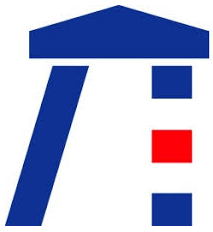


Numerical Analysis of Free Surface Flow and the Application to Ship Design

Master Thesis

by

Robabeh Izadshenas

Technische Universität Kaiserslautern
Fachbereich Mathematik

Supervisors: Prof. Dr. Bernd. Simeon

Acknowledgements

I would like to express my sincere appreciation to my supervisor Mr.Prof.Bernd Simeon for his helpful guidance, interesting topic and his support during the project. He was always there when I needed feedback and new ideas. I really appreciate his kindness and his support during this project. I also would like to express my gratitude to my family, they have always had my back, and I am thankful for their unconditional support.

Contents

List of Symbols **iv**

1 Introduction **1**

2 Mathematical Modelling **3**

2.1 Mathematical Equations 3

2.2 Continuity Equation 4

2.3 Momentum Equation 6

2.4 Free Surface Flow 9

2.5 Projection Method 9

2.6 Governing Equation 10

2.7 Computational Mesh 10

2.8 Discretization 10

3 VOF Method **12**

3.1 Interface Tracking Methods 12

3.2 Interface capturing Methods 13

3.2.1 VOF Method 14

3.2.2 Geometric VOF Method 15

3.2.3 VOF Advection 23

3.2.4 Rotation of Slotted Disk 24

3.2.5 Summery 25

4 OpenFOAM **27**

4.1 Turbulence Model: $k-\omega$ SST 27

4.2 OpenFOAM 28

4.2.1 Creating Mesh 28

4.2.2 Initialization of the Boundary 29

4.2.3 Phsical Properties 30

4.2.4 Numerical Solvers and Algorithms 31

4.2.5 Stability 37

4.2.6 Forces, Post-processing 37

4.2.7 Information of Resources 37

5 Results **39**

5.1 Physical Domain and Boundary Conditions 39

5.2 Results and Discussion 40

Bibliography **48**

A OpenFoam Dictionary **53**

A.1 interDyMFoam solver 53

A.2 interFlow solver 62

A.3 isoAvector 66

List of Symbols

v	Velocity
ρ	Density
F, α	Volume Fraction Function
ν	Viscosity
$\frac{\partial u}{\partial t}$	Derivative of velocity w.r.t time
p	Pressure
P_{rgh}	Modified pressure
k	Turbulent Kinetic energy
ω	Eddy Dissipation Rate
m	Mass
n	Normal Vector
C	Courant Number
I	Turbulent intensity
dt	Time step
div	Divergence
∇	Gradient

Chapter 1

Introduction

Computational Fluid Dynamics (CFD) is a part of fluid mechanics which is the combination of numerical methods, mathematical modeling and software tools to solve and analyze problems related to fluid flows. In principle, CFD helps to design comfortable and safe living environments by predicting that how to prevent natural disasters in our daily life by simulating the fluid flow around vehicles like aircraft, car or ship under different situations. The main basis of CFD problems are the Navier–Stokes equations. Therefore, CFD allows the fluid mass flow, velocity, density or any fluid property to be calculated at all locations within the flow field. The interaction of the fluid with its solid boundaries, such as fluid-dynamic forces can also be estimated.

Numerical simulations of multi-phase flow have been investigated in variety range of industrial level such as petroleum refining, biological flows and interaction of air with the sea surface [1]. Multi-phase flow simulation consists of two or more discrete phases segregated with proper interface. To solve such a problem with numerical simulation, keeping the sharp interface during fluid transportation is a difficult task in the modelling of interfacial flows [2] The interface between two phases can be modelled by a scalar transport equation [3]. The modelling consists of construction and movement of the interface. In this thesis, namely, a Volume of Fluid (VOF) method has been presented which consists of VOF method. Valume of fluid method is based on interface capturing approach which is an effective approach for interface modelling. Although it has has two steps: reconstruction and advection of the interface between two fluids [4].First, the interface is defined by calculating the volume fractions of each fluid. Then a transportation algorithm is employed for the movement of the interface .The main challenge in the modelling of the interfacial fluid flow is the implementation of a method which can effciently move the sharp interface without stretching and wrinkling [5].[4]. In the reconstruction step, the interface is approximated by a straight line. Next, the area of the geometrical shape (represented by triangles or rectangles) below the lines is calculated to evaluate the volume of fluid based on

the position of the interface. The interface is approximated in the subsequent time steps by using the volume of fluid data of the previous time step and this is where the main difficulty (maintaining the sharpness of the interface) in the interface approximation arises [4]. In this thesis, an analytical relation between the fractional volumes and interface position has been implemented. In this work, first of all the mathematical equations are defined in order to get Navier Stokes equations and how we can discretize them. After that, the whole process of VOF method like how to generate the interface and how to calculate the flux in each computational cell are presented and finally we use the OpenFoam technology to investigate the application of VOF method for a physical problem like ship with different solvers and algorithm to solve the Navier-Stokes equations in combination with volume fraction function.

Chapter 2

Mathematical Modelling

This chapter provides the initial requirements of our purpose like Navier-Stokes equations.

2.1 Mathematical Equations

The properties of the fluid flow (such as velocity and pressure) are governed by the continuity and momentum equations which are the mathematical interpretations of the mass and momentum conservation principles [6]. These governing equations are represented by a set of PDEs as follows [7].

- Continuity Equation:The conservative continuity equation for compressible fluid is given by

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho u) = 0 \tag{2.1}$$

  where t stands for time, ρ denotes density, u is the velocity vector and ∇ represents the nabla operator.

- Momentum Equation: The transportation of fluid is described by the momentum equation as follows

$$\rho(\frac{\partial u}{\partial t} + u.\nabla u) = -\nabla p + \nabla.T + f \tag{2.2}$$

  where T stress tensor and f is sum of external force.

This set of PDEs represents the flow of each viscous fluids and are known as the Navier-Stokes (N-S) equations [8]. These non-linear and coupled equations are difficult to solve analytically, and numerical solutions are generally used for simulation.

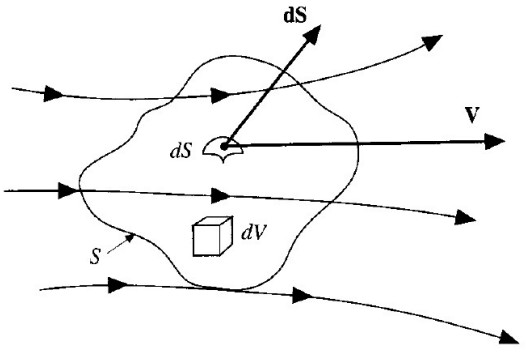


Figure 2.1: Finite control volume fixed in space

2.2 Continuity Equation

The continuity equation certifies conservation of mass, to get better intuition about this physical principle which is as follows[6]

Consider a control volume as figure 2.1 with the flow velocity **V** and the vector elemental surface area **dS** on the control surface. Also, $d\mathcal{V}$ be an elemental volume inside the finite control volume. Exerted mass is conserved to this control volume means that Net mass flow out of control volume through surface S is equal to time rate of decrease of mass inside control volume. To make it easier consider B as the net mass flow out of control volume through surface S and C as the time rate of decrease of mass inside control volume as below

$$B = C \tag{2.3}$$

Since the mass flow of a moving fluid across any fixed surface is equal to the product of density × area of surface × component of velocity perpendicular to the surface, therefore, the elemental mass flow across the area dS is

$$\rho V_n dS = \rho \mathbf{V}.\mathbf{dS} \tag{2.4}$$

The figure 2.1 shows that by convection, **dS** always points in a direction out of the control volume. Hence, when **V** also points out of control volume the product ρ**V**.**dS** is positive. Moreover, when **V** points out of the control volume, the mass flow is physically leaving the control volume; i.e it is an outflow. Hence, a positive ρ**V**.**dS** denotes an outflow. In contrast, when **V** points inward, the mass flow is physically entering the control volume; i.e it is an inflow. Hence a negative ρ**V**.**dS** denotes an inflow. The net mass flow out of the entire control volume through the control surface S is the summation over S of the elemental mass flow expressed in 2.4. In the limit, this becomes a surface integral, which is physically B the net mass flow out of the control volume through

surface S, that is,

$$B = \iint_S \rho \mathbf{V}.\mathbf{dS} \tag{2.5}$$

On the other hand, the total mass contained within the elemental control volume $d\mathcal{V}$ is $\rho d\mathcal{V}$. therefore, the total mass inside the control volume and the time rate of increase of mass inside $\mathcal{V}$ are $\iiint_{\mathcal{V}} \rho d\mathcal{V}$ and $\frac{\partial}{\partial t}\iiint_{\mathcal{V}} \rho d\mathcal{V}$, respectively. Obviusly, the time rate of decrease of mass inside $\mathcal{V}$ is the negative of above which is

$$-\frac{\partial}{\partial t}\iiint_{\mathcal{V}} \rho d\mathcal{V} = C \tag{2.6}$$

Thus, substituting 2.5 and 2.6 into2.3, we get

$$\iint_S \rho \mathbf{V}.\mathbf{dS} = -\frac{\partial}{\partial t}\iiint_{\mathcal{V}} \rho d\mathcal{V} \tag{2.7}$$

or

$$\frac{\partial}{\partial t}\iiint_{\mathcal{V}} \rho d\mathcal{V} + \iint_S \rho \mathbf{V}.\mathbf{dS} = 0 \tag{2.8}$$

This is the integral form of continuity equation. Since the control volume used for the derivation of 2.8 is fixed in space, the limits of integration for the integrals 2.8 are constant. and hence the time derivative $\frac{\partial}{\partial t}$ can be placed inside the integral.

$$\iiint_{\mathcal{V}} \frac{\partial \rho}{\partial t} \rho d\mathcal{V} + \iint_S \rho \mathbf{V}.\mathbf{dS} = 0 \tag{2.9}$$

Applying the divergence theorem, the surface integral in 2.9 can be interpret as a volume integral:

$$\iint_S (\rho \mathbf{V}).\mathbf{dS} = \iiint_{\mathcal{V}} \nabla.(\rho \mathbf{V}) d\mathcal{V} \tag{2.10}$$

Substituing 2.10 into 2.9 , we get

$$\iiint_{\mathcal{V}} \frac{\partial \rho}{\partial t} d\mathcal{V} + \iiint_{\mathcal{V}} \nabla.(\rho \mathbf{V}) d\mathcal{V} = 0 \tag{2.11}$$

or

$$\iiint_{\mathcal{V}} \left[\frac{\partial \rho}{\partial t} + \nabla.(\rho \mathbf{V}) d\mathcal{V} \right] = 0 \tag{2.12}$$

Since the integral in 2.12 to be zero is for the integrand to be zero at every point within the control volume. Therefore, we have the continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho \mathbf{V}) = 0 \tag{2.13}$$

One has to know that

$$\nabla.(\rho \mathbf{V}) \equiv (\rho \nabla.\mathbf{V}) + (\mathbf{V}.\nabla \rho) \tag{2.14}$$

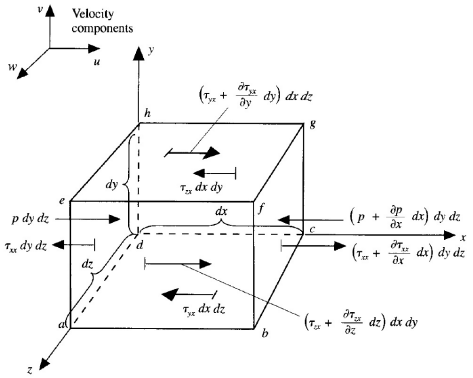


Figure 2.2: Infinitesimally small, moving element

And substitutional derivative, which physically means that time rate of change following a moving fluid element is define as

$$\frac{D}{Dt} \equiv \underbrace{\frac{\partial}{\partial t}}_{\text{Local derivative}} + \underbrace{(\mathbf{V}.\nabla)}_{\text{Convective derivative}} \tag{2.15}$$

Physically which consist of local derivative, the time rate of change at fixed point and convective derivative, the time rate of change due to the movement of the fluid element from one location to another in the flow field where the flow properties are specially different.

2.3 Momentum Equation

The momentum equation is equivalent to another physical principle namely, Newton's second law of motion $F = ma$. Figure 2.2 represents that net force on the fluid element equals its mass times the acceleration of the element. Only the forces in x direction are shown.

By considering only the $x$component of Newton's second law,

$$F_x = ma_x \tag{2.16}$$

F_x and a_x are the scalar x components of the force and acceleration, respectively. To get deep inside the forces, there are two sources:

- Body forces, which act directly on the volumetric mass of the fluid element. These forces " act at distance"; examples are gravitational electric, and magnetic forces. The Body force per unit mass acting on the fluid element denotes by f, which f_x is its x component with the volume of the fluid element $(dxdydz)$. Therefore, Body force on fluid element acting in x direction is

$$\rho f_x(dxdydz) \tag{2.17}$$

- Surface forces, which act directly on the surface of fluid element. They are due to only two sources: (a) the pressure distribution acting on the surface, imposed by the outside fluid surrounding the fluid element, and (b) the shear and normal stress distributions acting on the surface, also imposed by outside fluid "tugging" or pushing on surface by means of friction. The shear and normal stresses in fluid are related to the time rate of change of the deformation of the fluid element as shown in figure 2.2 only for the xy plane are τ_{xy} and τ_{xx}, respectively. Considering all surface force, pressure force, shear and normal stress, the net force in x direction is as follows[6]:

$$\begin{aligned} Surface force = & \left[p - \left(p + \frac{\partial p}{\partial x} dx \right) \right] dydz \\ & + \left[\left(\tau_{xx} + \frac{\partial \tau_{xx}}{\partial x} dx \right) - \tau_{xx} \right] dydz \\ & + \left[\left(\tau_{yx} + \frac{\partial \tau_{yx}}{\partial y} dy \right) - \tau_{yx} \right] dxdz \\ & + \left[\left(\tau_{zx} + \frac{\partial \tau_{zx}}{\partial z} dz \right) - \tau_{zx} \right] dxdy \end{aligned} \tag{2.18}$$

The total force in x direction is given by the sum of 2.17 and 2.18 as follows:

$$F_x = \left[-\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} \right] dxdydz + \rho f_x dxdydz \tag{2.19}$$

Considering the right hand side of the Eq.2.16, the mass of the fluid element is fixed and is equal to

$$m = \rho dxdydz \tag{2.20}$$

And the acceleration is the time rate of the velocity denoted by a_x which is simply the time rate of u, therefore

$$a_x = \frac{Du}{Dt} \tag{2.21}$$

and we get

$$\rho \frac{Du}{Dt} = \left[-\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} \right] + \rho f_x \tag{2.22}$$

Simoultanesly we can write it for y and z direction. to extend it in conservation form we do as follows:

$$\rho \frac{Du}{Dt} = \rho \frac{\partial u}{\partial t} + \rho \mathbf{V}.\nabla u \tag{2.23}$$

by expending the derivative

$$\frac{\partial \rho u}{\partial t} = \rho \frac{\partial u}{\partial t} + u \frac{\partial \rho}{\partial t} \tag{2.24}$$

and

$$\nabla.(\rho u\mathbf{V}) = u\nabla.(\rho\mathbf{V}) + (\rho\mathbf{V}).\nabla u \tag{2.25}$$

by considering $\rho\frac{\partial(u)}{\partial t} = \frac{\partial(\rho u)}{\partial t} - u\frac{\partial\rho}{\rho t}$, $\rho V.\nabla u = \nabla.(\rho uV) - u\nabla.(\rho V)$ and substitute them into 2.23

$$\begin{aligned}\rho\frac{Du}{Dt} &= \frac{\partial(\rho u)}{\partial t} - u\frac{\partial\rho}{\rho t} - u\nabla.(\rho\mathbf{V}) + \nabla.(\rho u\mathbf{V}) \\ &= \frac{\partial(\rho u)}{\partial t} - u\left[\frac{\partial\rho}{\partial t} + \nabla.(\rho\mathbf{V})\right] + \nabla.(\rho u\mathbf{V})\end{aligned} \tag{2.26}$$

Since the big bracket is the continuity equation hence this term is zero and we have

$$\rho\frac{Du}{Dt} = \frac{\partial(\rho u)}{\partial t} + \nabla.(\rho u\mathbf{V}) \tag{2.27}$$

Substitute eq.2.27 into eq.2.22

$$\frac{\partial(\rho u)}{\partial t} + \nabla.(\rho u\mathbf{V}) = \left[-\frac{\partial p}{\partial x} + \frac{\partial\tau_{xx}}{\partial x} + \frac{\partial\tau_{yx}}{\partial y} + \frac{\partial\tau_{zx}}{\partial z}\right] + \rho f_x \tag{2.28}$$

$$\tau_{xx} = \lambda(\nabla.\mathbf{V}) + 2\mu\frac{\partial u}{\partial x} \tag{2.29}$$

$$\tau_{xy} = \tau_{yx} = \mu\left[\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right] \tag{2.30}$$

$$\tau_{xz} = \tau_{zx} = \mu\left[\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right] \tag{2.31}$$

Where μ is molecular viscosity coefficient and λ is the second viscosity coefficient which has been considering is $\lambda = -\frac{2}{3}\mu$. By the general form of the momentum equation 2.2, the stress tenssor for the Newtonian fluid can be interpret [9]

$$T = -(p + \frac{2}{3}\mu\nabla.u)I + 2\mu D \tag{2.32}$$

where I is the unit tensor, D is the rate of the strain (deformation) tensor For a Newtonian fluid the viscous stresses are proportional to the rates of deformation, yielding the following stress tensor formulation

$$D = \frac{1}{2}((\nabla u) + (\nabla u)^T) \tag{2.33}$$

Therefore, By the equations 2.2,2.33 the consequence of the incompressible Navier-Stokes equation for the momentum transport of a Newtonian flow has the form:

$$\frac{\partial u}{\partial t} + u.\nabla\mathbf{V} = \frac{-1}{\rho}\nabla P + \frac{1}{\rho}\nabla.\mu((\nabla u) + (\nabla u)^T) + f \tag{2.34}$$

2.4 Free Surface Flow

Since only the first position of the interface is known, the position of the interface in latter times has to be figured out. Therefore, the boundary conditions at the interface can be derived by applying the conservation laws to the interface where conservation of momentum and mass must be confirmed. The conservation of mass at the interface provides known kinematic condition

$$\rho_1(u_1 - u_I).\hat{n} = \rho_2(u_2 - u_I).\hat{n} = 0 \tag{2.35}$$

Where subscript 1 and 2 refer to phase 1 and phase 2, and u_I equation 2.35 states that the normal component of the velocity of the fluid phases at the interface are equal to the normal component of the velocity of the interface. This implies that mass flux across the interface is zero and that the interface is a sharp boundary separating the two fluids[9]. Conservation of momentum at the interface, or the so-called dynamic condition, requires forces at the interface to be at equilibrium [9].

$$(pI + 2\mu D)_1.\hat{n} - (pI + 2\mu D)_2.\hat{n} = (\sigma k - \nabla_S \sigma).\hat{n} \tag{2.36}$$

where $\hat{n}$ is the unit normal to the interface, σ is the surface tension, ∇_S is the surface gradient, and k is the local mean curvature of the interface has the interpretation

$$K = \nabla.\hat{n} \tag{2.37}$$

Equation 2.36 consist of normal and tangential components where the normal component is has the interpretation

$$p_1 - p_2 = \hat{n}.(\mu_1 D_1 - \mu_2 D_2).\hat{n} + k\sigma \tag{2.38}$$

And the tangential component reads

$$\nabla_S \sigma = \hat{n} \times (\mu_1 D_1 - \mu_2 D_2).\hat{n} \tag{2.39}$$

2.5 Projection Method

The projection method is a way of solving an incompressible time dependent fuid flow problems. Originally introduced by Alexandre Chorin in 1967[10] as an efficient means of solving the incompressible Navier-Stokes equations. The power of the projection method is that the computations of the velocity and the pressure fields are separated.

2.6 Governing Equation

The incompressible Navier-Stokes equations can be written as

$$\frac{\partial u}{\partial t} + u.\nabla u = -\frac{1}{\rho}\nabla p + \nu\nabla^2 u \tag{2.40}$$

$$\nabla.u = 0 \tag{2.41}$$

2.7 Computational Mesh

The governing equations are solved on a computational mesh. The mesh used in this document is uniform with mesh cells of width Δx and height, Δy. The grid divides the domain in to $nx \times ny$ cells where nx and ny are the number of cells in the x and y directions, respectively. A staggered grid is used to store the variables where the pressure is stored at the cell center and the velocities are stored at the cell faces [11].

2.8 Discretization

Discretise the momentum equation with respect to time using an explicit scheme (forward difference). In an explicit scheme, the following approximates the time derivative:

$$\frac{u^{n+1} - u^n}{\Delta t} = \frac{-1}{\rho}\nabla p^{n+1} - u^n.\nabla u^n + \nu\nabla^2 u^n \tag{2.42}$$

To introduce continuity we solve this equation using the predictor-corrector or fractional step methodology. In this framework the Navier-Stokes equations are solved in two steps. The first step, known as the predictor step, is to compute an intermediate velocity u^* by solving the momentum equation but omitting the effect of pressure„

$$\frac{u^* - u^n}{\Delta t} = -u^n.\nabla u^n + \nu\nabla^2 u^n \tag{2.43}$$

The second step, known as the corrector step, is to solve for the new velocity u^{n+1} and include the influence of the pressure leading to

$$\frac{u^{n+1} - u^*}{\Delta t} = \frac{-1}{\rho}\nabla p^{n+1} \tag{2.44}$$

It is easy to show that eq. 2.42 = eq.2.43+ eq.2.44 The pressure is found such that u^{n+1} satisfies the continuity equation by solving

$$-\nabla^2 p^{n+1} = \frac{-\rho}{\Delta t}\nabla.u^* \tag{2.45}$$

which can be derived by taking the divergence of equation 2.44 and enforcing $\nabla u^{n+1} = 0$. This equation is referred to as the pressure Poisson equation. The convective and viscous terms in equation2.43 are discretized using finite differences which approximate the derivatives using neighboring values. The predictor step for u velocity can be written as[12]

$$u^*_{i,j} = u^n_{i,j} + \Delta t \Bigg[-\frac{1}{h}(u^n_e u^n_e - u^n_w u_w + u^n_n u^n_n - u^n_s u^n_s) + \frac{\mu}{\rho h^2}(u^n_{i+1,j} + u^n_{i-1,j} + u^n_{i,j+1} + u^n_{i,j-1} - 4 \times u_{i,j}) \Bigg] \tag{2.46}$$

$$v^*_{i,j} = v^n_{i,j} + \Delta t \Bigg[-\frac{1}{h}(v^n_e v^n_e - v^n_w v^n_w + v^n_n v^n_n - v^n_s v^n_s) + \frac{\mu}{\rho h^2}(v^n_{i+1,j} + v^n_{i-1,j} + v^n_{i,j+1} + v^n_{i,j-1} - 4 \times v_{i,j}) \Bigg] \tag{2.47}$$

Where u^* and v^* are the components of u^* in x and y direction, respectively. Besides

$$u^n_e = \frac{u^n_{i,j} + u^n_{i+1,j}}{2} \quad u^n_w = \frac{u^n_{i,j} + u^n_{i-1,j}}{2} \tag{2.48}$$

$$u^n_n = \frac{u^n_{i,j} + u^n_{i,j+1}}{2} \quad u^n_s = \frac{u^n_{i,j} + u^n_{i,j-1}}{2} \tag{2.49}$$

$$p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1} - 4p_{i,j} = \frac{\rho h}{\Delta t}(u^*_{i,j} - u^*_{i-1,j} + v^*_{i,j-1} - v^*_{i,j+1}) \tag{2.50}$$

Therefore by the pressure obtained from above eq.2.50 the velocity can be solved as follows:

$$u^{n+1}_{i,j} = u^*_{i,j} - \frac{\Delta t}{\rho}\frac{(p_{i+1,j} - p_{i,j})}{h} \tag{2.51}$$

$$v^{n+1}_{i,j} = v^*_{i,j} - \frac{\Delta t}{\rho}\frac{(p_{i+1,j} - p_{i,j})}{h} \tag{2.52}$$

To calculate ghost node values we use linear interpolation.

Chapter 3

VOF Method

To invesitigate the complicated wave and their interaction in CFD simulation of multiphase flows have been significantly progressed and is supported by the most popular one consists of 1)interface tracking and 2)interface capturing [13], [14]

3.1 Interface Tracking Methods

Interface tracking methods represent that the interface between two fluids is explicitly tracked during the fluid motion to maintain a sharp discontinuity in the fluid viscosity and density. The motion of the interface is tracked either by marker points located on the surface or by attaching it to a mesh boundary surface . One of the interface tracking methods is Front Tracking Method (FT) [15] and

- Front Tracking Method [15, 16, 17] advects the marked interface from an initial configuration a separate front, on a fixed grid, is used to identify the interface for each phase with the help of additional computational elements introduced explicitly. These elements are known as marker particles on the surface and form a moving internal boundary cannot keep the interface completely sharp to provide stability and smoothness. An irregular grid is constructed in the vicinity of the interface and finite difference stencil is used to calculate fluid properties[18] Spacing between the marker particles is one of the complexities of this method. the main problem in the front tracking methods is the treatment of the topology of the front due to fact that the interaction of the interface grid with the stationary grid is very complicated.

- Another interface tracking method is moving mesh method. The pioneering work was done by Welch[19]and Hu et al.[20] Moving-mesh

methods can be used to track the location, account for changes in the interface topology, and resolve small-scale structures in the interfaces. The interface mesh points move with the interface in a Lagrangian fashion similar fluid elements are maintained in cells which are near to the interface and the fluid always coincide with the specified region helping the actual tracking of the fluid surface [21]. The interface is integrated with these points and is tracked by the nodes joined on both phases. The movement of the interface is determined exploiting the knowledge of velocities known at the current time step [22]. A simple interpolation between the fluid boundaries and the interface determines the motion of the mesh [19].

The computational mesh is adapted to fit the interface in interface tracking methods. The advantage of such methods is that they keep the sharpness of interface for which the exact position is known throughout the calculation. However, they require significant treatment when the interface is faced with large deformation. In terms of large deformation, a numerical inaccuracy in the solution of the flow field may occur due to the high mesh distortion, hence, this method is not proper for free surface problems such as dam breaking and wave propagation where complex interface breakup and merging occurs [21].

3.2 Interface capturing Methods

In Interface capturing Methods the sharp interface between two fluids is determined by an indicator function. Using the fluid velocities, the interface advection across the mesh is modelled by advecting the indicator function with the flowing fluid.The indicator function can be chosen as a level set function [23] or a volume fraction function [24]

- Level Set Method was first introduced by Osher[25] in 1988. The basic idea is that an arbitrary function takes different values. For example, considering positive values identify in one fluid and a negative one in the other as defined originally in [23]. Therefore, the interface is defined at those points where the function has a zero value. This method is known as the Level Set Method (LSM).. Althogth the LSM automatically takes care of the merging and breaking of the interface, handling sharp corners with high accuracy the drawback to this techniques is that, it does not conserve the mass naturally. This makes it a computationally expensive method [26]

- Volume of Fluid Methods [24] is one of best method to track the interface in two phases and free surface flow. This method is based on the volume

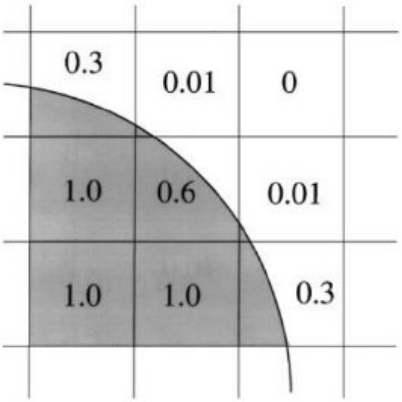


Figure 3.1: Volume fraction Function values

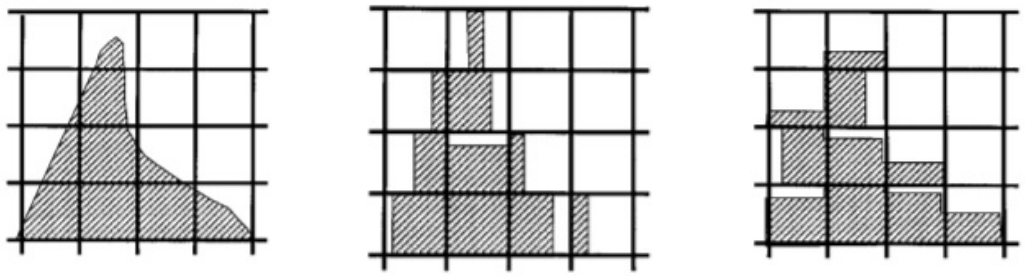
fraction function(is also called colour function) which is the step function and determine the fraction of volume occupied by one fluid. More precisely, taking the value 1 in one phase and 0 in the other phase [27], also, values between zero and one refers to the cell contains both fluids and is therefore an interface cell[2]. To find out more, the interface is reconstructed from the value of the volume fraction function[27] and this method solves a scalar transport equation in an Eulerian manner, therefore, the volume fraction of a mesh cell is conserved. The value of indicator function in 3.1 illustrate that the volume fraction function get the value one which the cell is fully covered by the fluid and in contrast presenting the value zero for the cell which is not occupied by that fluid and the value between zero and one defining the location of interface. Since the numerical implementation of the LSM is expensive and the interface reconstruction technique of the VOF method provides better approximation than LSM, therefore, we invistigate the VOF mothod and use it in our simulations.

3.2.1 VOF Method

Initilay VOF method presented by Hirt and Nichols [24] uses the VOF function F to show the movement of a fluid surface. The VOF advection equation which is an illustration of the interface is defined as

$$\frac{\partial F}{\partial t} + V.\nabla F = 0 \tag{3.1}$$

There are two steps in the VOF methods: the reconstruction of the interface and the advection of the interface. In this frame work different approaches have been used which are based on either an algebraic or a geometric VOF method.



(a) Actual Fluid **(b)** SLIC x-direction **(c)** SLIC y-direction

Figure 3.2: SLIC Interface Reconstruction

3.2.2 Geometric VOF Method

Geometric VOF methods get on with two steps. First, the interface is reconstructed in each computational cell from the knowledge of the volume fraction field. The interface is represented by a series of line segments connecting the sides of the cells. The interface may be approximated either by a straight line or a polynomial which divides the computational cell into two parts containing the calculated volume of each fluid. Second, the reconstructed interface is advected by computing the fluxed volume across each computational cell using geometric methods[28]. The essential approaches in this group include:

- The Simple Line Interface Calculation (SLIC) method [29] represents the straight line regenerated the interface which is selected as parallel to the coordinate axes. It is a direction-split algorithm and during each direction sweep, only cell neighbours in the sweep direction are used to determine the interface reconstruction [4]. This is depicted in Figures 3.2 (a) and (b), respectively, which the reconstructions used for an x-sweep and y-sweep of the interface configuration. Once the approximate interface reconstruction has been made, fluxes of volume for each material are calculated geometrically.

- Another method in the geometric approach is the Piecewise Linear Interface Calculation (PLIC) method which is lunched by Youngs. In this approach The interface is approximated by a straight line segment in each cell, which the slope of the line is given by the interface normal with an angle to coordinate axes which is shown in figure3.3. Therefore, it consists of an estimation of normal vector and volume fraction calculation based on the interface position. Thus, this can reconstruct the interface between fluids with second order accuracy and keep the interface sharper than SLIC method [2]. Because of unphysical flotsam was found in SLIC case,therefore, the PLIC method has been detcted more proper and accurate for the reconstruction step[30]. One approach that

(a) Actual Fluid **(b)** PLIC

Figure 3.3: PLIC Interface Reconstruction

considered as a family of PLIC is Youngs model which was first introduced by Youngs in 1982 [31] and developed by Rudman [32] with more details. In this method, first by an estimation of the slope of the interface location, the free surface is defined as a straight line with the slope of β(the angle the interface makes with the x-axis) in each cell of the computational domain. The position of this line segment in each cell is defined such that the area reconstructed from the line and the perimeter of the cell is equal to the amount of volume of fluid, F. Therefore, the transferred flux from the cell faces can be calculated by geometry of the polygon from this reconstruction.
Presuming that $F_{i,j}$ is predetermined in every cell, the first step is to calculate first-order upwind fluxes. Then, the fluxes of every cell can be calculated by considering the values in each cell. To do so, the angle β, between free surface and x-coordinate, must be calculated which can be done in different ways. Here to define β the gradient of F is used to calculate a surface normal from which β is determined in a straightforward fashion[2]. The approach of choosing the normal vector, however, can affect the accuracy of the final results. This formulation for uniform grid is as follows:

$$n^x_{i,j} = \frac{1}{\delta x}(F_{i+1,j+1} + 2F_{i+1,j} + F_{i+1,j-1} - F_{i-1,j+1} - 2F_{i-1,j} - F_{i-1,j-1}), \quad (3.2)$$

$$n^y_{i,j} = \frac{1}{\delta y}(F_{i+1,j+1} + 2F_{i,j+1} + F_{i-1,j+1} - F_{i+1,j-1} - 2F_{i,j-1} - F_{i-1,j-1}), \quad (3.3)$$

Therefore, the angle β can be calculated by components of the normal unit vector:

$$\beta = \tan^{-1}\left(\frac{-n^x}{n^y}\right) \quad (-\pi < \beta < \pi). \quad (3.4)$$

Determining the angle α to be

$$\alpha = \tan^{-1}\left(\frac{\delta x}{\delta y}\tan\beta\right) \quad (0 \le \alpha \le \pi/2). \quad (3.5)$$

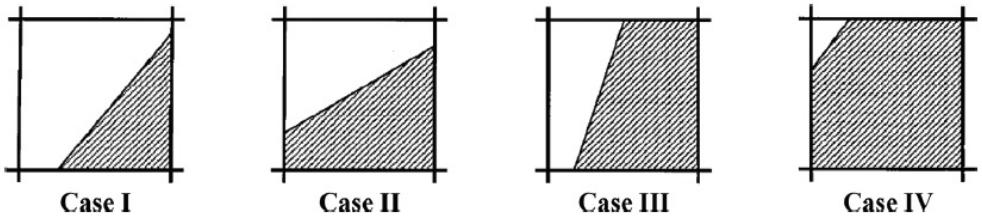


Figure 3.4: Four possible interface reconstructions for Youngs' Method

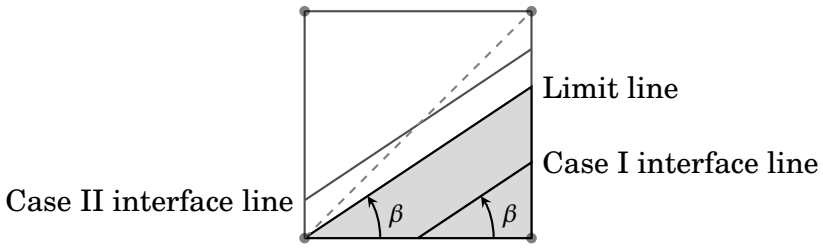


Figure 3.5: Reconstruction case

The interface cell can be rotated in such a way that α lies in the range $0° \leq \alpha \leq 90°$. Hence, there are four possible interface configurations which are depicted in Figure3.4.

The line cutting the cell can be presented by the equation:

$$n^x x + n^y y = d \tag{3.6}$$

where d is a constant. Therefore, the angle of the interface and normal vectors are defined in equations 3.4, 3.2 and 3.3, respectively. Conservation of volume requires the interface within the cell to be positioned such that the colored area under the interface becomes

$$A = F\delta x\delta y \tag{3.7}$$

This is the constraint that needs to be enforced in order to determine the reconstruction case for an interface with a given β. Consider the case depicted in Figure3.5, where for a given value of β the solid line cutting the cell from the bottom left corner is the limiting case separating case I and II. The area of the dark fluid under the limit line is

$$A_{lim} = \frac{1}{2}\delta x^2 \tan\beta \tag{3.8}$$

By substituting the defintion of A we have:

$$F_{lim} = \frac{1}{2}\frac{\delta x}{\delta y}\tan\beta \tag{3.9}$$

if $\alpha < \frac{\pi}{4}$ **then**
if $F \leq \frac{1}{2}\tan\alpha$ **then**
  case I
else if $F \leq 1 - \frac{1}{2}\tan\alpha$ **then**
  case II
else
  case IV
else
if $F \leq \frac{1}{2}\coth\alpha$ **then**
  case I
else if $F \leq 1 - \frac{1}{2}\cot\alpha$ **then**
  case III
else
  case IV
end if
end if

By the defined angle α we get

$$F_{lim} = \frac{1}{2}\tan\alpha \tag{3.10}$$

It can be shown that the dashed line cutting the cell diagonally in half corresponds to $\alpha_{half} = \frac{\pi}{4}$. Thus for the case under consideration with $\alpha < \alpha_{half}$ if $F < F_{lim}$ the interface must lie below the limit line which corresponds to the reconstruction case I. On the other hand, if $F < 1 - F_{lim}$ the interface lies above the limit line invoking the reconstruction case II.Other cases can be analysed analogously. Since by Figure 3.6 we have defined.

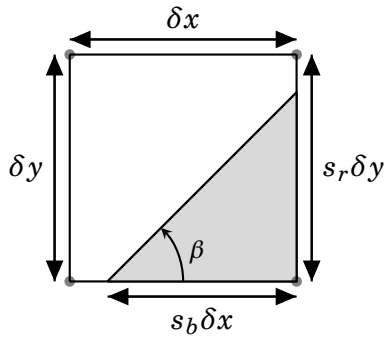


Figure 3.6: Reconstructed interface and its relevant parameters

The four side fractions can be calculated when the case has been detected. The side fractions are the fractions of the top, right, bottom and left sides of the cell that lie within the fluid (s_t, s_r, s_b and s_l). By equation3.5 we get:

$$\tan\alpha = \frac{s_r}{s_b} \tag{3.11}$$

And conservation of volume provide:

$$s_r s_b = 2F \tag{3.12}$$

Therefore, we obtain:

$$s_r = (2F\tan\alpha)^{1/2} \tag{3.13}$$

Thus, in similar fashion we get:

$$s_b = (\frac{2F}{\tan\alpha})^{1/2} \tag{3.14}$$

$$\tan\beta = \frac{s_r \delta y}{s_b \delta x} \tag{3.15}$$

By the known side fractions, the fluid fluxes (F_t, F_r, F_b, and F_l) can be calculated geometrically which is shown in these tables [33], and the flux calculations is given in those tables 3.1, 3.2.

	case I	case II
s_t	0	0
s_r	$\sqrt{2F\tan\alpha}$	$F+\frac{1}{2}\tan\alpha$
s_b	$\sqrt{2F\cot\alpha}$	1
s_l	0	$F-\frac{1}{2}\tan\alpha$
if $U_t>0$	**if** $U_t\delta t\le(1-s_r)\delta y$ $F_t=0$ **else** $F_t=\frac{1}{2}[U_t\delta t-(1-s_r)\delta y]^2\cot\alpha$	**if** $U_t\delta t\le(1-s_r)\delta y$ $F_t=0$ **else if** $U_t\delta t\le(1-s_l)\delta y$ **else** $F_t=U_t\delta t\delta x-(1-F)\delta x\delta y$
if $U_r>0$	**if** $U_r\delta t\ge s_b\delta x$ $F_r=F\delta x\delta y$ **else** $F_r=\frac{1}{2}U_r\delta t\big(2-\frac{U_r\delta t}{s_b\delta x}\big)s_r\delta y$	$F_r=U_r\delta t(s_r\delta y-\frac{1}{2}U_r\delta t\tan\alpha)$
if $U_b>0$	**if** $U_b\ge s_r\delta y$ $F_b=F\delta x\delta y$ **else** $F_b=\frac{1}{2}U_b\delta t(2-\frac{U_b\delta t}{s_r\delta y})s_b\delta x$	**if** $U_b\delta t\ge s_l\delta y$ $F_b=U_b\delta t\delta x$ **else if** $U_b\delta t\le s_r\delta y$ $F_b=\frac{1}{2}(U_b\delta t-s_l\delta y)^2\cot\alpha$ **else** $F_b=F\delta x\delta y$
if $U_l>0$	**if**$U_l\delta t\le(1-s_b)\delta x$ $F_l=0$ **else** $F_l=\frac{1}{2}(U_l\delta t-(1-s_b)\delta x)^2\tan\alpha$	$F_l=U_l\delta t(s_l\delta y+\frac{1}{2}U_l\delta t\tan\alpha)$

Table 3.1: The fluid Fluxes for case I and II.

	case III	case IV
s_t	$F - \frac{1}{2}\cot\alpha$	$1 - \sqrt{2F\cot\alpha}$
s_r	1	1
s_b	F + $\frac{1}{2}\cot\alpha$	1
s_l	0	$1 - \sqrt{2F\tan\alpha}$
if $U_t > 0$	$F_t = U_t\delta t(s_t\delta x + \frac{1}{2}U_t\delta t\cot\alpha)$	**if** $U_t\delta t \geq (1-s_l)\delta y$ $F_t = U_t\delta t\delta x - (1-\alpha)\delta x\delta y$ **else** $F_t = U_t\delta t(s_t\delta x + \frac{1}{2}U_t\delta t\cot\alpha)$
if $U_r > 0$	**if** $U_r\delta t \leq s_t\delta x$ $F_r = U_r\delta t\delta y$ **else if** $U_r\delta t \leq s_b\delta x$ $F_r = U_r\delta t\delta y -$ $\frac{1}{2}(U_r\delta t - s_t\delta x)^2\tan\alpha$ **else** $F_r = F\delta x\delta y$	**if** $U_r\delta t \leq s_t\delta x$ $F_r = U_r\delta t\delta y$ **else** $F_r = U_r\delta t\delta y -$ $\frac{1}{2}(U_r\delta t - s_t\delta x)^2\tan\alpha$
if $U_b > 0$	$F_b = U_b\delta t(s_b\delta x - \frac{1}{2}U_b\delta t\cot\alpha)$	**if** $U_b\delta t \leq s_l\delta y$ $F_b = U_b\delta t\delta x$ **else** $F_b = U_b\delta t\delta x -$ $\frac{1}{2}(U_b\delta t - s_l\delta y)^2\cot\alpha$
if $U_l > 0$	**if**$U_l\delta t \leq s_b\delta x$ $F_l = U_l\delta t\delta y - (1-F)\delta x\delta y$ **else** $F_l = U_l\delta t(s_l\delta y + \frac{1}{2}U_l\delta t\tan\alpha)$	**if** $U_l\delta t \geq (1-s_t)\delta x$

Table 3.2: The fluid Fluxes for case III and IV.

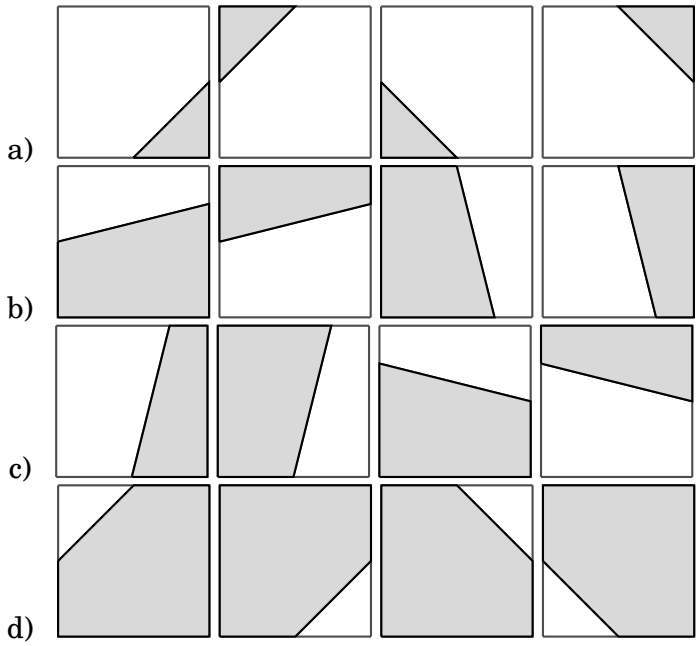


Figure 3.7: Different interface positions. (a) case I, (b) case II, (c) case III, (d) case IV

Therefor,Figure 3.7 shows the various possibility of the interface position in each case which gives us better insight of the rotation of interface in PLIC family [34]. Besides, to make it clear, the side fraction calculated by flux fraction of figure 3.8 represented as below

$$\begin{aligned} F_{i,j} &= \frac{\square ABCD}{\delta x \delta y} \\ &\Rightarrow F_{i,j} = \frac{0.5 s_b \delta x (s_l + s_r)\delta y}{\delta x \delta y} \\ &\Rightarrow F_{i,j} = 0.5(s_l + s_r)(s_b = 1) \\ &\Rightarrow \tan\alpha = \frac{(s_r - s_l)\delta y}{s_b \delta x} = s_r - s_l \\ s_l &= F_{i,j} - 0.5\tan\alpha \\ s_r &= F_{i,j} + 0.5\tan\alpha \end{aligned} \tag{3.16}$$

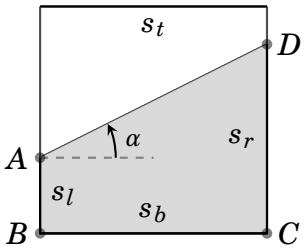


Figure 3.8: Geometrical estimation of side fraction

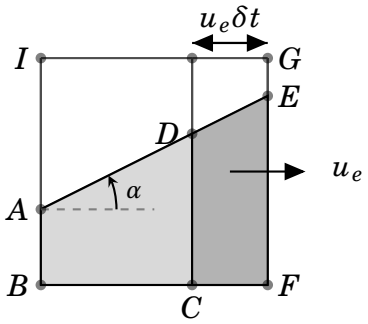


Figure 3.9: Calculating area of cell filled by fluid

Thus, by the known side fraction in each case, the area of the cell which filled by the fluid can be calculated, which is as follows for figure 3.9

$$\begin{aligned} \square CDEF &= 0.5 \times CF \times (CD + FE) \\ CF &= u_e \delta t, \quad FE = s_r dy \\ \tan\alpha &= \frac{FE - CD}{CF} \\ \Rightarrow CD &= FE - CF\tan\alpha \\ \square CDEF &= 0.5 \times CF \times (2FE - CF\tan\alpha) \\ \square CDEF &= u_e \delta t \times (s_r \delta y - 0.5 u_e \delta t \tan\alpha) \end{aligned} \tag{3.17}$$

The other fluxes can be calculated in the same manner.

3.2.3 VOF Advection

The VOF advection is governed by equation 3.1, which for a divergence free velocity field can be written as

$$\frac{\partial F}{\partial t} + \nabla.(\vec{V}F) \tag{3.18}$$

Discretization with respect to time and space and using the theory of finite volume provide

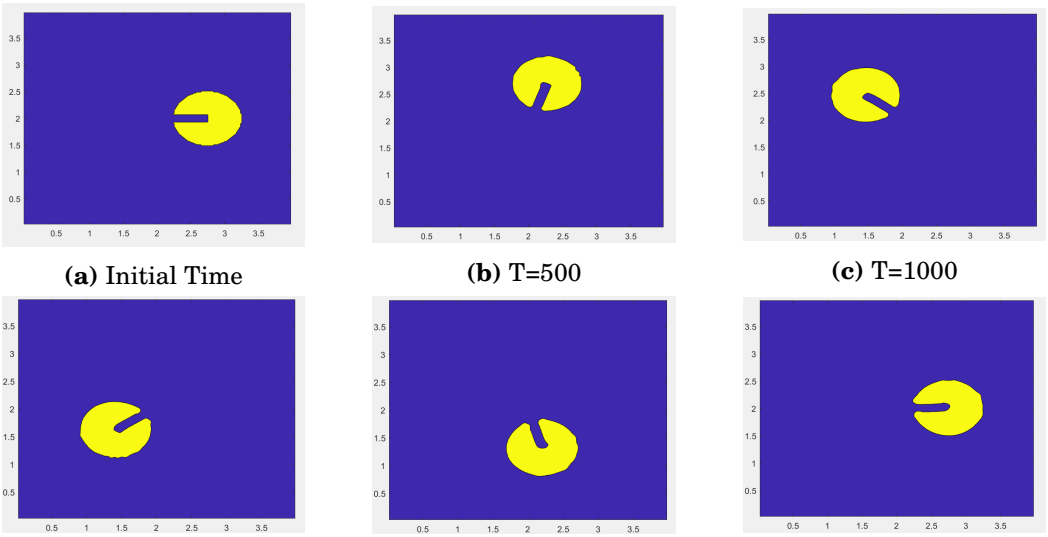
$$\frac{F_{i,j}^{n+1}-F_{i,j}^{n}}{\delta t}+\frac{(Fu)_{i-1/2,j}-(Fu)_{i+1/2,j}}{\delta x}+\frac{(Fu)_{i,j-1/2}-(Fu)_{i,j+1/2}}{\delta y}=0 \qquad (3.19)$$

$$F_{i,j}^{n+1}=F_{i,j}^{n}+\frac{\delta t}{\delta x}\Big[(Fu)_{i-1/2,j}-(Fu)_{i+1/2,j}\Big]+\frac{\delta t}{\delta y}\Big[(Fu)_{i,j-1/2}-(Fu)_{i,j+1/2}\Big] \qquad (3.20)$$

where $F_{i,j}^{n}$ is the volum function value for the $i,j-th$ cell at the $n-th$ time step, $u_{i-1/2},u_{i+1/2}$ are the horizontal component of the velocity and $u_{j-1/2},u_{j+1/2}$ the vertical component. $Fu_{i-1/2},Fu_{i+1/2},Fu_{j-1/2},Fu_{j+1/2}$ and are flux rate which present the amount of fluid passing through each of the cell's four faces per time.

3.2.4 Rotation of Slotted Disk

Rotational Slotted Disk is an example which follows the above scenario and it is known by Zalesak's problem, in which a slotted circle is rotated through one or more revolutions. Here the mesh size is 200×200 with the physical domain 4×4, where the diameter of the slotted circle is 50 mesh cells and slot width is six mesh cells. Also, the axis of rotation is the centre of the computational domain which is $(2.0,2.0)$ and the circle centre is initially at $(2.0,2.75)$ and rotates around the center with velocity field $u_x=-\Omega(y-y_0)$ and $v_y=\Omega(x-x_0)$. Since this velocity is divergence free throughout the domain, the interface of this disk is expected to retain the original shape at the end of rotation. The test ends when the first (anticlockwise) revolution at time 2524 of the slotted disk is completed which is shown in figure3.10.Besides, the volume fraction is considered in such a way that inside disk has value one and outside of disk has value zero initially. It is sensible that the Young's method gives sharp interfaces with acceptable interface shapes, although the sharp corners at both ends of the slot become rounded [35].



(a) Initial Time **(b)** T=500 **(c)** T=1000

(d) T=1500 **(e)** T=2000 **(f)** T=2524

Figure 3.10: Young's Interface Reconstruction

3.2.5 Summery

General notation VOF approah for the incompressible, is othermal and immiscible fluid, is the single set of continuity and momentum equation for the whole domain [36]. The continuity equation is written as:

$$\nabla.\vec{U} = 0 \tag{3.21}$$

where $\vec{U}$ is the fluid velocity. The momentum equation is

$$\frac{\partial \rho \vec{U}}{\partial t} + \nabla.(\vec{U}\vec{U}) = -\nabla p + \nabla.\mu(\nabla\vec{U} + \nabla\vec{U}^T) + \rho\vec{g} + \vec{F}_{st} \tag{3.22}$$

where last term represents the surface tension forces, the second last term is the viscous term, $\vec{g}$ is the acceleration due to gravity and p is the pressure. As we have already discussed about VOF method, we know that scalar function called volume fraction for each phase detects as

$$\alpha = \begin{cases} 0 & \text{within phase 2 or gas} \\ 0 < \alpha < 1 & \text{at the interface} \\ 1 & \text{within phase 1 or liquid} \end{cases} \tag{3.23}$$

The fluid properties such as density ρ and viscosity μ in a control volume are calculated as

$$\rho = \alpha_1\rho_1 + (1-\alpha_1)\rho_2 \tag{3.24}$$

$$\mu = \alpha_1 \mu_1 + (1 - \alpha_1)\mu_2 \tag{3.25}$$

where $1,2$ which refers to the liquid and gas phases, notice that $\alpha_1 + \alpha_2 = 1$ This indicates that we use the average of the ρ and μ in the equation3.22. Advection of the volume fraction of α is implemented

$$\frac{\partial \alpha}{\partial t} + \nabla.(\alpha \vec{U}) + \nabla.(\alpha(1-\alpha)\vec{U_r}) = 0 \tag{3.26}$$

where the third term is an artificial compression term used to sharpen the interface [58,61]. The artificial compression term uses a relative velocity$\vec{U_r}$ defined as

$$\vec{U_r} = C_\alpha \vec{U} \frac{\nabla \alpha}{|\nabla \alpha|} \tag{3.27}$$

where c_α is the sharpening factor which get the value between 0 and 1 [36].

Chapter 4

OpenFOAM

4.1 Turbulence Model: $k-\omega$ SST

The $k-\omega$ SST (Shear Stress Transport) model is known turbulence model which is the combination of $k-\epsilon$ and $k-\omega$ models by rewriting both k and ϵ equations in terms of ω. [37]. In this model two new variables are solved by introducing two additional transport equations which is as follows:

$$\frac{\partial(\rho k)}{\partial t}+\frac{\partial(\rho U_i k)}{\partial x_i}=\tilde{P}_k-\beta^*\rho k\omega+\frac{\partial}{\partial x_i}[(\mu+\sigma_k\mu_t)\frac{\partial k}{\partial x_i}] \tag{4.1}$$

$$\begin{aligned}\frac{\partial(\rho\omega)}{\partial t}+\frac{\partial(\rho U_i\omega)}{\partial x_i}=\frac{\alpha\tilde{P}_k}{\nu_t}-\beta\rho\omega^2+\frac{\partial}{\partial x_i}[(\mu+\sigma_\omega\mu_t)\frac{\partial\omega}{\partial x_i}]\\ +2(1-F_1)\rho_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial\omega}{\partial x_i}\end{aligned} \tag{4.2}$$

where, k is the total turbulent kinetic energy, ω is the specific eddy dissipation rate, $\tilde{P}_k$ is the turbulent kinetic energy production rate, β, β^*, σ_k,σ_ω and $\sigma_{\omega 2}$ are constants.Using a turbulence model needs to calculate the initial values for the turbulent parameters must be calculated. For the $k-\omega$ SST model, initial values for k and ω can be define as follows:

$$k=\frac{3}{2}(IU_\infty) \tag{4.3}$$

$$\omega=\frac{\sqrt{k}}{\beta^{*0.25}l} \tag{4.4}$$

Where, I is the turbulent intensity, β^* = 0.09 and l is a turbulent length scale.

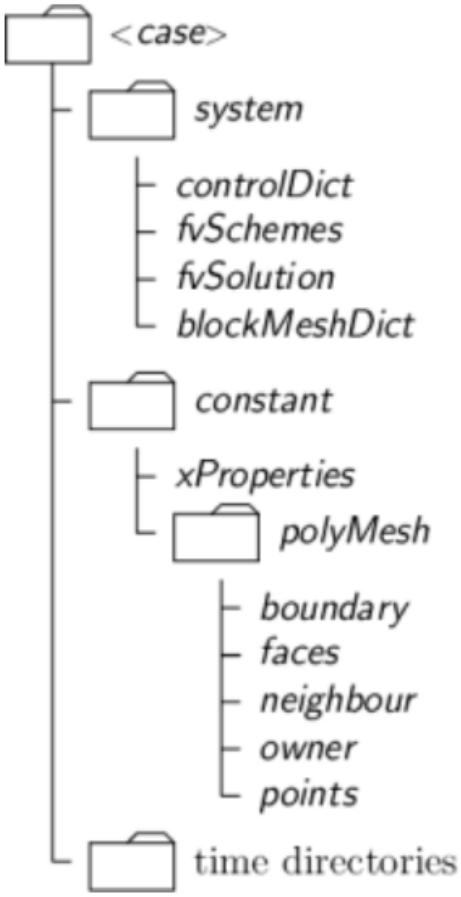


Figure 4.1: OpenFoam case folder

4.2 OpenFOAM

OpenFOAM is the abbreviation of Open source Field Operation And Manipulation which is a C++ libraries that can be employed to generated a wide range of computational solvers for problems in continuum mechanics with a focus on finite volume discretization [38]. OpenFOAM also consists of several ready solvers, utilities, and applications that can be directly used. In principle, the user can manipulate meshes, geometries, and discretization techniques at a high level of coding which consists of consists of Pre-processing, Solving and Post-processing

4.2.1 Creating Mesh

In OpenFOAM, mesh can create with blockMesh by defining the mesh in blockMeshDict as an input file which is stored in system or in consatant/polyMesh dictionary for a given case [39] as shown in figure 4.1. In this file one

should specify the vertices of the physical domain and define its blocks and assign a name to each patch which consists of intended vertices for each segment. If there exists a geometry inside the computational domain, one can create that by one the the cad software as an STL file and read it by snappyHexMeshDict which can refine the mesh in proper area such as around the hull and free surface. in snapyHexMesh, the mesh is refined around the geometry for a number of levels specified by the user. For each level of refinement, a three-dimensional cell is split into eight new cells. Therefore, a high number of refinement levels can drastically increase the number of cells in the mesh. In addition, other areas of the domain can be refined by specifying regions of interest and refinement levels to be used in the wanted region. SnappyHexMesh then proceeds to snapping the mesh to the surface, i.e. fitting the mesh smoothly around the geometry. The last step in snappyHexMesh is adding layers to the geometry. However, when a free surface is involved, some extra steps are needed in order to refine the free surface region before doing snappyHexMesh like Selecting the free surface region by using topoSet utility and Refining mesh, based on the cells selected in topoSet, using the refineMesh utility. Toposet and mesh refinining are executed multiple times for achieving an appropriate free surface refinement. Besides, dynamicMesh dictionary controls deformation and morphing of the mesh during a simulation.Here the dynamicMotionSolverFvMesh which is define the type of the Mesh motion, based on solved mesh motion for rigid body motion [40].

4.2.2 Initialization of the Boundary

After mesh generation is finished, one can set the initial values and boundary conditions for the fields like pressure, velocity and etc for each segment of geometry. This is stored in 0 sub-directory of the considered case. There are different type of boundary conditions which are as follows[41]. **calculated** No special constraint is set on the flow parameter when using this boundary condition. Instead, it is calculated based on other flow parameters at the same boundary.
fixedFluxPressure This is a gradient condition for the pressure, where the gradient is user specified. For a value of zero, a zero gradient condition is applied and the flux over the boundary is determined entirely by the velocity.
Value This boundary condition sets a fixed value on the patch when used. It can for instance be used for a uniform inflow, and to impose a no-slip condition on surfaces by specyfying zero velocity.
inletOutlet In principle, this boundary condition is the same as setting a zero gradient. However, it can be said to be a stronger condition based on the fact that it can handle return flow. Therefore, the user can specify if return flow is allowed or not, and what values to use if return flow occurs.
kqRWallFunction Specifies the wall treatment of the turbulent kinetic en-

	ρ	ν
water	998.8	1.090×10^{6}
air	1	1.48×10^{-5}

Table 4.1: Fluid properties

ergy to be in the logarithmic region.
nutKWallFunction Specifies the wall treatment of the turbulent kinematic viscosity to be in the logarithmic region.
omegaWallFunction Specifies wall treatment of specific eddy dissipation rate, ω.
outletPhaseMeanVelocity Used at the outlet. The mean velocity of the flow is specified in order to obtain equal flux of mass flow rate out of the domain as into the domain.
pressureInletOutletVelocity Applied to boundaries where the pressure is specified, i.e. the atmosphere boundary. It imposes a zero gradient condition for outflow.
slip The slip condition is essentially a symmetry condition. It implies that there is no shear force and no flow through the boundary, and that the velocity at the boundary is parallel to the boundary.
symmetryPlane A condition than ensures symmetry about the boundary. It is used to ensure that no outflow occurs along the boundary.
totalPressure Sets the static pressure at the boundary based upon a specification of the total pressure. Used for a boundary that borders to the atmosphere.
variableHeightFlowRate Used for the phase fraction, alpha.water. It allows for a varying water height at the outlet, which is needed when surface waves are created.
zeroGradient When used, a zero gradient is applied for the parameter over the boundary. This means that no change over the boundary is desired, i.e. the flow parameter is constant at the boundary.

4.2.3 Phsical Properties

The physical properties like density, kinematic viscosity and etc. are going to set in transportProperties dictionary constant folder, but in multiphase case we have different number of phases. Therefore, we need to specify the properties of all phases with respect to each material.

4.2.4 Numerical Solvers and Algorithms

InterDyMFoam is used for flows that involve two incompressible, isothermal, immiscible fluids using a volume of fluid (VOF) phase-fraction based interface capturing approach (The OpenFOAM Foundation, 2017), which provides optional mesh motion and for the mesh deformation. The algorithm which is used in InterDyMFoam to solve Navier-Stokes Equation is Pressure-Implicit Method for Pressure linked equation (PIMPLE). PIMPLE is a combination of Pressure-Implicit with Splitting of Operators (PISO) and Semi-Implicit Method for Pressure Linked Equation (SIMPLE) [42, 43].
SIMPLE Algorithm:

1. Guess the pressure at each grid point.
2. Solve the momentum equations to find the velocity components.
3. Solve a pressure-correction equation to find a corrected pressure at each grid point.
4. Correct the pressure and velocity according to given equations.
5. Replace the previous intermediate values of the pressure and velocity with the new corrected values. Step 2 through 5 are repeated until the solution converges.

PISO Algorithm:

1. Predictor step, where the pressure field prevailing at time level i is used in the implicit solution of the momentum equations.
2. First corrector step, where a new pressure field is sought along with a revised velocity field that will satisfy conversation of mass.
3. Second corrector step, where the pressure field and velocities are revised again.
4. More corrector steps can be made in the same way as step two and three. However, it is argued that only two corrector steps are sufficient for for most purposes.

The main principal of the PIMPLE algorithm is that Within one time step, a steady-state solution with the user specified tolerance is searched, after the solution found, it moves on in time. Therefore, to ensure that explicit parts of the equations are converged, it requires the outer correction loops. After reaching a defined tolerance within the steady-state calculation, it is time to leave the outer correction loop and move on in time. This is done till

we reach the end time of the simulation. The setting of PIMPLE should be done in fvSolution dictionary in system folder with the setting parameters nOuterCorrectors and the nCorrectors (inner loops) which are related to the PIMPLE and to the PISO algorithm, respectively [44].The parameter, nOuterCorrectors is the number of outer correctors used by the PIMPLE algorithm and specifies how many times the entire system of equations should be solved within one time step. To make it easier, for example if the PIMPLE in fvSolution is set to,

```
PIMPLE
{
// Outer Loops ( Pressure - Momentum Correction )
nOuterCorrectors                                    3;
//Inner Loops ( Pressure Correction )
nCorrectors                                         1;
}
```

It means that, for one time step, three outer loops are calculated and within one outer loop, the pressure is calculated once.The benefit of merging the SIMPLE and PISO algorithms into the PIMPLE algorithm, is that one can use larger Courant numbers ($Co >> 1$) and therefore, the time step can be increased and still achieve stability in the numerical simulations. To solve the pressure, the geometric-algebraic multi-grid (GAMG) solver is used. The principle behind is to generate a quick solution on a mesh with a small number of cells, and mapping the solution onto a finer grid, using the first solution as an initial guess. Using the GAMG solver has been proven to be faster than other solvers, as long as the computational cost outweighs the time it takes for refining the mesh [39]. The velocity and turbulent quantities are solved by using Gauss-Seidel method. It is one of the most efficient and useful point-iterative procedures for a large system of equations[43]. For modelling the interface between water and air, interDyFoam uses the VOF method. It gives a phase fraction, "alpha.water", based on the two phases water and air in each cell. In fact, in OpenFOAM the volume fraction equation:

$$\frac{\partial \alpha}{\partial t} + \nabla.(\alpha u) = 0 \tag{4.5}$$

Is supported by the FCT (Flux Corrector Transport) theory is MULES (Multi-dimensional Universal Limiter for Explicit Solution) relay on Zalesak's weight to combine the low order scheme and high order scheme[45], which is as follows. firstly rewriting the advection equation to integral form

$$\int_{\Omega_i} \frac{\partial \alpha}{\partial t} + \int_{\partial\Omega_i} (\alpha u).n ds = 0 \tag{4.6}$$

where Ω_i represents each cell, $\partial\Omega_i$ is the cell boundary and n is the cell bound-

ary normal. discretize the first term

$$\frac{\alpha_i^{n+1} - \alpha_i^n}{\Delta t} = \frac{1}{|\Omega_i|} \sum_{f \in \partial\Omega_i} (F_u + \lambda_M F_c)^n \tag{4.7}$$

where F_u and F_c are the advective fuxes and λ_M is the delimiter taking the value 1 at the surface and 0 elsewhere. These advective fuxes are defined as

$$F_u = \Phi_f \alpha_{f,upwind} \tag{4.8}$$

and

$$F_c = \Phi_f \alpha_f + \Phi_{rf} \alpha_{rf}(1 - \alpha_{rf}) - F_u \tag{4.9}$$

where Φ_f is the volumetric face flux. The subscript f denotes that the quantity is evaluated at the face, $upwind$ that an upwind scheme is used and the quantities with the subsrcipt rf are given below. Φ_{rf} is given by

$$\Phi_{rf} = min\Big(C_\alpha \frac{|\Phi_f|}{|S_f|}, max\Big[\frac{|\Phi_f|}{|S_f|}\Big]\Big)(n.S_f) \tag{4.10}$$

where C_α is a user specified parameter reducing interface smearing, S_f is the cell face area vector, and n_f is the face centered interface normal vector. α_{rf} is given by

$$\alpha_{rf} = \alpha_P + \frac{\alpha_N - \alpha_P}{2}[1 - \chi(\Phi_f)(1 - \lambda_{\alpha r})] \tag{4.11}$$

where N and P denotes the upwind and downwind neighbour respectively, $\lambda_{\alpha r}$ is a limiter and $\chi$is a step function taking the value 1 where volumetric face flux is positive and -1 where it is negative. For $\lambda_M = 0$ which is away from the interface, the expression inside the sum in equation 4.7 reduces to F_u, which uses a simple upwind scheme for the advection term. For $\lambda_M = 1$, which is at the interface, the expression inside the sum becomes a combination of a higher order scheme for the advection represented by the term $\Phi_f \alpha_f$ and a compressive flux term represented by the term $\Phi_{rf} \alpha_{rf}(1 - \alpha)_{rf}$. The higher order scheme gives a more accurate advection at the surface and the compression flux term reduces the surface smearing. At the same time computational effort is reduced away from the surface. Numerical diffusion at the interface is also reduced. Therefore, the MULES algorithm to calculate the Zalesak's weight consists of four steps; First step is determine the corrected local extrema for each cell, followed by the second step to sum up the inflow and outflows for each cell. Step three is the main part, which will be executed nLimiterIter times. In the last step, MULES calculates the weight of each cell and face. Due to the structure of OpenFOAM, the same face may have a different weight for the neighbor cells. Thus, MULES do these process to get the minimum of the weight of one face. Therefore, it is very efficient method that guarantee boundedness of scalar fields, specially in phase/mass-fractions. MULES method is

basically explicit which define a strict Courant number limit, and hence time step limit, when running the solvers, which is only partially relived by the introduction of time step subcycling. To summarize, a simple upwind solution of the uncompressed α equation can be obtained by setting MULESCorr to true. The parameter cAlpha controls the amount of interface compressionwhere 0 corresponds to no compression; 1 corresponds to conservative compression; and, anything larger than 1, relates to enhanced compression of the interface. The factor nAlphaSubCycles represents the number of subcycles within the α equation for each timestep(splitting of the time step into nAlphaSubCycles in the solution of the α equation) [46]. Finally, the parameter nAlphaCorr, specifies how many times the alpha field should be solved within a time step, meaning that first the alpha field is solved for, and this new solution is then used in solving for the alpha field again. Actually, number of correction forα, to improve quality of solution via fixed point iteration. Beside MULLES method, there is another way to solve the volume fraction equation at the interface via VOF method which is called isoAdvector scheme [47]. Therefore, consider continuous phase fraction field $H(x,t)$:

$$\alpha_i = \frac{1}{V_i}\int_{\Omega_i} H(x,t)dV \tag{4.12}$$

Total volume of tracked phase transported accross face j during one time step:

$$\Delta V_j(t,\Delta t) = \int_t^{\Delta t}\int_{F_j} H(x,\tau)u(x,\tau)dSd\tau \tag{4.13}$$

This is the quantity that is estimated in the isoAdvector method. It is estimated using the quantities α_i, u_i and Φ_j which are known at time t, where Φ_j is the face flux across face j.

$$\Phi_j(t) = \int_{F_j} u(x,t)dS \tag{4.14}$$

Update phase fraction for next time step:

$$\alpha_i(t+\Delta t) = \alpha_i(t)\frac{1}{V_i}\sum_{j\in B_i} s_{ij}\Delta V_j(t,\Delta t) \tag{4.15}$$

isoAdvector estimates $\Delta V_j(t;\Delta t)$

isoAdvector Algorithm

1. Initialize with upwind cell face value $\Delta V_j = \alpha_{upwind;j}\Phi_j\Delta t$ at time t
2. Find the surface cells with $0 < \alpha(t) < 1$
3. For each surface cell:

(a) Calculate the initial isosurface, which is shown in figure4.2

- Seeking isovalue f that cuts cell into correct phase fractions
- Interpolate cell phase fractions to cell vertices. Cell vertex fractions $f_1 ... f_N$
- Decide where cell edges are cut for different isovalues $f_1 ... f_N$. Cut if isovalue is between the edge vertex fractions f_k and f_l:

$$x_{cut} = x_k + \frac{f - f_k}{f_l - f_k}(x_l - x_k) \tag{4.16}$$

- Calculate $\tilde{\alpha}$ for each $f_1 ... f_N$ and use result to construct polynomial for $\tilde{\alpha}$
- Find correct isovalue with $|\tilde{\alpha}(f) - \alpha_i| < tol$

(b) Estimate the movement of the isosurface during a time step, shown in figure4.3

- Calculate isosurface center x_s and normal vector n_s
- Interpolate the cell velocity u_i to x_s to get velocity vector $\boldsymbol{U}_s$
- Isosurface motion:

$$U_s = \boldsymbol{U}_s . n \tag{4.17}$$

(c) Calculate submerged face area

- Estimate when the isosurface will reach vertex k of face j

$$t_k \approx t + (\boldsymbol{X}_k - x_s)\frac{n_s}{U_s} \tag{4.18}$$

- Use this to calculate the total face area of one cell that is inside the tracked phase(Fluid A) during one time step, A_j

(d) Estimate $\Delta V_j(t, \Delta t)$ and update α_i

- Time integral of A_j over the time step multiplied by the face flux gives estimate for $\Delta V_j(t, \Delta t)$
-

$$\alpha_i(t + \Delta t) = \alpha_i(t) - \frac{1}{V_i} \sum_{j \in B_i} s_{ij} \Delta V_j(t, \Delta t) \tag{4.19}$$

(e) To avoid values of $\alpha < 0$ or $\alpha > 1$ a bounding procedure is required. This is done by letting excess of phase A be redistributed to downwind cells in the case of $\alpha > 1$. If $\alpha < 0$, the equations are rewritten in terms of phase B and excess of phase B is redistributed to downwind cells. Redistribution is done to ensure volume conservation. In rare cases strict clipping of the value might be required, however this does not give volume conservation.

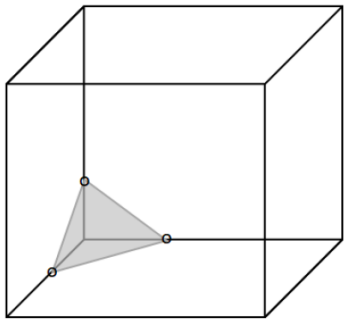


Figure 4.2: The initial isosurface in a hexahedral cell where one corner is submerged in the tracked phase. The locations where the edges are cut by the isosurface are marked with circles.

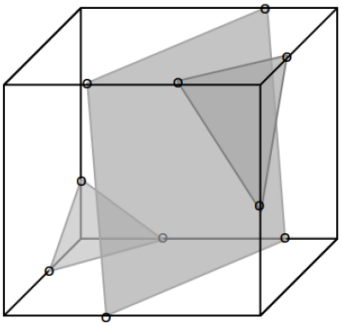


Figure 4.3: The propagating isosurface.

Therefore, to use isoAdvector some other parameters must be set in fvSolution which consists of isofaceTol refers to error tolerance on alpha when cutting surface cells into sub-cells, surfCellTol shows only cells with surfCellTol < alpha < 1- surfCellTol are treated as surface cells and nAlphaBounds 3 presents number of times the bounding step should try to correct unboundedness.

In steady state analyses the time derivatives are set to zero. However, in the transient analyses, using interDyFoam, the time derivatives are discretised using Euler discretisation. This is a first order and implicit method. (The OpenFOAM Foundation, 2017). One can adjust the preferred finite volume discretisation schemes in the fvSchemes in the system folder. In discretisation schemes, robustness and accuracy it should be considered. Therefore, A linear upwind method is used for discretisation of divergence terms for the velocity and the turbulent parameters. The linear upwind method is a second-order, upwind-biased, unbounded method that requires discretisation of the velocity gradient to be specified [48].

4.2.5 Stability

In numerical point of view it is essential to ensure numerical stability. If the calculations are numerically unstable, they will not converge towards the correct results and a simulation may crash. For simulations described in this work, the time dependent NS-equation is solved. Therefore, time integration is performed numerically, and stability must be preserved in order for the results to converge. The stability criterion that have to be satisfied is the Couran- Friedrichs-Lewy (CFL) condition, which is formulated in equation 4.20 Here, Δt is the time step and Δx is the local element size in one dimension

$$C = \frac{U\Delta t}{\Delta x} \tag{4.20}$$

By interpreting the CFL equation, it can be said that it gives the movement of a fluid particle over a number of cells per time step. For explicit differentiation schemes, this number should be kept below unity. However, when implicit differentiation schemes are used for the divergence terms of the velocity, the CFL condition can be higher [44]. The reason is that implicit schemes can tolerate that a fluid particle moves over more than one cell during a time step. However, care must be taken. A too large CFL number may still yield an unstable solution. Since the interDyFoam solver (which is based on the PIMPLE algorithm) is implicit, and can therefore tolerate CFL numbers that are larger than one. A time step sensitivity analysis in terms of the CFL number is performed in order to maintain stability in the numerical analyses.

4.2.6 Forces, Post-processing

OpenFOAM is able to calculate forces for each solid surface like the ship hull. OpenFOAM divides the total force into two components, pressure resistance, R_P , and viscous resistance, R_V . The sum of these two force components constitute the total resistance, R_T.

After pre-processing and solving the problem, it is time to visualize the solution. Thus, a version of ParaView which is supported by OpenFOAM, can read the data in OpenFOAM, with the options such as geometry surfaces, cutting planes, vector plots and streamlines. Therefore, users can create animations with ParaView properly.

4.2.7 Information of Resources

In order to have accurate results CFD simulation requires enough memory for meshing and running analysis.Also, processor power is needed for speed in the calculations. Therefore, a normal laptop cannot support the the numerical calculation for large test case. In this work, I ran the simulation on a PC

Memory	7,7 GiB
Processor	Intel®Core™i7 CPU 870 @ 2.93GHz ×8
Graphics	AMD®
OS type	64-bit
Disk	1,2 TB

Table 4.2: PC Information

Memory	13199419 KiB
Processor	16
Model name	Intel®Xeon ®CPU E5-2667 v2 @ 3.30Ghz

Table 4.3: Cluster schorsch Information

and cluster schorsch with linux system which the information is shown in tables4.2 and 4.3.

Chapter 5

Results

The results are discussed in this chapter, which is investigating the DTCHUll ship with wave and its resistance[49].

5.1 Physical Domain and Boundary Conditions

The physical domain are determined in figure 5.1 which is involved with inlet, outlet, bottom, atmosphere, hull, front and back side. The hull is positioned at interface between water and air.

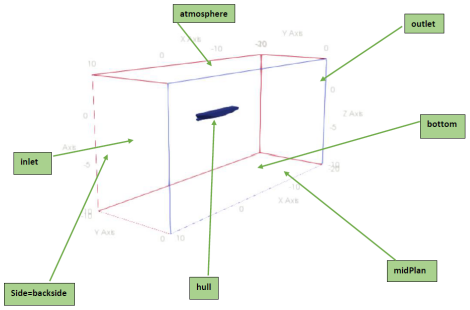


Figure 5.1: Physical Domain

Parameter/Patch	U	alpha	k	ν_t	ω
inlet	waveVelocity	waveAlpha	fixedValue	fixedValue	fixedValue
outlet	outletPhaseMeanVelocity	variableHeightFlowRate	inletOutlet	zeroGradient	inletOutlet
atmosphere	pressureInletOutletVelocity	inletOutlet	inletOutlet	zeroGradient	inletOutlet
hull	movingWallVelocity	zeroGradient	kqRWallFunction	nutkRoughWallFunction	omegaWallFunction

Table 5.1: Boundary conditions

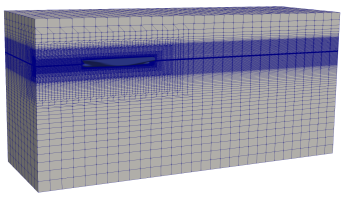


Figure 5.2: Refined Physical Domain

The Boundary conditions for this physical problem at each side of the domain are shown in table 5.1. After doing blockMesh to generate the physical domain's mesh, one has to import the DTCHull stl file into domain by snappyHexMesh. To refine the mesh around the hull topoSet chooses the refinement box around the hull and refineMesh refines the mesh. This is shown in figure 5.2.

5.2 Results and Discussion

To run the simulation in parallel, one should define the number of processors, and a method for decomposition in decomposepardic. For instance, I used 8 and 16 processors and scotch method for different cases, which requires no geometric input from the user and attempts to minimise the number of processor boundaries. After setting some parameters in controlDict, fvSoluntion and fvScheme, the simulation is ready to run for multi-phase case. In this simulation I used interDyMFoam solver with aim of solving Navier-Stokes equation with Volume fraction equation with MULES algorithm by VOF method to estimate the resistance of the ship with respect to the wave speed. Additionally, it is also compute the total drag force in the direction of fluid. Therefore, by considering the wave with the initial velocity $U = 1.668m/s$ with type stokes II, I ran the simulation for the final time 20 and time step $1e-5$. After doing some postprocessing with ParaView, some parameters like velocity profile, modifid preassure, alpha.water (volume fraction) which give us better intuis-

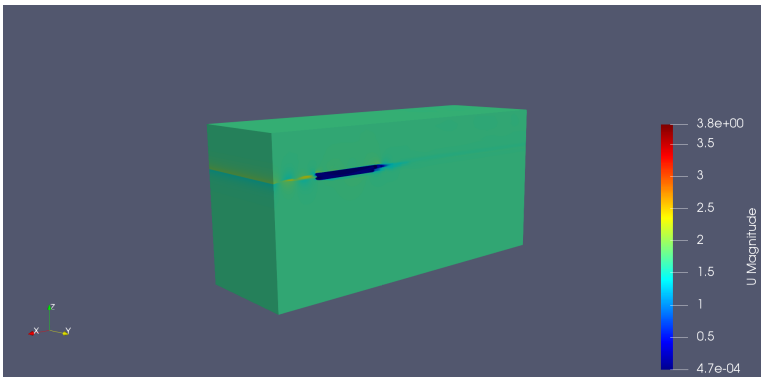


Figure 5.3: Velocity profile

sion about the simulation are shown as follows: Figure 5.3 shows the velocity profile in the whole domain during the running time.Also after making a slice in specific part of the domain figure 5.4 depicts the the velocity in the middle of the domain.in both pictures we can see the pattern of fluid which is affected by wave velocity.

The Modified pressure also shows in figure 5.5 which obviously has been changing during the time step especially at the interface.The conversion of volume fraction is shown in figure 5.6 which α is changing between 0 and 1 at the interface. By this picture one can easily figure out the red part consists of water and the blue part is covered by air. The figures 5.7 and 5.8 indicate the modified pressure and volume fraction changes on the hull. Clearly the left side of the hull facing much changes rather than the right side of the hull. This is because of the wave which is coming from inlet. To have better view of the direction of flow the pictures 5.9 and 5.10 indicate the custom stream line of velocity specially at inlet and stream line of velocity at interface, respectively.Besides, the figure 5.11 shows the stream line of volume fraction function which clearly can show the prominence and indentation of the free surface.

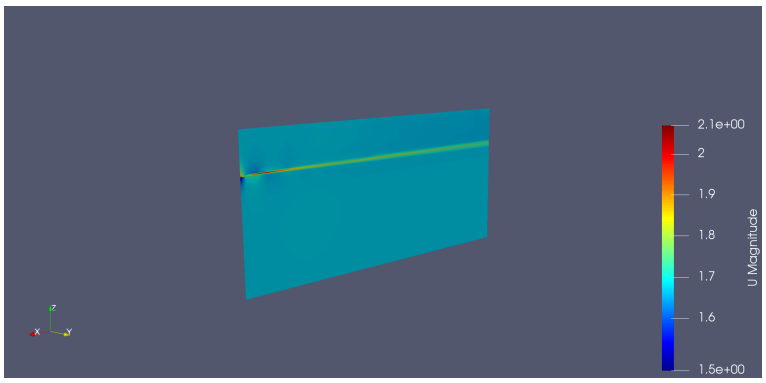


Figure 5.4: Velocity inside the domain by making slice

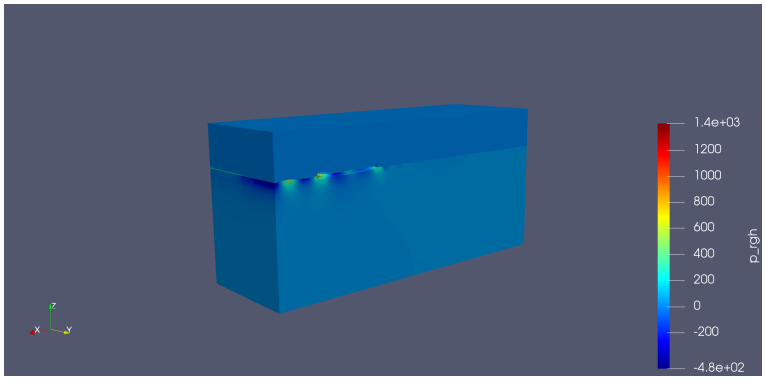


Figure 5.5: Modified preassure profile

Figure 5.6: Volum Fraction alpha.water Profile

Figure 5.7: Modified Pressure on hull

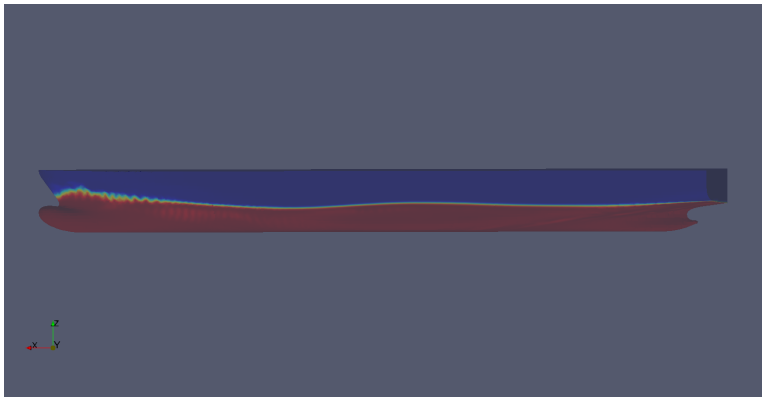


Figure 5.8: Volume fraction alpha.water on hull

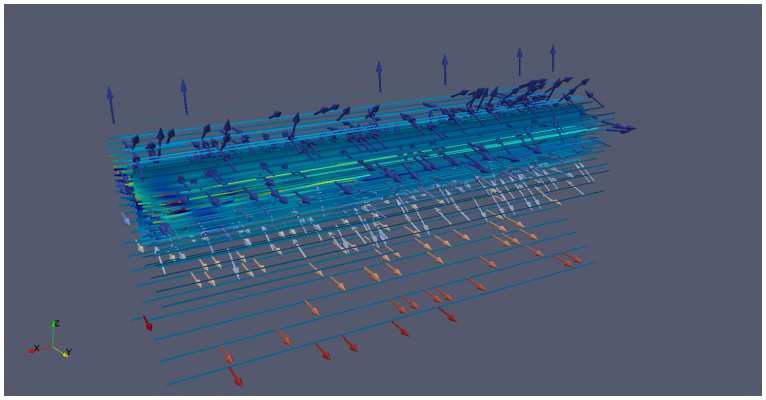


Figure 5.9: Custom streamline of inlet

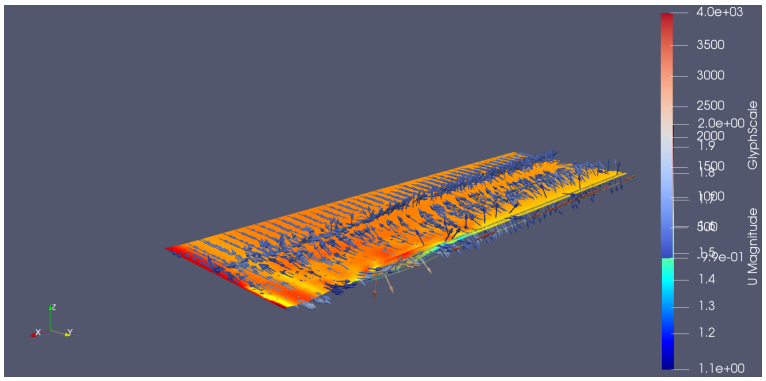


Figure 5.10: Velocity stream line

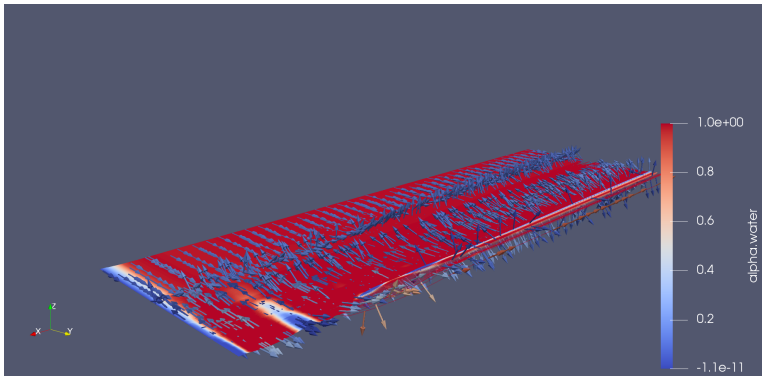


Figure 5.11: stream line of volume fraction alpha.water profile

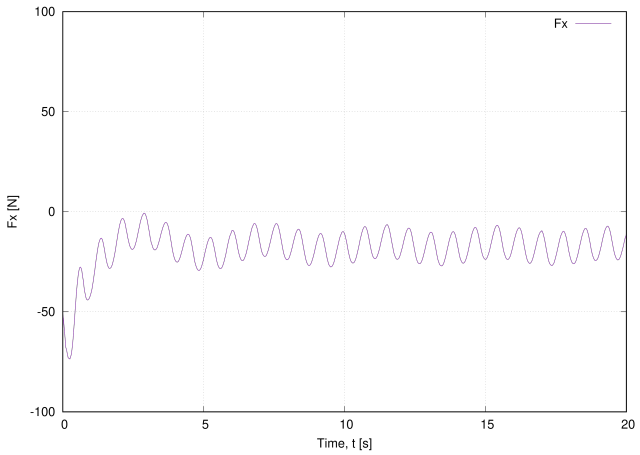


Figure 5.12: Total Drag Force in x-direction

In this simulation DTCHullwave consists of wave with the velocity $U = 1.668m/s$ and using the interDyMFoam solver which is run for two different final time 20 and 40 with the time step $1e-5$. Therefore, the figures 5.12 and 5.13 report the total drag force in x-direction, which is coming from the pressure and viscosity in x direction for the final time 20 and 40, respectively. In both figures there are some oscillation but in figure 5.12 the oscillation is changing during the time steps while in figure 5.13 after time 20 the oscillation has the same pattern till the final time 40. Similarly, the simulation of DTCHull [49] is based on meshing and simulating the flow field around a DTCHull model with interFoam solver at a velocity equal to $U = 1.668m/s$. In this simulation there is no wave and it converges after 4000 iterations with time step 1. Here, the total drag force in x-direction are shown in figure 5.14.

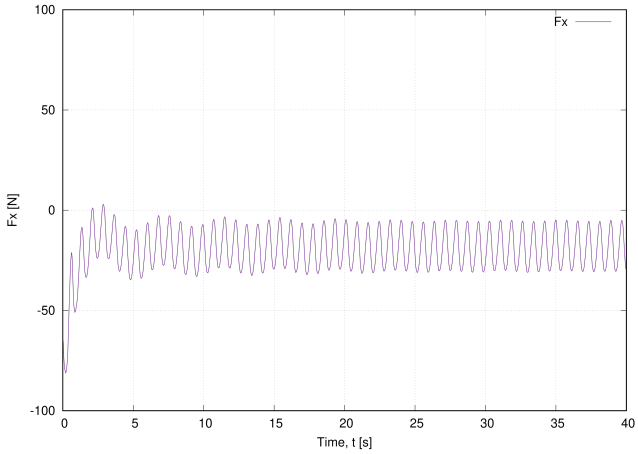


Figure 5.13: Total Drag Force in x-direction

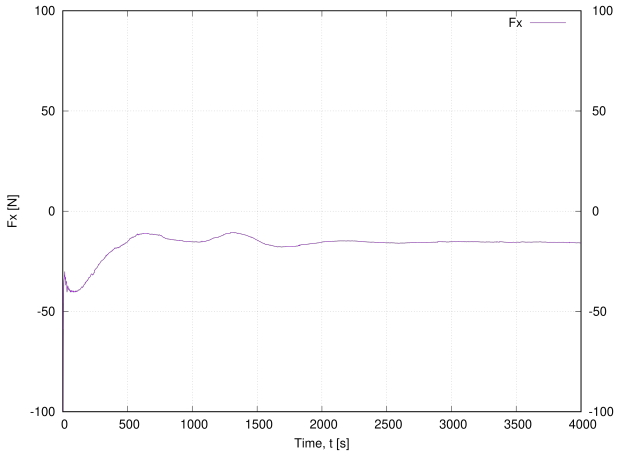


Figure 5.14: Total Drag Force in x-direction

Figure 5.15: Velocity profile with isoAdvector

So far the above simulation DTCHullwave has been used the interDyM-solver with MULES algorithm for solving the volume fraction function which the result is stable after some times. To go further, this simulation also has been run with another solver interFlow which uses isoAdvector algorithm to solve the volume fraction function at interface with its own specific parameters. By investigation the simulation's result which is shown in figure 5.15 is not stable with isoAdvector rather than the one which has used MULES algorithm. But there are some physical problem which only solving the volume fraction function by isoAdvector which is really faster and keep the simulation much more stable(there is an example in appendix which is solved with isoAdvector). In fact, the isoAdvector algorithm is working better than the MULES one only in some cases [47]. However, for future work one can develop this algorithm in such a way to get more resistance of a ship during the time steps. Also, one can try different type of ships to see how do the different geometries affect the resistance, solving by considering longer time and smaller time step. Moreover investigating the effect of different wave velocity for such a multiphase simulations might be another possibility in this direction.

Bibliography

[1] W. K. Melville, "The role of surface-wave breaking in air-sea interaction," *Annual Review of Fluid Mechanics*, vol. 28, no. 1, pp. 279–321, 1996. [Online]. Available: https://doi.org/10.1146/annurev.fl.28.010196.001431

[2] W. J. Rider and D. B. Kothe, "Reconstructing volume tracking," *Journal of Computational Physics*, vol. 141, no. 2, pp. 112 – 152, 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002199919895906X

[3] D. Greaves, "A quadtree adaptive method for simulating fluid flows with moving interfaces," *Journal of Computational Physics*, vol. 194, no. 10, pp. 35–56, 2 2004.

[4] M. Rudman, "Volume-tracking methods for interfacial flow calculations," *International Journal for Numerical Methods in Fluids*, vol. 24, no. 7, pp. 671–691, 1997. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291097-0363%2819970415%2924%3A7%3C671%3A%3AAID-FLD508%3E3.0.CO%3B2-9

[5] S. Aliabadi and K. Shujaee, "Free-surface flow simulations using parallel finite element method," *Simulation*, vol. 76, no. 5, pp. 257–262, 2001.

[6] J. D. Anderson, "Computational fluid dynamics (the basics with application)," in *McGraw-Hill Book Co. Singapore.*, 1995.

[7] H. K. Versteeg and W. Malalasekera, "An introduction to computational fluid dynam- ics:(the finite volume approach)." in *Pearson Prentice Hall, London.*, 1995.

[8] G. Galdi, "An introduction to the navier-stokes initial-boundary value problem." in *In Fun- damental Directions in Mathematical Fluid Mechanics, pages 198, Switzerland. Verlag*, 2000.

[9] J. H. Ferziger and M. Perić, *Computational Methods for Fluid Dynamics*, 2nd ed. Berlin: Springer, 1999.

[10] A. J. Chorin, "The numerical solution of the navier-stokes equations for an incompressible fluid," *Bull. Amer. Math. Soc.*, vol. 73, no. 6, pp. 928–931, 11 1967. [Online]. Available: https://projecteuclid.org:443/euclid.bams/1183529112

[11] M. Owkes, "A guide to writing your first cfd solver," Tech. Rep., 2017. [Online]. Available: http://www.montana.edu/mowkes/research/source-codes/GuideToCFD.pdf

[12] D. o. M. E. METU, "Matlab code for the projection method and bc details for the lid-driven cavity problem," Tech. Rep., 2016. [Online]. Available: http://courses.me.metu.edu.tr/courses/me705/files/ME2070520Fall/202016/20Handout/208.pdf

[13] G. Chen, C. Kharif, S. Zaleski, and J. Li, "Two-dimensional Navier–Stokes simulation of breaking waves," *Physics of Fluids*, vol. 11, no. 1, pp. 121–133, 1999. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01307123

[14] J. Ferziger and M. Peric, "Computational methods for fluid dynamics." in *Springer-Verlag Berlin Heidelberg, Germany.*, 1999.

[15] S. O. Unverdi and G. Tryggvason, "A front-tracking method for viscous, incompressible, multi-fluid flows," *Journal of Computational Physics*, vol. 100, no. 1, pp. 25 – 37, 1992. [Online]. Available: http://www.sciencedirect.com/science/article/pii/002199919290307K

[16] H. Terashima and G. Tryggvason, "A front-tracking/ghost-fluid method for fluid interfaces in compressible flows," *J. Comput. Physics*, vol. 228, pp. 4012–4037, 2009.

[17] O. Gloth, D. Hänel, L. T. Tran, and R. Vilsmeier, "A front tracking method on unstructured grids," 2003.

[18] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y.-J. Jan, "A front-tracking method for the computations of multiphase flow," *Journal of Computational Physics*, vol. 169, no. 2, pp. 708 – 759, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999101967269

[19] S. W. Welch, "Local simulation of two-phase flows including interface tracking with mass transfer," *Journal of Computational Physics*, vol. 121, no. 1, pp. 142 – 154, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999185711850

[20] H. H. Hu, N. Patankar, and M. Zhu, "Direct numerical simulations of fluid–solid systems using the arbitrary lagrangian–eulerian technique," *Journal of Computational Physics*, vol. 169, no. 2, pp. 427 – 462, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999100965926

[21] M. S. Kim and W. I. Lee, "A new vof-based numerical scheme for the simulation of fluid flow with free surface. part i: New free surface-tracking algorithm and its verification," *International Journal for Numerical Methods in Fluids*, vol. 42, no. 7, pp. 765–790, 2003. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.553

[22] W. Tao, D. Cai, X. Yan, N. Ken-ichi, and B. Lembege, "Scalability analysis of parallel particle-in-cell codes on computational grids," *Computer Physics Communications*, vol. 179, no. 12, pp. 855 – 864, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010465508002518

[23] M. Sussman, P. Smereka, and S. Osher, "A level set approach for computing solutions to incompressible two-phase flow," *Journal of Computational Physics*, vol. 114, no. 1, pp. 146 – 159, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999184711557

[24] C. Hirt and B. Nichols, "Volume of fluid (vof) method for the dynamics of free boundaries," *Journal of Computational Physics*, vol. 39, no. 1, pp. 201 – 225, 1981. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0021999181901455

[25] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations," *Journal of Computational Physics*, vol. 79, no. 1, pp. 12 – 49, 1988. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0021999188900022

[26] C. Devals, M. Heniche, F. Bertrand, P. A. Tanguy, and R. E. Hayes, "A two-phase flow interface capturing finite element method," *International Journal for Numerical Methods in Fluids*, vol. 53, no. 5, pp. 735–751, 2007. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.1303

[27] R. Scardovelli and S. Zaleski, "Interface reconstruction with least-square fit and split eulerian–lagrangian advection," *International Journal for Numerical Methods in Fluids*, vol. 41, no. 3, pp. 251–274, 2003. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.431

[28] S. Mirjalili, S. S. Jain, and M. Dodd, "Interface-capturing methods for two-phase flows: An overview and recent developments," *Center for Turbulence Research Annual Research Briefs*, vol. 2017, pp. 117–135, 2017.

[29] W. F. Noh and P. Woodward, "Slic (simple line interface calculation)," in *Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics June 28 – July 2, 1976 Twente University, Enschede*, A. I. van de Vooren and P. J. Zandbergen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1976, pp. 330–340.

[30] W. Rider and D. Kothe, "Comments on modeling interfacial flows with volume-of-fluid methods," 02 1995.

[31] D. Youngs, "Time-dependent multi-material flow with large fluid distortion," *Num. Method Fluid Dyn.*, vol. 24, pp. 273–285, 01 1982.

[32] M. Rudman, "Volume-tracking methods for interfacial flow calculations," *International Journal for Numerical Methods in Fluids*, vol. 24, no. 7, pp. 671–691, 1997. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291097-0363%2819970415%2924%3A7%3C671%3A%3AAID-FLD508%3E3.0.CO%3B2-9

[33] M. J. Ketabdari, "Free surface flow simulation using vof method," in *Numerical Simulation*, R. Lopez-Ruiz, Ed. Rijeka: IntechOpen, 2016, ch. 15. [Online]. Available: https://doi.org/10.5772/64161

[34] S. Saincher and J. Banerjee, "An operator-split plic-vof algorithm for two-phase flow: Issues in implementation," 12 2012.

[35] R. Scardovelli and S. Zaleski, "Direct numerical simulation of free-surface and interfacial flow," *Annual review of fluid mechanics*, vol. 31, no. 1, pp. 567–603, 1999.

[36] Vachaparambil and Einarsrud, "Corrections to: Comparison of surface tension models for the volume of fluid method," *Processes*, vol. 8, p. 152, 01 2020.

[37] H. K. Versteeg and W. Malalasekera, *An Introduction to Computational Fluid Dynamics: The Finite Volume Method.* Harlow, England: Pearson Education Ltd, 2007.

[38] F. Moukalled, L. Mangani, M. Darwish *et al.*, *The finite volume method in computational fluid dynamics*. Springer, 2016, vol. 113.

[39] C. Greenshields, "Openfoam v6 user guide: 2.3 breaking of a dam," 2018. [Online]. Available: https:

//cfd.direct/openfoam/user-guide/v6-dambreak/#:~:text=The%20cAlpha%20keyword%20is%20a,is%20employed%20in%20this%20example.

[40] OpenFOAMWiki, "Dynamicmeshdict," 2016. [Online]. Available: https://openfoamwiki.net/index.php/DynamicMeshDict

[41] O. L. E. Group), "Standard boundary conditions," 2018. [Online]. Available: https://www.openfoam.com/documentation/user-guide/standard-boundaryconditions.php

[42] C. Greenshields, "Openfoam v6 user guide: 4.5 solution and algorithm control," 2018. [Online]. Available: https://cfd.direct/openfoam/user-guide/v6-fvsolution/

[43] T. G. Drozda, "Computational fluid mechanics and heat transfer," *AIAA Journal*, vol. 51, no. 11, pp. 2751–2751, 2013. [Online]. Available: https://doi.org/10.2514/1.J052452

[44] T. Holzmann, *Mathematics, Numerics, Derivations and OpenFOAM®*, 11 2019.

[45] Z. Lin, W. Yang, H. Zhou, X. Xu, L. Sun, Y. Zhang, and Y. Tang, "Communication optimization for multiphase flow solver in the library of openfoam," *Water*, vol. 10, no. 10, p. 1461, 2018.

[46] B. E. Larsen, D. R. Fuhrman, and J. Roenby, "Performance of interfoam on the simulation of progressive waves," *Coastal Engineering Journal*, vol. 61, no. 3, p. 380–400, May 2019. [Online]. Available: http://dx.doi.org/10.1080/21664250.2019.1609713

[47] J. Roenby, H. Bredmose, and H. Jasak, "A computational method for sharp interface advection," *Royal Society Open Science*, vol. 3, 2016.

[48] C. Greenshields, "Openfoam v6 user guide: 4.4 numerical schemes," 2018. [Online]. Available: https://cfd.direct/openfoam/user-guide/v6-fvschemes/

[49] H. Weller, "Dtchull and dtchullwave," 2017. [Online]. Available: https://github.com/OpenFOAM/OpenFOAM-5.x/tree/master/tutorials/multiphase

[50] J. Roenby, "isoadvector," 2019. [Online]. Available: https://github.com/isoAdvector/isoAdvector

Appendix A

OpenFoam Dictionary

A.1 interDyMFoam solver

```
/*---------------*- C++ -*-----------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version: 5
|
|   \\  /    A nd    |Web: www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * *  * * * * * * * * * * //

application     interDyMFoam;

startFrom       latestTime;

startTime       0;
```

```
stopAt          endTime;

endTime         40;

deltaT          0.00001;

writeControl    adjustableRunTime;

writeInterval   1;

purgeWrite      0;

writeFormat     ascii;

writePrecision  9;

writeCompression uncompressed;

timeFormat      general;

timePrecision   9;

runTimeModifiable yes;

adjustTimeStep  yes;

maxCo           10;
maxAlphaCo      5;
maxDeltaT       0.015;

libs            ("libwaves.so");

functions
{


forces
      {
        type            forces;
        libs            ("libforces.so");
        patches         (hull);
        rhoInf          998.8;
```

```
        log             on;
        writeControl timeStep;
        writeInterval 2;
        CofR            (2.929541 0 0.2);
    }

}


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

```
/*--------------*- C++ -*----------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:5
|
|   \\  /    A nd           | Web: www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * //

solvers
{
    "alpha.water.*"
    {
        nAlphaCorr      3;
        nAlphaSubCycles 1;
        cAlpha          1;
        icAlpha         0;

        MULESCorr       yes;
        nLimiterIter    10;
        alphaApplyPrevCorr  yes;

        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-10;
        relTol          0;
        minIter         1;
    }

    "pcorr.*"
```

```
    {
        solver          GAMG;

        smoother        DIC;

        tolerance       0.1;
        relTol          0;
    };

    p_rgh
    {
        solver          GAMG;

        smoother        DIC;

        tolerance       1e-7;
        relTol          0.001;
    };

    p_rghFinal
    {
        $p_rgh;
        relTol          0;
    }

    "(U|k|omega).*"
    {
        solver          smoothSolver;

        smoother        symGaussSeidel;
        nSweeps         2;

        tolerance       1e-8;
        relTol          0.001;
        minIter         1;
    };
}

PIMPLE
{
    momentumPredictor yes;

    nOuterCorrectors 3;
```

```
    nCorrectors       2;
    nNonOrthogonalCorrectors 2;

    correctPhi        yes;
    moveMeshOuterCorrectors yes;
    turbOnFinalIterOnly yes;
}

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

cache
{
    grad(U);
}


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

```
/*---------------*- C++ -*----------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version: 5
|
|   \\  /    A nd           | Web: www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * *  * * * * * * * * //

solvers
{
    "alpha.water.*"
    {
        nAlphaCorr      3;
        nAlphaSubCycles 1;
        cAlpha          1;
        icAlpha         0;

        MULESCorr       yes;
        nLimiterIter    10;
        alphaApplyPrevCorr  yes;

        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-10;
        relTol          0;
        minIter         1;
    }
```

```
    "pcorr.*"
    {
        solver          GAMG;

        smoother        DIC;

        tolerance       0.1;
        relTol          0;
    };

    p_rgh
    {
        solver          GAMG;

        smoother        DIC;

        tolerance       1e-7;
        relTol          0.001;
    };

    p_rghFinal
    {
        $p_rgh;
        relTol          0;
    }

    "(U|k|omega).*"
    {
        solver          smoothSolver;

        smoother        symGaussSeidel;
        nSweeps         2;

        tolerance       1e-8;
        relTol          0.001;
        minIter         1;
    };
}

PIMPLE
{
    momentumPredictor yes;
```

```
    nOuterCorrectors 3;
    nCorrectors      2;
    nNonOrthogonalCorrectors 2;

    correctPhi       yes;
    moveMeshOuterCorrectors yes;
    turbOnFinalIterOnly yes;
}

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

cache
{
    grad(U);
}


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

A.2 interFlow solver

```
/*---------------*- C++ -*----------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version: 5
|
|   \\  /    A nd      | Web: www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * //

application     interFlow;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         40;

deltaT          0.00001;

writeControl    adjustableRunTime;

writeInterval   1;

purgeWrite      0;
```

```
writeFormat      ascii;

writePrecision   9;

writeCompression uncompressed;

timeFormat       general;

timePrecision    9;

runTimeModifiable yes;

adjustTimeStep   yes;

maxCo            10;
maxAlphaCo       5;
maxDeltaT        0.015;

libs             ("libwaves.so");

functions
{



forces
      {
        type            forces;
        libs            ("libforces.so");
        patches         (hull);
        rhoInf          998.8;
        log             on;
        writeControl timeStep;
        writeInterval 2;
        CofR            (2.929541 0 0.2);
    }

}


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

```
/*--------------*- C++ -*----------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version: 5
|
|   \\  /    A nd          | Web: www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * ** * * * * * //

solvers
{
    "alpha.water.*"
    {
        interfaceMethod "isoAdvector";
        isoFaceTol      1e-6;
        surfCellTol     1e-6;
        snapTol         1e-12;
        nAlphaBounds    3;
        clip            true;


        nAlphaCorr      3;
        nAlphaSubCycles 1;
        cAlpha          1;
        //icAlpha         0;

        MULESCorr       yes;
        nLimiterIter    10;
        alphaApplyPrevCorr  yes;
```

```
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-10;
        relTol          0;
        minIter         1;
    }

    "pcorr.*"
    {
        solver          GAMG;

        smoother        DIC;

        tolerance       0.1;
        relTol          0;
    };

    p_rgh
    {
        solver          GAMG;

        smoother        DIC;

        tolerance       1e-7;
        relTol          0.001;
    };

    p_rghFinal
    {
        $p_rgh;
        tolerance       1e-08;
        relTol          0;
    }

    "(U|k|omega).*"
    {
        solver          smoothSolver;

        smoother        symGaussSeidel;
        nSweeps         2;

        tolerance       1e-8;
        relTol          0.001;
```

```
        minIter          1;
    };
}

PIMPLE
{
    momentumPredictor yes;

    nOuterCorrectors 3;
    nCorrectors      2;
    nNonOrthogonalCorrectors 2;

    correctPhi       yes;
    moveMeshOuterCorrectors yes;
    turbOnFinalIterOnly yes;
}

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

cache
{
    grad(U);
}


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

A.3 isoAvector

Here there is an example which solves the volume fraction function with isoAvector in 2D. The figure A.1 show its changes in different time steps[50].

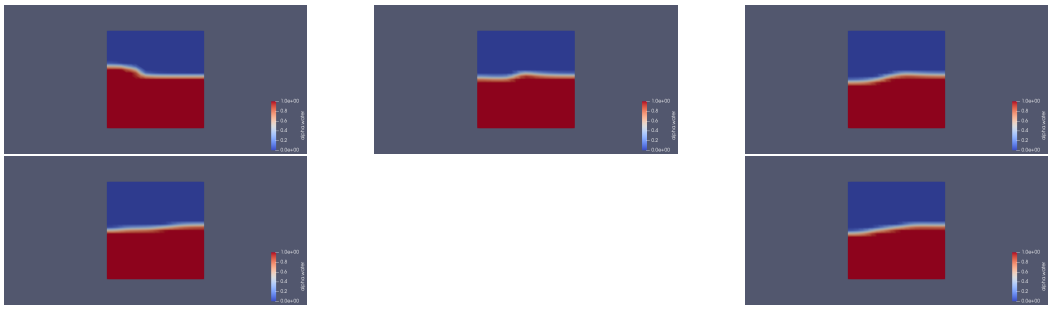


Figure A.1: Different time steps of volume fraction at interface