</td>
<td>1 m/s</td>
<td>0.05 sec</td>
<td>100 x 10</td>
</tr>
<tr>
<td>Fig 4.5b</td>
<td>0.75 x 0.5 mm²</td>
<td>1 m/s</td>
<td>0.05 sec</td>
<td>400 x 10</td>
</tr>
</tbody>
</table>

Table 4.1: Mesh size comparison

4.1.3 Effect of Initial Velocity

In this case study, there are two scenarios, and it is assumed that there is a uniform flow (0.8 m/s and 1 m/s) in the channel. This simulation is to study how the velocity changes in a certain node and how long it takes to reach a fully developed flow.

Figure 4.6 does not show real channel mesh size. It was used for illustration. The velocity variation with time in point A and point B is shown in Fig 4.7 and Fig 4.8 for two different conditions.

Two situations with different initial velocities are performed: 0.8 m/s and 1 m/s. Fig 4.7a shows how the liquid entering the channel with uniform 1 m/s changes velocities with time in the outlet bottom cell. Fig 4.7b shows how the liquid entering the channel with uniform 1 m/s changes velocities with time in the outlet mid-height cell. The time interval is 0.05 sec, and the flow reaches the fully developed region at approximately 0.4 sec.

Fig. 4.8 demonstrates the same thing as the Fig. 4.7. The only difference between them is the initial velocity in the channel. Initial velocity is 1 m/s in this figure, and when the flow
CHAPTER 4. RESULTS AND DISCUSSION

Figure 4.7: Specific cell velocity changes with time

(a) B point’s velocity variation with time
(b) A point’s velocity variation with time

(reaches the fully developed flow, the velocities have expected values. Fig. 4.8a represents the last node located on the bottom side of the channel, and the velocity approaches zero because of the viscous stress. On the other hand, Fig. 4.8b refers to the last node located in the middle of the channel, and as expected, it has a maximum velocity of around 1.5 m/s. So the parabolic velocity profile is obtained.

Figure 4.8: Specific cell velocity changes with time

Independent of the velocity (0.8 m/s or 1 m/s) in the channel, the liquid entering the channel with 1 m/s, the flow is fully developed with 1.5 m/s. The only difference between Fig 4.8 and Fig 4.7 is the time to reach fully developed. Thus, it is obtained that the simulation results are comparable with the analytical solution even under different conditions.)
To verify the MULES algorithm, simulation with a 2-D bubble in a uniform \( v = 1 \text{ m/s} \) velocity field is performed. The channel and its size are shown in Figure 4.9a. A bubble was created in the channel, and the bubble’s \( \alpha \) values are considered as 1. The water’s \( \alpha \) values are 0. The interface’s values are between 0 and 1, as shown in Figure 4.9b. The color bar shows the void fraction values in Fig 4.9b.

The momentum and continuity equations were not solved in the MULES verification cases. Only the void fraction equation was solved.

![Figure 4.9: (a) Domain geometry and the initial conditions (b) void fraction distribution and void fraction color bar](image)

Figure 4.10 shows the bubble’s progression in the vertical channel. In the beginning, the bubble is placed near the bottom of the channel. It can be seen that there is no significant difference between the initial and final shape of the bubble. The bubble’s shape is mostly
conserved since the velocity in the channel is uniform.

Figure 4.10: The flow of a bubble in the channel

Figure 4.11 shows the void fraction distribution the bubble’s center plane in the initial and final time steps. While in the beginning, the interface is sharp. The void fraction inside of the bubble shows a slight decrease at the final time. The summation of the void fraction’s initial and final conditions are compared to check whether the summation of the void fraction is conserved. The total $\alpha$ values are conserved with 0.2 % error. This error is probably caused by the numerical diffusion. The bubble’s shape was mostly conserved since the velocity is uniform in the channel.
In another case study, the simulation is run in the same channel, but with a parabolic velocity profile. The bubble evolution is shown in Figure 4.12. As it can be seen that even though the interface is sharp at the beginning, it shows some diffusion with time. The bubble shape takes the form of the velocity profile eventually.

Figure 4.13 shows the void fraction distribution of the bubble’s center plane in the initial and final time steps, and the color bar shows the void fraction values. The bubble has diffused more since the velocity is parabolic. The total $\alpha$ values are again conserved with 0.2% error.
(a) \( t = 0 \text{sec} \)  
(b) \( t = 0.3 \text{sec} \)  
(c) \( t = 0.6 \text{sec} \)  
(d) \( t = 0.9 \text{sec} \)  
(e) \( t = 1.5 \text{sec} \)  
(f) void fraction  

**Figure 4.12:** The flow of a bubble in the channel

void fraction color bar

**Figure 4.13:** Void fraction distribution
4.2. MULES Algorithm Verification

Figure 4.14 shows the histograms of the initial and final time void fraction in the channel. Fig 4.15 is the zoomed-in view for initial and final conditions of void fraction distribution in the channel. There is some difference between initial and final void fraction distributions. As can be seen, no counts was observed around 0.7 – 0.9 in the beginning. Since the bubble is diffused, the void fraction counts at 1 decreased and counts between 0.7 – 0.9 increased in the final time step. However, there is no considerable changes in lower void fraction counts.

![Void fraction distribution](image1.png)

(a) Initial condition  (b) Final condition

Figure 4.14: Void fraction distribution

![Void fraction distribution zoom-in viewed](image2.png)

(a) Initial condition  (b) Final condition

Figure 4.15: Void fraction distribution zoom-in viewed

To further verify the MULES algorithm, a classical notched disk case is simulated. The
disk is located in 1x1 m$^2$ pool that has a rotational velocity. The rotational velocity equations are shown below.

$$v_x = rsin\Theta$$

$$v_y = rcos\Theta$$

The mesh size is 1x1 mm$^2$. As it can be seen in Fig 4.16 $\alpha = 1$ shows the notched disk, water $\alpha$ values are 0 and the interface $\alpha$ values are between 0 and 1. The color map shows the void fraction values in Fig 4.16. Figure 4.16a shows the initial condition. Then with the rotational velocity field, the notched disk started to turn around itself. The disk has diffused slightly at $t = 0.15$ s, but its shape was well conserved. Also, the total $\alpha$ values were conserved.

![Figure 4.16: A turning notched disk in a vortex pool](image)

Figure 4.16 a shows the histogram of the void fraction in the pool at the beginning and Fig 4.17 b shows the final time step. Since it can not be seen clearly, it was zoomed in using Fig 4.18. It can be seen that the interface is sharp at the beginning. The effect of the vortex velocity, the notched disk was diffused. Although total $\alpha$ values are preserved, the effect of the diffusion, interface $\alpha$ values increased.
4.2. MULES Algorithm Verification

(a) Initial condition  
(b) Final condition

Figure 4.17: Void fraction distribution of the notched disk

(a) Initial condition  
(b) Final condition

Figure 4.18: Void fraction distribution of the notched disk

After obtaining reasonable results from the MULES algorithm, the effect of several other important parameters such as $C_\alpha$, $\lambda$ and the mesh size are also studied.

4.2.1 Effect of $C_\alpha$ on the MULES

As mentioned in Chapter 3, $C_\alpha$ is the compression constant, and it is one of the factors that affect the compressive flux. The default $C_\alpha$ is set to 1.5 in OpenFoam.

The code was run for different $C_\alpha$ values in a channel (0.5x0.2 m$^2$) having uniform velocity.
The mesh size is 2x2 mm², and the bubble diameter is 80 mm. Figure 4.19 shows how the bubble has diffused at 0.4 s for different $C_\alpha$. The color bar shows the void fraction values.

When $C_\alpha$ is increased, since it causes $F^r$ to increase, the bubble is compressed more.

Figure 4.20 shows how the total $\alpha$ values vary in different $C_\alpha$. The summation of the initial condition void fraction and final condition void fraction in the channel are compared to obtain the total error. When the $C_\alpha$ is equal to 3.5 or 5, the total void fraction can not be conserved, and the error rises. While the error rate is 0.4% at $C_\alpha = 0.5$, the total void fraction is increased since there is not enough artificial compressive flux. When $C_\alpha$ is equal to 1.5, the total void fraction error is 0.2%, and it does not exceed the total void fraction.
4.2.2 Effect of the Mesh Size

To investigate the effect of mesh size on the MULES algorithm, the code was run with different meshes, which are 1x1 mm$^2$, 2x2 mm$^2$, and 2.5x2.5 mm$^2$. Figure 4.21 shows the bubbles and bubbles’ interface in the water channel having uniform velocity and the color bar shows the void fraction values. It can be seen that the interface is sharp for the finer mesh, whereas the coarse mesh shows a softer interface.
The diffusion of the void fraction at the bubble’s cross-section in the initial and final conditions is shown in Fig. 4.22. In comparison, the bubble has a sharp interface when the fine mesh (1x1 mm$^2$) was used. The diffusion increases when the mesh size increases, and the interface was softened near the end of the simulation for the coarse mesh (2x2 mm$^2$).
4.2. MULES Algorithm Verification

4.2.3 Effect of $\lambda$ on the MULES

As mentioned in the previous chapter on the MULES algorithm, to calculate face $\lambda$ values, an iteration loop is used in every time step. $\lambda$ is defined as the number of iterations for flux correction. This section investigates how the number of loops affects the total void fraction.

Figure 4.23 shows the change of total void fraction with time when different number of
CHAPTER 4. RESULTS AND DISCUSSION

Figure 4.23: Total void fraction values’ variation with time

iteration loops are used. It can be seen that when the iteration number is equal to 1, the total void fraction cannot be conserved. The total void fraction tends to increase with time. The summation of the void fraction initial condition and final condition in the channel are compared to find the error. When the iteration number equals 2, the total void fraction increases slightly with an error around 0.5% after 0.4 s. When the number of the iteration is equal to 3 or 5, it gives a reasonable result with a 0.2% error.

4.3 Jet Fragmentation Simulation

After the MULES algorithm is verified, the SIMPLE algorithm is combined with the MULES algorithm to solve the jet fragmentation problem. Firstly, the MULES algorithm is solved in every time interval to find out the void fraction. Then the SIMPLE algorithm is solved to obtain velocity and pressure in every cell.
To be able to compare the results with the previous studies in the literature, the same domain size as the KTH experiment and Thakre and his colleagues’ simulation [13] was chosen. The length of the pool is 500 mm, and the width of the pool is 75 mm. The molten material enters the pool from the top middle of the top boundary. Except for the inlet cells, the other cells placed at the top of the pool are specified as outlet cells in order to satisfy the mass conservation. The domain which is used in the simulation is shown in Fig 4.24. The simulation is carried out different mesh sizes $5 \times 1.5 \text{ mm}^2$, $4 \times 2 \text{ mm}^2$, and $4 \times 1.25 \text{ mm}^2$. The finer mesh ($4 \times 1.25 \text{ mm}^2$) results are shown in the current study since it gave more sensitive and reasonable results. The domain is divided into 6250 ($125 \times 50$) control volumes and the simulation is run with the coarse mesh.

When different inlet diameters and different inlet velocities of the molten material are injected into a water pool, the jet breakup length is studied using the simulation, and the results are compared to numerical [13] and experimental [3] studies found in the literature.
Fluid properties are shown as below.

\[ \rho_{\text{water}} = 1000 \text{ kg/m}^3, \quad \rho_{\text{corium}} = 9700 \text{ kg/m}^3, \quad \mu_{\text{water}} = 8.9 \times 10^{-4} \text{ Pa.s}, \quad \mu_{\text{corium}} = 1.94 \times 10^{-3} \text{ Pa.s}, \quad \sigma = 1 \text{ N/m}, \quad P = 1 \text{ atm}, \quad T = 20 \ ^\circ \text{C} \]

4.3.1 Effect of Jet Diameter

The code is run with different inlet diameters in order to understand how the jet diameter \((D_{\text{jet}})\) affects the jet breakup length \((L)\).

Figure 4.25 shows how the corium jets with different inlet diameters break up and fragment into droplets in the water pool. As it can be seen that inlet diameter has a strong effect to the jet breakup length. When the inlet diameter is increased from 6 mm to 12 mm, the melt breaks up much later. However, as it is mentioned above, the effect of jet diameter on the jet breakup length is observed in smaller diameters. If the diameter increases enough, the jet breakup length can decrease because the Weber number increases, and the regime approach the atomization regime. It triggers the instabilities happening on the leading edge and sideways on the jet surface.
4.3. Jet Fragmentation Simulation

(a) $U_{jet} = 1 \text{ m/s}, \ We = 6, \ D_{jet} = 6 \text{ mm}$

(b) $U_{jet} = 1 \text{ m/s}, \ We = 9, \ D_{jet} = 9 \text{ mm}$

(c) $U_{jet} = 1 \text{ m/s}, \ We = 12, \ D_{jet} = 12 \text{ mm}$

Figure 4.25: Jet breakup and fragmentation in the water pool (500x75 mm²)
Figure 4.26 shows the velocity fields as a zoom field for three different simulations. As the jet diameter effect was observed in smaller diameters and also the inlet jet velocity is at 1 m/s. For this reason, the velocity fields are similar. The jet creates a vortex velocity effect; it can be seen clearly on the leading edge of the jet.

(a) $D_{jet} = 6$ mm, $t = 0.3$ s  
(b) $D_{jet} = 9$ mm, $t = 0.3$ s  
(c) $D_{jet} = 12$ mm, $t = 0.3$ s

Figure 4.26: Velocity fields in the water pool (500x75 mm²)

The current results are compared with the results from Thakre’s simulation and the data from experiments at KTH as shown in Figure 4.27. The velocity range is between 1.4 m/s and 1.7 m/s in the experimental study. The inlet jet velocity is 1.5 m/s in the current simulation and Thakre’s simulation. Therefore there is a slight difference in the inlet velocity between these results. Compared with Thakre’s simulation results, the current simulation shows some underestimation in the jet breakup length, especially for bigger diameters. While the
difference between the two different simulations results is small in the smaller diameters, it increases slightly for bigger diameters. The same trend between experimental data and the current study can be observed. However, the current results show a slight underestimation for bigger diameters. The underestimation of the jet breakup length may be caused by the difference in inlet velocity and relatively coarse mesh used in the current study.

Figure 4.27: Jet breakup length comparison
4.3.2 Effect of Inlet Velocity

Ginsberg [4] identified several jet breakup regimes which are laminar, transition, turbulent, and atomization. These regimes are determined by ambient Weber number. When the ambient Weber number increases, the regime goes from the laminar to atomization. The jet breakup length is investigated in the Weber number range of $1.25 - 300$. The ambient Weber number is defined below. Since it is proportional to the velocity squared, it drastically increases as inlet velocity.

\[ We_a = \frac{\rho v^2 D_{jet}}{\sigma} \]

In the current study, $\rho = 1000 \text{ kg/m}^3$, $\sigma = 1 \text{ N/m}$ and $v$ range is from 0.5 m/s to 4 m/s. The size of the pool is 500x75 mm$^2$, and the mesh size is 4x1.25 mm$^2$.

Two types of instabilities are found to be important to the jet breakup/fragmentation behavior in the literature. These are Kelvin-Helmholtz instability (KH) and Taylor instability. Taylor instability could happen at the leading edge because of the jet’s deceleration in the surrounding medium [7]. Afterward, the jet enters the water pool; a continuous stripping occurs from the jet surface. It gives rise to thinning of the jet, and eventually, the coarse breakup could happen [5]. As the jet inlet velocity increases, the flow parallel to the surface of the jet dominates the surface tension effect and causes droplets’ stripping. This stripping can thin the jet and cause jet breakup, and this instability is defined Kelvin-Helmholtz instability [13]. Mostly, it would occur when the jet inlet velocity is higher.

The simulation is carried out with the different Weber numbers to study its impact on the jet breakup length. Figure 4.28 shows the jet breakup/fragmentation with different Weber numbers and $D_{jet} = 5 \text{ mm}$. Inlet jet velocity is lower ($U_{jet} = 1 \text{ m/s}$) in Fig 4.28a.
The jet is generally axisymmetric, and there is no instability on the surface. Deformation happens mostly at the leading edge. After the jet reaches some length, the jet breakup is triggered. Fig 4.28b shows the jet breakup pattern for \( We = 11 \) \( (U_{jet} = 1.5 \text{ m/s}) \). The breakup happens at the leading edge, and the large fragments leave from the jet breakup further and create smaller droplets. The transition between turbulent and atomization regimes is shown in Fig 4.28c for \( We = 80 \) \( (U_{jet} = 4 \text{ m/s}) \). Severe stripping from the surface of the jet to the flow can be observed. Since the mesh is relatively coarse, the instabilities can not be seen clearly. However, the jet breakup length can be obtained from the simulation results. The jet breakup length is greater for the case \( We = 11 \). After this point, if the jet inlet velocity increases, the flow enters the atomization regime, which results in smaller droplets. The jet breakup length is decreased dramatically.

![Figure 4.28: Jet breakup with different Weber numbers for \( D_{jet} = 5 \text{ mm} \)](image)

The simulation is carried out by using similar conditions to Thakre’s simulation to compare to his results. The dimensionless jet breakup length \( (L/D_{jet}) \) is observed with the different velocities for \( D_{jet} = 5 \text{ mm} \). Between the used Weber ranges, it was obtained two jet peaks for breakup length\( (\text{We}= 5 \text{ and } \text{We}= 80) \). The first peak refers to the regime changes from laminar to transition. The second peak refers to the regime changes from turbulent to
Figure 4.29: Breakup length variation with respect to Weber number

atomization. Between $\text{We} = 1 - 10$, Rayleigh-Taylor instability, which results in the coarse breakup, dominates the jet breakup/fragmentation. After the Weber number exceeds 80, the jet breakup length decreases since atomization is triggered. There is a combined effect of the coarse breakup and stripping mechanism. The stripping results in smaller size droplets that are called the atomization zone. As it can be seen in Figure 4.29 the results are comparable with Thakre’s simulation results.
Table 4.2 summarizes the current study and the previous numerical and experimental studies in similar conditions.

<table>
<thead>
<tr>
<th>Research</th>
<th>Method</th>
<th>Jet Velocity (m/s)</th>
<th>Jet Diameter (mm)</th>
<th>We (Weber Number)</th>
<th>Dimensionless Jet Breakup Length (L/D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thakre and Ma</td>
<td>VOF</td>
<td>1.5</td>
<td>5</td>
<td>11.25</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>22.5</td>
<td>16.5</td>
</tr>
<tr>
<td>Shibata and co-authors</td>
<td>MPS</td>
<td>2.773</td>
<td>13</td>
<td>100</td>
<td>10.5</td>
</tr>
<tr>
<td>Matsuo and co-authors</td>
<td></td>
<td>-</td>
<td>2</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>(Experiment)</td>
<td></td>
<td></td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Current Study</td>
<td>VOF</td>
<td>1.5</td>
<td>5</td>
<td>11.25</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>22.5</td>
<td>14.5</td>
</tr>
<tr>
<td>Mehrdad and co-authors</td>
<td>LS</td>
<td>2.5</td>
<td>20</td>
<td>125</td>
<td>13.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td>180</td>
<td>11.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.5</td>
<td></td>
<td>245</td>
<td>10</td>
</tr>
<tr>
<td>Manickam and co-authors</td>
<td></td>
<td>-</td>
<td>1.4-1.7</td>
<td>~11.25</td>
<td>14.4-18</td>
</tr>
<tr>
<td>(Experiment in KTH)</td>
<td></td>
<td></td>
<td>5</td>
<td>~20.25</td>
<td>16.3-17.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: Comparison of the current study and numerical and experimental studies found in the literature
Chapter 5

Conclusion and Future Work

In case of a severe nuclear accident, molten corium may pour into the water pool in the form of a jet. FCI phenomenon happens, and it could result in a steam explosion. Some experimental and numerical studies have observed the strong interaction between water and molten corium in the literature.

The primary purpose of this study is to investigate how the inlet jet velocity and jet diameter affect the jet fragmentation behavior and jet breakup length. A two-dimensional in-house-CFD code has been developed using MATLAB script language. To verify the code, the continuity and momentum equations were first discretized by using FVM to solve a single-phase flow problem in a horizontal channel under steady-state and unsteady conditions. The liquid having uniform velocity enters the channel, and after some time, the velocity profile becomes parabolic, which agrees with the analytical solution.

After good results were obtained for the single-phase problem, the governing equations for multi-phase flow were discretized. There are two significant differences between single-phase flow and multi-phase flow equations. The first one is the surface tension force term in the momentum equations, and the second one is an extra equation to solve the void fraction equation. The one-fluid approach was used, and it enables solving multi-phase flow problems using one set of governing equations.

The interface solution process plays an essential role in multi-phase flow problems. There
are several methods for reconstructing interface in the literature, such as PLIC and SLIC. Even though these methods have been used in the past studies, they are computationally expensive since the interface has to be reconstructed every step. The MULES algorithm has been used to capture the interface in this study. Since it does not need to explicitly reconstruct the interface, it facilitates code implementation and speeds up the numerical simulation. In addition, the algorithm prevents numerical diffusion of the interface. Further, this in-house code makes it easier to customize and add physical or numerical model for future studies. To the author’s knowledge, it is the first study that used the MULES algorithm to investigate corium jet breakup and fragmentation phenomena in numerical simulation.

The MULES algorithm has been verified with different case studies. Generally, good simulation results were obtained. The effect of compression constant \( C_\alpha \), the number of iteration loops for \( \lambda \) coefficient, and mesh size on the MULES algorithm were studied. It is confirmed that when \( C_\alpha = 1.5 \), the simulation gave the best results in the current study. The total \( \alpha \) value was conserved. When the number of iteration loops for \( \lambda \) coefficient is greater than 3, the algorithm could yield reasonable void fraction results. It is found that when the mesh size decreases, the interface is sharp, while the coarse mesh shows a softer interface.

Finally, a multi-phase flow MATLAB script has been developed. The SIMPLE algorithm was used to solve Navier-Stokes equations, and the MULES algorithm was used to solve the void fraction equation. To study jet breakup/fragmentation phenomena, the code was run in the same initial conditions and geometry as the experiment performed at KTH. The effects of the jet inlet velocity and jet diameter on the jet breakup/fragmentation have been investigated. It is seen that as the jet diameter increases, jet breakup length increases. This result is comparable with the experimental and numerical studies in the literature. Lastly, the jet velocity effect on the jet breakup length is consistent when it is compared to the
previous numerical study. These results support that the MULES algorithm is capable of simulating jet breakup/fragmentation in melt-coolant interaction.

A relatively coarse mesh has been used in the current study, and the mesh size can be refined in the future studies to observe the instabilities happening at the jet interface. Also, the oscillation of the free surface of the water pool has been ignored in the current study. The oscillation can be considered for future work. Moreover, the code can be extended by considering heat transfer, a significant factor in the jet fragmentation process.
Bibliography


[37] HK Versteeg and W Malalasekera. An introduction to computational fluid dynamics. 


[40] Santiago Márquez Damián. *An Extended Mixture Model for the Simultaneous Treatment 


[44] Suhas V. Patankar and D. B. Spalding. A calculation procedure for heat, mass and 
   momentum transfer in three-dimensional parabolic flows. *International Journal of Heat 

Appendices
Appendix A

The MULES Algorithm MATLAB Code

```matlab
% A) Unit Normal Calculation
%  Alpha Gradient
Alpha_grad_x = (Alpha_e0 - Alpha_w0)/dx;
Alpha_grad_y = (Alpha_n0 - Alpha_s0)/dy;

%  Magnitude of the Alpha Gradient
Mag_grad_Alpha = sqrt(Alpha_grad_x.^2 + Alpha_grad_y.^2);

%  Unit Normals
n_x = -Alpha_grad_x./(Mag_grad_Alpha + del_n);
n_y = -Alpha_grad_y./(Mag_grad_Alpha + del_n);

%  Face Unit Normal Flux
n_e_flux(:,1:jmax-1) = -(n_x(:,1:jmax-1) + n_x(:,2:jmax)).*dy/2;
n_e_flux(:,jmax) = 0;
n_w_flux(:,2:jmax) = -(n_x(:,1:jmax-1) + n_x(:,2:jmax)).*(-dy/2);
n_w_flux(:,1) = 0;
n_n_flux(1:imax-1,:) = -(n_y(1:imax-1,:) + n_y(2:imax,:)).*dx/2;
n_n_flux(imax,:) = -n_y(imax,:).*dx;
n_s_flux(2:imax,:) = -(n_y(1:imax-1,:) + n_y(2:imax,:)).*(-dx/2);
n_s_flux(1,:) = 0;

% B) Estimation of Flux : Feast, Fwest, Fnorth & Fsouth
Feast = u_0(:,2:jmax+1).*dy;
Fwest = -u_0(:,1:jmax).*dy;
Fnorth = v_0(2:imax+1,:).*dx;
Fsouth = -v_0(1:imax,:).*dx;
```
% C) Estimation of Low-Order Flux : FL_east, FL_west, FL_north & FL_south
% Low-Order Flux (computed using the upwind-difference scheme)
FL_east = nan(imax, jmax);
FL_west = nan(imax, jmax);
FL_north = nan(imax, jmax);
FL_south = nan(imax, jmax);
for i=1:imax
    for j=1:jmax-1
        if Feast(i, j)< 0
            FL_east(i, j) = Feast(i, j)*Alpha0(i, j+1);
        else
            FL_east(i, j) = Feast(i, j)*Alpha0(i, j);
        end
    end
    FL_east(i, jmax) = Feast(i, jmax)*Alpha0(i, jmax);
end
for j=2:jmax
    if Fwest(i, j)<0
        FL_west(i, j) = Fwest(i, j)*Alpha0(i, j-1);
    else
        FL_west(i, j) = Fwest(i, j)*Alpha0(i, j);
    end
end
FL_west(i, 1) = Fwest(i, 1)*Alpha0(i, 1);
end
for j=1:jmax
    for i=1:imax-1
        if Fnorth(i, j)<0
            FL_north(i, j) = Fnorth(i, j)*Alpha0(i+1, j);
        end
    end
    FL_north(i, jmax) = Fnorth(i, jmax)*Alpha0(i, jmax);
end
else
    FL_north(i,j) = Fnorth(i,j)*Alpha0(i,j);
end
end
FL_north(imax,j) = Fnorth(imax,j)*Alpha0(imax,j);
for i=2:imax
    if Fsouth(i,j)<0
        FL_south(i,j) = Fsouth(i,j)*Alpha0(i-1,j);
    else
        FL_south(i,j) = Fsouth(i,j)*Alpha0(i,j);
    end
end
FL_south(1,j)= Fsouth(1,j)*Alpha0(1,j);
end

% D)4 Compressive Flux which are East,West,North and South
Fnorth_comp = n_n_flux.* C_alpha.* abs(Fnorth./dx);
Fsouth_comp = n_s_flux.* C_alpha.* abs(Fsouth./dx);
Feast_comp = n_e_flux.* C_alpha.* abs(Feast./dy);
Fwest_comp = n_w_flux.* C_alpha.* abs(Fwest./dy);

% E) High Order Flux Calculation : East West North and South
Alpha_e = Alpha_e0;   Alpha_w = Alpha_w0;
Alpha_n = Alpha_n0;   Alpha_s = Alpha_s0;

%   East Side High Order Flux Calculation
for i=1:imax
    Alpha_e(i,1) = 0.5*(Alpha0(i,2) + Alpha0(i,1));
for j=2:jmax-1
Alpha_e(i,j) = 0.5*(Alpha0(i,j+1) + Alpha0(i,j));
end
end
FH_east = Alpha_e.*Feast + Alpha_e.*(1 - Alpha_e).*Feast_comp;
% West Side High Order Flux
for i=1:imax
    Alpha_w(i,2) = 0.5*(Alpha0(i,2) + Alpha0(i,1));
    for j=3:jmax
        Alpha_w(i,j) = 0.5*(Alpha0(i,j) + Alpha0(i,j-1));
    end
end
FH_west = Alpha_w.*Fwest + Alpha_w.*(1 - Alpha_w).*Fwest_comp;
% North Side High Order Flux
for j=1:jmax
    Alpha_n(1,j) = 0.5*(Alpha0(2,j) + Alpha0(1,j));
    for i=2:imax-1
        Alpha_n(i,j) = 0.5*(Alpha0(i+1,j) + Alpha0(i,j));
    end
end
FH_north = Alpha_n.*Fnorth + Alpha_n.*(1 - Alpha_n).*Fnorth_comp;
% South Side High Order Flux
for j=1:jmax
    Alpha_s(2,j) = 0.5*(Alpha0(2,j) + Alpha0(1,j));
    for i=3:imax
        Alpha_s(i,j) = 0.5*(Alpha0(i,j) + Alpha0(i-1,j));
    end
end
\[ F_{\text{south}} = \text{Alpha}_s \cdot F_{\text{south}} + \text{Alpha}_s \cdot (1 - \text{Alpha}_s) \cdot F_{\text{south\_comp}}; \]

\% F) Anti-Diffusive Flux (A)
\% We have 4 Anti-Diffusive Fluxes on the faces (East, West, North, South)
A_e = F_{\text{east}} - F_{\text{L\_east}}; \% East Side Anti-Diffusive Flux
A_w = F_{\text{west}} - F_{\text{L\_west}}; \% West Side Anti-Diffusive Flux
A_n = F_{\text{north}} - F_{\text{L\_north}}; \% North Side Anti-Diffusive Flux
A_s = F_{\text{south}} - F_{\text{L\_south}}; \% South Side Anti-Diffusive Flux

\% G) Calculating the Local Maxima and Local Minima Values for alpha at each cell
\[ \text{Alpha\_maxL}(2:\text{imax}-1,2:\text{jmax}-1) = \max(\max(\max(\text{Alpha}0(2:\text{imax}-1,2:\text{jmax}-1), \text{Alpha}0(3:\text{imax},2:\text{jmax}-1)), \text{Alpha}0(2:\text{imax}-1,1:\text{jmax}-2)); \]
\% At Boundaries
\[ \text{Alpha\_maxL}(1,2:\text{jmax}-1) = \max(\max(\max(\text{Alpha}0(1,2:\text{jmax}-1), \text{Alpha}0(1,3:\text{jmax})), \text{Alpha}0(2,2:\text{jmax}-1)), \text{Alpha}0(1,1:\text{jmax}-2)); \]
\[ \text{Alpha\_maxL}(\text{imax},2:\text{jmax}-1) = \max(\max(\max(\text{Alpha}0(\text{imax},2:\text{jmax}-1), \text{Alpha}0(\text{imax},3:\text{jmax})), \text{Alpha}0(\text{imax},1:\text{jmax}-2)); \]
\[ \text{Alpha\_maxL}(2:\text{imax}-1,1) = \max(\max(\max(\text{Alpha}0(2:\text{imax}-1,1), \text{Alpha}0(3:\text{imax},1)), \text{Alpha}0(2,1:\text{jmax}-2)), \text{Alpha}0(1,1:\text{jmax}-2); \]
\[ \text{Alpha\_maxL}(2:\text{imax}-1,\text{jmax}) = \max(\max(\max(\text{Alpha}0(2:\text{imax}-1,\text{jmax}), \text{Alpha}0(3:\text{imax},\text{jmax})), \text{Alpha}0(2,\text{imax}-2)), \text{Alpha}0(1,\text{imax}-2)); \]

\% At Corners
\[ \text{Alpha\_maxL}(1,1) = \max(\max(\text{Alpha}0(1,1), \text{Alpha}0(1,2)), \text{Alpha}0(2,1)); \]
\[ \text{Alpha\_maxL}(\text{imax},1) = \max(\max(\text{Alpha}0(\text{imax},1), \text{Alpha}0(\text{imax},2)), \text{Alpha}0(\text{imax}-1,1)); \]
\[ \text{Alpha\_maxL}(\text{imax},\text{jmax}) = \max(\max(\text{Alpha}0(\text{imax},\text{jmax}), \text{Alpha}0(\text{imax},\text{jmax}-1)), \text{Alpha}0(\text{imax}-2,\text{jmax})); \]
\[ \text{Alpha\_maxL}(\text{imax},\text{jmax}) = \max(\max(\text{Alpha}0(\text{imax},\text{jmax}), \text{Alpha}0(\text{imax},\text{jmax}-1)), \text{Alpha}0(\text{imax}-2,\text{jmax})); \]

\[ \text{Alpha\_minL}(2:\text{imax}-1,2:\text{jmax}-1) = \min(\min(\min(\min(\text{Alpha}0(2:\text{imax}-1,2:\text{jmax}-1), \text{Alpha}0(3:\text{imax},2:\text{jmax}-1)), \text{Alpha}0(2,1:\text{jmax}-2)))); \]
\% At Boundaries
\[ \text{Alpha\_minL}(1,2:\text{jmax}-1) = \min(\min(\min(\text{Alpha}0(1,2:\text{jmax}-1), \text{Alpha}0(1,3:\text{jmax})), \text{Alpha}0(2,2:\text{jmax}-1)), \text{Alpha}0(1,1:\text{jmax}-2)); \]
\[ \text{Alpha\_minL}(\text{imax},2:\text{jmax}-1) = \min(\min(\min(\text{Alpha}0(\text{imax},2:\text{jmax}-1), \text{Alpha}0(\text{imax},3:\text{jmax})), \text{Alpha}0(\text{imax},1:\text{jmax}-2)); \]
\[ \text{Alpha\_minL}(2:\text{imax}-1,1) = \min(\min(\min(\text{Alpha}0(2:\text{imax}-1,1), \text{Alpha}0(3:\text{imax},1)), \text{Alpha}0(2,1:\text{jmax}-2)), \text{Alpha}0(1,1:\text{jmax}-2)); \]
\[ \text{Alpha\_minL}(2:\text{imax}-1,\text{jmax}) = \min(\min(\min(\text{Alpha}0(2:\text{imax}-1,\text{jmax}), \text{Alpha}0(3:\text{imax},\text{jmax})), \text{Alpha}0(2,\text{imax}-2)), \text{Alpha}0(1,\text{imax}-2)); \]

\% At Corners
\[ \text{Alpha\_minL}(1,1) = \min(\min(\text{Alpha}0(1,1), \text{Alpha}0(1,2)), \text{Alpha}0(2,1)); \]
\[ \text{Alpha\_minL}(\text{imax},1) = \min(\min(\text{Alpha}0(\text{imax},1), \text{Alpha}0(\text{imax},2)), \text{Alpha}0(\text{imax}-1,1)); \]
\[ \text{Alpha\_minL}(\text{imax},\text{jmax}) = \min(\min(\min(\text{Alpha}0(\text{imax},\text{jmax}), \text{Alpha}0(\text{imax},\text{jmax}-1)), \text{Alpha}0(\text{imax}-2,\text{jmax})); \]

\% H) Correcting the local extrema of Alpha
\[ \text{Alpha\_max} = \min(\text{ones(size(Alpha\_maxL))), \text{Alpha\_maxL}); \]
\[ \text{Alpha\_min} = \max(\text{zeros(size(Alpha\_minL))), \text{Alpha\_minL}); \]

\% I) Total inflows and outflows for each cell
\[ F_{\text{pos}} = -(\text{A}_e < 0) \cdot (\text{A}_e + (\text{A}_n < 0) \cdot (\text{A}_n + (\text{A}_w < 0) \cdot (\text{A}_w + (\text{A}_s < 0) \cdot (\text{A}_s))); \]
\[ F_{\text{neg}} = (\text{A}_e > 0) \cdot (\text{A}_e + (\text{A}_n > 0) \cdot (\text{A}_n + (\text{A}_w > 0) \cdot (\text{A}_w + (\text{A}_s > 0) \cdot (\text{A}_s))); \]
% J) Net Flux Per Cell
% Q_pos, Q_neg
Q_pos = (dx*dy)/dt.*(Alpha_max - Alpha0) + (FL_east + FL_west + FL_north + FL_south);
Q_neg = (dx*dy)/dt.*(Alpha0 - Alpha_min) - (FL_east + FL_west + FL_north + FL_south);

% K) Computing the weighting factor for all faces (Lambda)
% Initialize Lambda at the faces to ones
Lambda_e = ones(imax,jmax);
Lambda_w = ones(imax,jmax);
Lambda_n = ones(imax,jmax);
Lambda_s = ones(imax,jmax);
for q = 1:5
% Lambda positive, and Lambda negative
Lambda_pos = max( min( -((A_e < 0).*A_e.*Lambda_e + (A_n < 0).*A_n.*Lambda_n + ...
                          (A_w < 0).*A_w.*Lambda_w + (A_s < 0).*A_s.*Lambda_s) + Q_neg),/ P_neg, ones(imax,jmax)),...
                          zeros(imax,jmax)));
Lambda_neg = max( min( ((A_e > 0).*A_e.*Lambda_e + (A_n > 0).*A_n.*Lambda_n + ...
                          (A_w > 0).*A_w.*Lambda_w + (A_s > 0).*A_s.*Lambda_s + Q_pos),/ P_pos, ones(imax,jmax)),...
                          zeros(imax,jmax)));
% Calculation of new Lambda values
% East Side of Lambda
for i = 1:imax
  for j = 1:jmax-1
    if A_e(i,j) >= 0
      Lambda_e = n(i,j) = min([Lambda_pos(i,j) Lambda_neg(i,j+1)]);
else
    Lambda_e_n(i,j) = \min([\Lambda_{\text{neg}}(i,j) \ \Lambda_{\text{pos}}(i,j+1)])
end
e
end
Lambda_e_n(i,j) = 1;
end
% West Side of Lambda
for i = 1:imax
    for j = 2:jmax
        if A_w(i,j) \geq 0
            Lambda_w_n(i,j) = \min([\Lambda_{\text{pos}}(i,j) \ \Lambda_{\text{neg}}(i,j-1)])
        else
            Lambda_w_n(i,j) = \min([\Lambda_{\text{neg}}(i,j) \ \Lambda_{\text{pos}}(i,j-1)])
        end
    end
Lambda_w_n(i,1) = 1;
end
% North Side of Lambda
for j = 1:jmax
    for i = 1:imax-1
        if A_n(i,j) \geq 0
            Lambda_n_n(i,j) = \min([\Lambda_{\text{pos}}(i,j) \ \Lambda_{\text{neg}}(i+1,j)])
        else
            Lambda_n_n(i,j) = \min([\Lambda_{\text{neg}}(i,j) \ \Lambda_{\text{pos}}(i+1,j)])
        end
    end
Lambda_n_n(imax,j) = 1;
end
% South Side of Lambda
for j = 1:jmax
    for i = 2:imax
        if A_s(i,j) >= 0
            Lambda_s_n(i,j) = min([Lambda_pos(i,j) Lambda_neg(i-1,j)]);
        else
            Lambda_s_n(i,j) = min([Lambda_neg(i,j) Lambda_pos(i-1,j)]);
        end
    end
    Lambda_s_n(1,j) = 1;
end

if sum(sum(abs((Lambda_e_n + Lambda_w_n + Lambda_n_n + Lambda_s_n) -
    (Lambda_e_e + Lambda_w_w + Lambda_n_w + Lambda_s_e)))) < 1e-8
    break
end
Lambda_e = Lambda_e_n;
Lambda_w = Lambda_w_n;
Lambda_n = Lambda_n_n;
Lambda_s = Lambda_s_n;
end % end of q lambda loop

% L) Corrected Flux
F_cor_north = FL_north + Lambda_n*A_n; % North Side Corrected Flux
F_cor_south = FL_south + Lambda_s*A_s; % South Side Corrected Flux
F_cor_east = FL_east + Lambda_e*A_e; % East Side Corrected Flux
F_cor_west = FL_west + Lambda_w*A_w; % West Side Corrected Flux

% M) Update of Alpha
Alpha = Alpha0 - dt/(dx*dy).*((F_cor_east + F_cor_west + F_cor_north + F_cor_south);

% N) Smoothing Alpha
Sig = 0.5; % smoothing sigma
Alpha_smooth(:, :) = imgaussfilt(Alpha(:, :), Sig);
Appendix B

The SIMPLE Algorithm MATLAB Code

```matlab
% Iterate till convergence
diff = Inf;
n = 1;

% (STEP 1): INITIAL GUESS OF VARIABLES: (U, V, P, m_e, m_w, m_n, m_s).
U_star = U_0;
V_star = V_0;
P = P_0;

m_e = rho_e.*u_0(:,2:jmax+1)*dy;
m_w = -rho_w.*u_0(:,1:jmax)*dy;
m_n = rho_n.*v_0(2:imax+1,:)*dx;
m_s = -rho_s.*v_0(1:imax,:)*dx;

% Initialize u, v
u = u_0;                      v = v_0;

% Initializing the prime variables
U_prime = zeros(imax,jmax);
u_e_prime = U_prime;
v_n_prime = U_prime;

% Define coefficients of the NS equations
H_p_u = zeros(imax*jmax,imax*jmax);
H_p_v = zeros(imax*jmax,imax*jmax);
B_p_u = zeros(imax*jmax,1);
B_p_v = zeros(imax*jmax,1);
D_p_uID = zeros(imax*jmax,1);
D_p_v1D = zeros(imax*jmax,1);
G_P_x1D = zeros(imax*jmax,1);
G_P_y1D = zeros(imax*jmax,1);
```
while diff > Tol

    % (STEP 2): SOLVING THE MOMENTUM EQUATIONS for U_star, V_star
    % Calculate the x&y-momentum Coefficients
    for i = 1:imax
        for j = 1:jmax
            a_P = rho(i,j)*dx*dy/dt;  % (Eq. 19)
            % Check if there is an east neighbouring cell
            if j ~= jmax
                a_E = -max([-m_e(i,j), -m_e(i,j)/2 + mu_e(i,j)*dy/dx, 0]);
                a_P = a_P - a_E + m_e(i,j);
            else
                a_P = a_P + 2*mu_e(i,j)*dy/dx;
            end
            % Check if there is a west neighbouring cell
            if j 1
                a_W = -max([-m_w(i,j), -m_w(i,j)/2 + mu_w(i,j)*dy/dx, 0]);
                a_P = a_P - a_W + m_w(i,j);
            else
                a_P = a_P + 2*mu_w(i,j)*dy/dx;
            end
            % Check if there is a north neighbouring cell
            if i 1
                a_N = -max([-m_n(i,j), -m_n(i,j)/2 + mu_n(i,j)*dx/dy, 0]);
                a_P = a_P - a_N + m_n(i,j);
            else
                a_P = a_P + 2*mu_n(i,j)*dx/dy;
            end
            % Check if there is a south neighbouring cell
            if i 1
                a_S = -max([-m_s(i,j), -m_s(i,j)/2 + mu_s(i,j)*dx/dx, 0]);
                a_P = a_P - a_S + m_s(i,j);
            else
                a_P = a_P + 2*mu_s(i,j)*dx/dx;
            end

        end
    end

end
\[
\begin{align*}
a_P &= a_P + 2 \mu_n(i, j) \cdot \frac{dx}{dy}; \\
&\text{end} \\
% \text{Check if there is a south neighbouring cell} \\
\text{if } i \sim= 1 \\
a_S &= -\max\{[-m_s(l, j)], -m_s(l, j)/2 + \mu_s(l, j) \cdot \frac{dx}{dy}, 0]\}; \\
a_P &= a_P - a_S + m_s(l, j); \\
&\text{else} \\
a_P &= a_P + 2 \mu_n(i, j) \cdot \frac{dx}{dy}; \\
&\text{end} \\
b_u &= \rho_0(i, j) \cdot U_0(i, j) \cdot \frac{dx}{dy} / dt + Fx(i, j) \cdot \frac{dx}{dy}; \\
\%
% \text{For } b_\_ \\
\text{if } i == \text{imax} \\
% \text{In this particular boundary we know there is a real value for } v_n, m_n, \text{ and } \mu_n \\
bv &= \rho_0(i, j) \cdot V_0(i, j) \cdot \frac{dx}{dy} / dt + Fy(i, j) \cdot \frac{dx}{dy} \ldots \\
&+ 2 \mu_n(i, j) \cdot v_0(i, j)\cdot \frac{dx}{dy} - m_n(i, j) \cdot v_0(i+1, j); \\
&\text{else} \\
bv &= \rho_0(i, j) \cdot V_0(i, j) \cdot \frac{dx}{dy} / dt + Fy(i, j) \cdot \frac{dx}{dy}; \\
&\text{end} \\
% \text{At last we consider the corners} \\
\text{if } i == 1 \&\& j == 1 \\
b_{p_u}(i, j, j, j) &= a_E / a_E; \\
b_{p_u}(i, j, j, j) &= a_N / a_E; \\
\text{elseif } i == 1 \&\& j == \text{jmax} \\
b_{p_u}(i, j, j, j) &= a_W / a_E; \\
b_{p_u}(i, j, j, j) &= a_N / a_E; \\
\text{elseif } i == \text{imax} \&\& j == \text{jmax} \\
b_{p_u}(i, j, j, j) &= 0; \\
b_{p_u}(i, j, j, j) &= 0;
\end{align*}
\]
% Chapter B. The SIMPLE Algorithm MATLAB Code

H_p_u(imax*(j-1)+i,imax*(j-2)+i) = a_W / a_P;
H_p_u(imax*(j-1)+i,imax*(j-1)+i-1) = a_S / a_P;

elseif i == imax && j == 1
    H_p_u(imax*(j-1)+i,imax*j+i) = a_E / a_P;
    H_p_u(imax*(j-1)+i,imax*(j-1)+i-1) = a_S / a_P;

% Then we consider the boundaries

elseif i == 1
    H_p_u(imax*(j-1)+i,imax*j+i) = a_E / a_P;
    H_p_u(imax*(j-1)+i,imax*(j-2)+i) = a_W / a_P;
    H_p_u(imax*(j-1)+i,imax*(j-1)+i+1) = a_N / a_P;

elseif i == imax
    H_p_u(imax*(j-1)+i,imax*j+i) = a_E / a_P;
    H_p_u(imax*(j-1)+i,imax*(j-2)+i) = a_W / a_P;
    H_p_u(imax*(j-1)+i,imax*(j-1)+i-1) = a_S / a_P;

elseif j == 1
    H_p_u(imax*(j-1)+i,imax*j+i) = a_E / a_P;
    H_p_u(imax*(j-1)+i,imax*(j-1)+i+1) = a_N / a_P;
    H_p_u(imax*(j-1)+i,imax*(j-1)+i-1) = a_S / a_P;

elseif j == jmax
    H_p_u(imax*(j-1)+i,imax*(j-2)+i) = a_W / a_P;
    H_p_u(imax*(j-1)+i,imax*(j-1)+i+1) = a_N / a_P;
    H_p_u(imax*(j-1)+i,imax*(j-1)+i-1) = a_S / a_P;

% At last, the general case

else
    H_p_u(imax*(j-1)+i,imax*j+i) = a_E / a_P;
\[ H_{p_u}(i_{max}*(j-1)+i, i_{max}*(j-2)+i) = \frac{a_W}{a_P}; \]
\[ H_{p_u}(i_{max}*(j-1)+i, i_{max}*(j-1)+i+1) = \frac{a_N}{a_P}; \]
\[ H_{p_u}(i_{max}*(j-1)+i, i_{max}*(j-1)+i-1) = \frac{a_S}{a_P}; \]
end

% The vector \textbf{B}_p for each
\[ B_{p_u}(i_{max}*(j-1)+i) = \frac{bu}{a_P}; \]
\[ B_{p_v}(i_{max}*(j-1)+i) = \frac{bv}{a_P}; \]

% The vector \textbf{D}_p for each
\[ D_{p_u}(i_{max}*(j-1)+i) = \frac{dx*dy}{a_P}; \]
\[ D_{p_v}(i_{max}*(j-1)+i) = \frac{dy*dx}{a_P}; \]

% Calculation of the pressure gradient
\textbf{if} \ j == j_{max}
\[ \text{grad}_P_x = (P(i, j) - P(i, j-1))/dx; \]
\textbf{else if} \ j == 1
\[ \text{grad}_P_x = (P(i, j+1) - P(i, j))/dx; \]
\textbf{else}
\[ \text{grad}_P_x = (P(i, j+1) - P(i, j-1))/2/dx; \]
\textbf{end}
% At the upper (surface)
\textbf{if} \ i == i_{max}
\[ \text{grad}_P_y = (2*P_{in} - P(i, j) - P(i-1, j))/2/dy; \]
else
    grad_P_y = (P(i+1,j) - P(i-1,j))/2/dy;
end

% Calculation of the pressure gradient in the vector form
G_P_x1D(imax*(j-1)+i) = grad_P_x;
G_P_y1D(imax*(j-1)+i) = grad_P_y;
end
end

% The coefficient matrix is the same for both x & y momentums
H_p_v = H_p_u;

% Solve the x-momentum equation
II = eye(imax*jmax);
U_star1D = (II + H_p_u) \ (-D_p_u1D.*G_P_x1D + B_p_u);

% Solve the y-momentum equation
V_star1D = (II + H_p_v) \ (-D_p_v1D.*G_P_y1D + B_p_v);

% Reevaluate the original 2D domain
U_star = reshape(U_star1D,imax,[]);
V_star = reshape(V_star1D,imax,[]);
G_P_x = reshape(G_P_x1D,imax,[]);
G_P_y = reshape(G_P_y1D,imax,[]);
D_p_u = reshape(D_p_u1D,imax,[]);
D_p_v = reshape(D_p_v1D,imax,[]);
% (STEP 3) USING RHEI_CHOW INTERPOLATION TO CALCULATE MASS FLOW RATE AT THE FACES

% East-face velocity using linear interpolation
u_e_st_bar = u(:,2:jmax+1); u_e_star = u_e_st_bar;
u_e_st_bar(:,1:jmax-1) = 0.5*(U_star(:,1:jmax-1) + U_star(:,2:jmax));
% East-face pressure gradient using linear interpolation
G_P_e_bar = 0.5*(G_P_x(:,1:jmax-1) + G_P_x(:,2:jmax));
G_P_e = (P(:,2:jmax) - P(:,1:jmax-1))/dx;
% Pressure coefficient at the east face
D_ue_bar = D_p_u;
D_ue_bar(:,1:jmax-1) = 0.5*(D_p_u(:,1:jmax-1) + D_p_u(:,2:jmax));
% Rhie-chow equation
u_e_star(:,1:jmax-1) = u_e_st_bar(:,1:jmax-1) - D_ue_bar(:,1:jmax-1).*(G_P_e - G_P_e_bar);
m_e_star = rho_e.*u_e_star*dy;

% West-face velocity using linear interpolation
u_w_st_bar = u(:,1:jmax); u_w_star = u_w_st_bar;
u_w_st_bar(:,2:jmax) = 0.5*(U_star(:,2:jmax) + U_star(:,1:jmax-1));
% West-face pressure gradient using linear interpolation
G_P_w_bar = 0.5*(G_P_x(:,1:jmax-1) + G_P_x(:,2:jmax));
G_P_w = (P(:,1:jmax-1) - P(:,2:jmax))/(dx);
% Pressure coefficient at the west face
D_uw_bar = D_p_u;
D_uw_bar(:,2:jmax) = 0.5*(D_p_u(:,1:jmax-1) + D_p_u(:,2:jmax));
% Rhie-chow equation
u_w_star(:,2:jmax) = u_w_st_bar(:,2:jmax) - D_uw_bar(:,2:jmax).*G_P_w - G_P_w_bar);
m_w_star = -rho_w.*u_w_star*dy;
% North-face velocity using linear interpolation
v_n_st_bar = v(2:imax, :) - v_n_star = v_n_st_bar;

v_n_st_bar(1:imax-1,:) = 0.5*(V_star(1:imax-1,:) + V_star(2:imax,:));

% North-face pressure gradient using linear interpolation
G_P_n_bar = 0.5*(G_P_y(1:imax-1,:) + G_P_y(2:imax,:));

G_P_n = (P(2:imax,:) - P(1:imax-1,:))/dy;

% Pressure coefficient at the north face
D_vn_bar = D_p_v;

D_vn_bar(1:imax-1,:) = 0.5*(D_p_v(1:imax-1,:) + D_p_v(2:imax,:));

% Rhie-chow equation
v_n_star(1:imax-1,:) = v_n_st_bar(1:imax-1,:) - D_vn_bar(1:imax-1,:).*\(G_P_n - G_P_n_bar\)

m_n_star = rho_n.*v_n_star*dy;

% South-face velocity using linear interpolation
v_s_st_bar = v(1:imax,:); v_s_star = v_s_st_bar;

v_s_st_bar(2:imax,:) = 0.5*(V_star(2:imax,:) + V_star(1:imax-1,:));

% South-face pressure gradient using linear interpolation
G_P_s_bar = 0.5*(G_P_y(2:imax,:) + G_P_y(1:imax-1,:));

G_P_s = (P(1:imax-1,:) - P(2:imax,:))/(-dy);

% Pressure coefficient at the south face
D_vs_bar = D_p_v;

D_vs_bar(2:imax,:) = 0.5*(D_p_v(2:imax,:) + D_p_v(1:imax-1,:));

% Rhie-chow equation
v_s_star(2:imax,:) = v_s_st_bar(2:imax,:) - D_vs_bar(2:imax,:).*\(G_P_s - G_P_s_bar\)

m_s_star = -rho_s.*v_s_star*dy;
% (STEP 4): SOLVING THE PRESSURE CORRECTION EQUATION
% Initialize the coefficient matrix where Pressure distribution in a 1D vector form
H_p_P = zeros(imax*jmax,imax*jmax);
B_p_P = zeros(imax*jmax,1);
A_p_P = zeros(imax*jmax,imax*jmax);

% Calculate the Pressure Correction coefficients
for i = 1:imax
    for j = 1:jmax
        a_P = 0;
        b_P = (rho(i,j) - rho_0(i,j))*dx*dy/dt;
        if j == jmax
            a_E = rho_e(i,j)*D_ue_bar(i,j)*dy/dx;
            a_P = a_P - a_E;
            b_P = b_P + m_e_star(i,j);
        else
            a_E = 0;
        end
        if j == 1
            a_W = rho_w(i,j)*D_uw_bar(i,j)*dy/dx;
            a_P = a_P - a_W;
            b_P = b_P + m_w_star(i,j);
        else
            a_W = 0;
        end
        if i == imax
            a_N = rho_n(i,j)*D_vn_bar(i,j)*dx/dy;
        end
    end
end
\[ a_{N} = \text{rho}_n(i,j) \times D_{vn \text{ bar}}(i,j) \times dx/dy; \]

\[ a_{P} = a_{P} - a_{N}; \]

\[ b_{P} = b_{P} + m_{n \text{ star}}(i,j); \]

\[ \text{else} \]

\[ a_{N} = \text{rho}_n(i,j) \times D_{vn \text{ bar}}(i,j) \times dx/(0.5 \times dy); \]

\[ a_{P} = a_{P} - a_{N}; \]

\[ b_{P} = b_{P} + m_{n \text{ star}}(i,j); \]

\[ \text{end} \]

\[ \text{if} \ i = 1 \]

\[ a_{S} = \text{rho}_s(i,j) \times D_{vs \text{ bar}}(i,j) \times dx/dy; \]

\[ a_{P} = a_{P} - a_{S}; \]

\[ b_{P} = b_{P} + m_{s \text{ star}}(i,j); \]

\[ \text{else} \]

\[ a_{S} = \text{rho}(i,j) \times D_{vs \text{ bar}}(i,j) \times dx/(0.5 \times dy); \]

\[ a_{P} = a_{P} - a_{S}; \]

\[ \text{end} \]

\[ \% \text{Calculating } H_{p}, B_{p} \text{ for Prime Pressure} \]

\[ \% \text{First we consider corners} \]

\[ \text{if} \ i = 1 \&\& \ j = 1 \]

\[ H_{p \ P}(i,j-1) + i, i\max \times (j+1) = a_{E}; \]

\[ H_{p \ P}(i,j-1) + i, i\max \times (j-1) = a_{N}; \]

\[ \text{elseif} \ i = 1 \&\& \ j = j\max \]

\[ H_{p \ P}(i,j-1) + i, i\max \times (j-2) + i = a_{W}; \]

\[ H_{p \ P}(i,j-1) + i, i\max \times (j-1) + i+1 = a_{N}; \]

\[ \text{elseif} \ i = i\max \&\& \ j = j\max \]

\[ H_{p \ P}(i,j-1) + i, i\max \times (j-2) + i = a_{W}; \]

\[ H_{p \ P}(i,j-1) + i, i\max \times (j-1) + i-1 = a_{S}; \]
elseif i == imax && j == jmax
    \( H_{pP}(i_{n-1}+i, i_{n-2}+i) = a_W \)
    \( H_{pP}(i_{n-1}+i, i_{n-1}+i-1) = a_S \)
elseif i == imax && j == 1
    \( H_{pP}(i_{n-1}+i, i_{n-2}+i) = a_E \)
    \( H_{pP}(i_{n-1}+i, i_{n-1}+i-1) = a_S \)

% Then we consider the boundaries
elseif i == 1
    \( H_{pP}(i_{n-1}+i, i_{n-2}+i) = a_E \)
    \( H_{pP}(i_{n-1}+i, i_{n-2}+i) = a_W \)
    \( H_{pP}(i_{n-1}+i, i_{n-1}+i+1) = a_N \)
elseif j == imax
    \( H_{pP}(i_{n-1}+i, i_{n-2}+i) = a_E \)
    \( H_{pP}(i_{n-1}+i, i_{n-2}+i) = a_W \)
    \( H_{pP}(i_{n-1}+i, i_{n-1}+i-1) = a_S \)
elseif j == 1
    \( H_{pP}(i_{n-1}+i, i_{n-2}+i) = a_E \)
    \( H_{pP}(i_{n-1}+i, i_{n-1}+i) = a_N \)
    \( H_{pP}(i_{n-1}+i, i_{n-1}+i-1) = a_S \)
elseif j == jmax
    \( H_{pP}(i_{n-1}+i, i_{n-2}+i) = a_W \)
    \( H_{pP}(i_{n-1}+i, i_{n-1}+i) = a_N \)
    \( H_{pP}(i_{n-1}+i, i_{n-1}+i-1) = a_S \)

% At last, the general case
else
\begin{verbatim}
H_p_P(imax*(j-l)+i,imax*j+l) = a_E;
H_p_P(imax*(j-l)+i,imax*(j-2)+i) = a_W;
H_p_P(imax*(j-l)+i,imax*(j-l)+i) = a_N;
H_p_P(imax*(j-l)+i,imax*(j-l)+i-1) = a_S;
end

B_p_P(imax*(j-l)+i) = b_P;
A_p_P(imax*(j-l)+i,imax*(j-l)+i) = a_P;
end

% Solve the Pressure Correction equation
L_p_P = A_p_P + H_p_P;
P_prime1D = L_p_P \ B_p_P;

% Reevaluate the Pressure field in 2D
P_prime = reshape(P_prime1D,imax,[]);

% (STEP 5): UPDATE THE VELOCITIES & PRESSURE FIELDS
% Calculate Pressure Gradients for prime pressure at x-dir
G_P_prime_x = 0.*G_P_x;
G_P_prime_x(:,2:jmax-1) = (P_prime(:,3:jmax) - P_prime(:,1:jmax-2))./2/dx;
G_P_prime_x(:,jmax) = (P_prime(:,jmax)-P_prime(:,jmax-1))/dx;
G_P_prime_x(:,1) = (P_prime(:,2) - P_prime(:,1))/dx;
\end{verbatim}
% Calculate Pressure Gradients for prime pressure at y-dir
G_P_prime_y = 0.*G_P_y;
G_P_prime_y(2:imax-1,:) = (P_prime(3:imax,:) - P_prime(1:imax-2,:))./2/dy;
G_P_prime_y(imax,:) = (- P_prime(imax,:) - P_prime(imax-1,:))/2/dy;
G_P_prime_y(1,:) = (P_prime(2,:) - P_prime(1,:))/dy;

% Calculate Pressure gradients at faces
G_P_p_e = (P_prime(:,2:jmax)-P_prime(:,1:jmax-1))./dx;
G_P_p_w = (P_prime(:,1:jmax-1)-P_prime(:,2:jmax))./(-dx);
G_P_p_n = (P_prime(2:imax,:) - P_prime(1:imax-1,:))./dy;
G_P_p_s = (P_prime(1:imax-1,:) - P_prime(2:imax,:))./(-dy);
% Calculate U_prime, V_prime, m_f_prime
U_prime = - D_p_u .* G_P_prime_x;
V_prime = - D_p_v .* G_P_prime_y;
% Calculate u_prime, v_prime at faces
u_e_prime(:,1:jmax-1) = -D_ue_bar(:,1:jmax-1).*G_P_p_e;
u_w_prime(:,2:jmax) = -D_uw_bar(:,2:jmax).*G_P_p_w;
v_n_prime(1:imax-1,:) = -D_vn_bar(1:imax-1,:).*G_P_p_n;
v_s_prime(2:imax,:) = -D_vs_bar(2:imax,:).*G_P_p_s;

% Calculate the prime mass flow rate at faces
m_e_prime = rho_e .* u_e_prime * dy;
m_w_prime = -rho_w .* u_w_prime * dy;
m_n_prime = rho_n .* v_n_prime * dx;
m_s_prime = -rho_s .* v_s_prime * dx;
% Calculate the prime mass flow rate at faces
m_e_prime = rho_e .* u_e_prime * dy;
m_w_prime = -rho_w .* u_w_prime * dy;
m_n_prime = rho_n .* v_n_prime * dx;
m_s_prime = -rho_s .* v_s_prime * dx;

% CORRECT THE PRESSURE
P_star = P + alpha_P.* P_prime;

% UPDATE THE OTHER VARIABLES
U_star = U_star + U_prime;
V_star = V_star + V_prime;
m_e = m_e_star + m_e_prime;
m_w = m_w_star + m_w_prime;
m_n = m_n_star + m_n_prime;
m_s = m_s_star + m_s_prime;

for i = 1:imax
    for j = 2:jmax
        u(i,j) = (U_star(i,j-1) + U_star(i,j))/2;
    end
end
for j = 1:jmax
    for i = 2:imax
        v(i,j) = (V_star(i-1,j) + V_star(i,j))/2;
    end
end

% CALCULATE THE DIFFERENCE BETWEEN NEW AND OLD STEPS
diff = max(max(abs(P - P_star)));

% (STEP 6): TREAT NEW PRESSURE AS THE P_STAR OF THE NEXT ITERATION
P = P_star;

% NEW ITERATION
n = n + 1;
end
U = U_star;
V = V_star;