Development of a Novel Accurate VOF Method for Incompressible Two-phase Fluid Flow

Submitted in partial fulfillment of the requirements
of the degree of

Doctor of Philosophy

by

Sourabh Pramod Bhat
Roll No: 134010004

Supervisor
Prof. Dr. J. C. Mandal

Department of Aerospace Engineering
INDIAN INSTITUTE OF TECHNOLOGY BOMBAY
April 2020

Dedicated to my ever supportive **Mother**.

Thesis Approval

This thesis entitled "**Development of a Novel Accurate VOF Method for Incompressible Two-phase Fluid Flow**" by Sourabh Bhat is approved for the degree of Doctor of Philosophy.

Supervisor
(Prof. J. C. Mandal)

Examiner
(Prof. Shivasubramanian Gopalakrishnan)

Examiner
(Prof. Jayanti Sreenivas)

Chairman
(Prof. Sanjay Mahajani)

Date: 07 December 2020

Place: Mumbai

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

Sourabh Bhat
Roll No. 134010004

Date: _____

Abstract

The focus of this work is on development of an accurate and efficient methods for capturing fluid interfaces in multiphase flows. Many of the currently available computer codes for solving multiphase flows make use of volume of fluid (VOF) method, using geometric techniques, like the piecewise linear interface calculation (PLIC). The VOF method works very well for structured Cartesian mesh. However, its extension to arbitrarily shaped unstructured mesh is difficult and inefficient. Efforts have been made by various researchers to extend the geometry based methods to unstructured mesh. This has resulted in fairly complicated interface capturing methods, which do not scale well to practical problems, due to their dependence on mesh type. The Godunov-type methods, on the other hand, are generic and work well with arbitrarily shaped cells. This is because, the flux is computed using a Riemann solver instead of cell geometry. This class of methods have been extensively studied in the field of computational fluid dynamics for solving hyperbolic partial differential equations (PDE) and are known to be accurate and efficient. The VOF method uses an additional PDE, for advection of volume fraction, which is of hyperbolic nature. Therefore, the Godunov-type methods can be used for capturing the interface in an efficient way. However, even with reasonably high-order accurate solution reconstruction, the diffusion error begins to dominate over time, resulting in smearing of fluid interfaces. This is due to the discontinuous volume fraction near the fluid interfaces, which results in artificial dissipation being added to maintain stability.

In this work, the incompressible two-phase fluid flow equations are written using the artificial compressibility formulation. Thus, the Navier-Stokes equations along with the volume fraction equation behave like a hyperbolic-parabolic system of PDEs. This allows for computation of convective fluxes using approximate Riemann solvers. The first major contribution of this research work is the design of a novel Riemann solver for incompressible two-phase flow system. The design of the new Riemann solver is based on the observation (using generalized Riemann invariant analysis) that the fluid interface behaves like a contact wave, traveling at the speed of the fluid. Exploiting this fact, a three wave Riemann solver is designed similar to the Harten Lax and van-Leer with contact (HLLC) Riemann solver. The new solver calculates an accurate convective flux and reduces numerical dissipation. The efficacy of the new solver is evaluated by systematically solving various problems, testing different aspects of the solver. Additionally, the Riemann solver is derived for a three-dimension coordinate

system so that it can be readily applied to practical problems. The three-dimensional capability is tested by solving a complex bubble merging non-axisymmetric problem.

The second major contribution of this work is the development of an accurate and extensible interface reconstruction method. It has been observed that, in spite of reduced diffusion by the new Riemann solver, the interface smears out over long time of evolution. This can be overcome to some extent by using interface compression algorithms. Hence interface compression was utilized with the Riemann solver in the first part of this work. To eliminate the interface smearing however, the interface needs to be treated in a special way, by using sub-cell reconstruction, as in the case of the PLIC method. Considering the limitations of the PLIC method to handle arbitrary meshes, a new generic method of interface reconstruction is developed. The new method is inspired from the marching-cubes method which adapts easily to any polygonal or polyhedral cells. Additionally, the new reconstruction method is designed to work with Riemann solvers, so that there is no need for geometric flux computation like in the case of PLIC method. This makes the method easy to adapt to various cell-shapes and the method works well with unstructured mesh. Moreover, the method can be simplified for triangular cells by utilizing the symmetries of the triangle. The efficacy of the new interface reconstruction method is evaluated by solving various scalar problems on unstructured mesh and by comparing the results to exact solutions and interface compression algorithm. The applicability of the new method in two-phase flows is then demonstrated by solving transient two-phase flow problems.

Keywords: *Incompressible two-phase flows, Volume of fluid method, Artificial compressibility formulation, Finite volume method, Dual-time stepping algorithm, Novel Riemann solver, HLLC-VOF, Novel interface reconstruction, Computational fluid dynamics, CFD code based on object-oriented principles.*

Contents

List of Figures

List of Tables

Nomenclature

C	:	volume fraction
x, y, z	:	Cartesian coordinates
t	:	real time
Δt	:	real time step
τ	:	pseudo time
$\Delta\tau$	:	pseudo time step
i	:	index used to identify the central cell
j	:	index used to identify the neighboring cells or cell faces
$\mathbf{U}$	:	vector of conservative variables
$\mathbf{W}$	:	vector of real variables
$\mathbf{F}, \mathbf{G}, \mathbf{H}$	:	x, y, z components of convective fluxes
$\mathbf{F_v}, \mathbf{G_v}, \mathbf{H_v}$	:	x, y, z components of viscous fluxes
$\mathbf{T}$	:	surface tension flux
$\mathbf{F}_c$	:	interface compression flux
$\mathbf{S}$	:	source term
g_x, g_y, g_z	:	x, y, z components of acceleration due to gravity
p	:	hydrostatic pressure
β	:	artificial compressibility parameter
$\vec{V}$	:	fluid velocity vector
u, v, w	:	x, y, z components of fluid velocity
ρ	:	density of the fluid
μ	:	dynamic viscosity of the fluid
σ	:	coefficient of surface tension
Ω_P	:	volume of cell P
Γ_m	:	surface area of face m
Λ_c	:	convective spectral radius
Λ_v	:	viscous spectral radius
$\hat{n}$	:	unit normal vector to the cell face
n_x, n_y, n_z	:	x, y, z components of unit normal vector to cell face
$\hat{n}_i$	:	unit normal vector to fluid interface
$\vec{V}_c$	:	interface compression velocity
η	:	interface compression coefficient
λ	:	eigenvalue

$\hat{x}, \hat{y}, \hat{z}$	:	cell face aligned coordinate system
S	:	wave speed in a Riemann solver
k	:	wave number
E_o	:	Eötvös number
M	:	Morton number

Superscripts

$\overline{\mathbf{U}}$	:	cell averaged value
$\hat{\mathbf{U}}$	:	value in the face aligned coordinate system

Subscripts

f	:	value at the cell face
L	:	left state value of the Riemann problem, or left wave speed when used as S_L
R	:	right state value of the Riemann problem, or right wave speed when used as S_R
$*L$	:	state between left and intermediate wave of Riemann solution
$*R$	:	state between right and intermediate wave of Riemann solution
$*$	:	intermediate wave speed, used as S_*

Abbreviations

CFD	:	Computational Fluid Dynamics
VOF	:	Volume of Fluid
LSM	:	Level Set Method
PDE	:	Partial Differential Equation
MAC	:	Marker and Cell
PLIC	:	Piecewise Linear Interface Calculation
SLIC	:	Simple Line Interface Calculation
SDWLS	:	Solution Dependent Weighted Least Square
ENO	:	Essentially Non-Oscillatory
WENO	:	Weighted Essentially Non-Oscillatory
HLL	:	Harten Lax and van-Leer
HLLC	:	Harten Lax and van-Leer with Contact
VTK	:	Visual Toolkit

Chapter 1

Introduction

In the past few decades, computers have become faster and more accessible for scientific studies. Even desktop and laptop computers have become powerful enough to deal with fairly complex problems in different fields of sciences. This has led to development of new, efficient and robust computational methods to solve many problems faced by the mankind, which were unsolvable before. A lot of the physical processes in the nature around us, can be mathematically described by partial differential equations (PDEs). The PDEs can be formulated using fundamental conservation principles and many of such governing equations are available in the scientific literature. However, we still lack "suitable" methods to solve the PDEs and therefore, unable to analyze the physical processes with sufficient accuracy. One such example, where the equations were available far before the techniques to solve them, are the Navier-Stokes equations. These set of equations define conservation laws, which govern the flow of viscous fluids. These equations were described in the early 19th century, but we still struggle to solve these equations for complex flows, even two centuries later.

The Navier-Stokes equations are used to model gas and liquid flows. If the fluid flow consists of a single fluid, either gas-phase or liquid-phase, then the flow is called a "single-phase flow". On the other hand, if two immiscible fluids coexist in the flow, separated by interfaces, then such a flow is called a "two-phase flow". Two-phase flows are almost everywhere around us. Our daily experiences are filled with examples of two-phase flows, such as drinking water from a glass involves an interface separating the water from the air, washing involves some liquid-air interfaces with surfactants to minimize the surface tension, and almost every other day-to-day activity involve two-phase flows. Many engineering and design industries thrive on understanding of two-phase flows, like the ship-building industry for building sea-worthy vessels [1], automotive and power industry for designing high-performing engines, heating and cooling systems [2], manufacturing industry for lubrication and cooling of machine components [3], metal casting industry for designing better moulds and understanding of the solidification process [4], and many more. Our space-exploration endeavors have called for reliable multiphase computer simulations, to understand the behavior of fluids under micro-gravity, as such physical experiments are difficult to setup on the

Earth's surface [5]. Many biological processes, in human bodies, occur due to liquids that phase-separate from the cytoplasm [6]. Such processes can be understood through computer simulations of two-phase flows [7]. Another interesting application of two-phase flows is in the entertainment industry, in movies and computer games, where such simulations are used for rendering photo realistic effects [8, 9]. Here, only a few applications of two-phase flows are enlisted. However, there are innumerable problems, in different fields of engineering and science involving two-phase flows. Therefore, it is of paramount importance to develop robust, accurate and reliable numerical methods to solve two-phase flow problems.

1.1 Background

The Navier-Stokes equations, for single-phase flows, are routinely solved numerically by engineers and scientists using standard software packages. The numerical methods for such flows are well established and mature enough, that the industries can rely on them for design and development of new products. However, the same cannot be said about two-phase flows, as these numerical methods are still under active research and development. The initial attempts towards solving free-surface flows can be seen since 1965 when the method called marker-and-cell (MAC) was developed, at Los Alamos National Laboratory by Harlow and Welch [10]. In the MAC method, the presence of the fluid is indicated by marker particles. The MAC method is a combination of Eulerian and Lagrangian techniques, where a mesh consists of non-moving cells and the particles are advected using Lagrangian methods from one cell to another. In the MAC method, a significantly large number of marker particles are necessary to correctly capture the free surface, thus increasing the storage requirements, especially in three-dimensional problems [11]. Also, there is a need for application of boundary condition at the free surface, as the flow is solved only in the region with markers. This problem is somewhat alleviated when solving problems involving two-fluids [12], with flow variables being solved in both fluids. Nevertheless, the problem of insufficient marker particles in some pockets of the flow and high storage requirements still remained to be resolved.

To overcome the shortcomings of the MAC method, the VOF method has been developed by Hirt and Nichols [11]. In the VOF method, the distributed fluid markers in the MAC method are replaced by a marker function for volume fraction. The cell averaged volume fraction is saved in each cell. Hence, the marker function assumes a value of "1" in the cell filled with the first fluid and "0" in the cell filled with the second fluid. A cell having a value between 1 and 0 indicates the presence of the interface in the cell. A naive implementation of the VOF method, using an upwind scheme, will result in excessive numerical dissipation, causing smearing of the fluid interface. To overcome this issue, normally the interface is reconstructed using local volume fractions, by either the simple-line-interface-calculation (SLIC) [11, 13] or piecewise-linear-interface-

calculation (PLIC) [14]. The material properties such as fluid density and viscosity are calculated from the volume fraction in the cell.

Another approach, called the front tracking method, is introduced by Unverdi and Tryggvason [15] in 1992. This is a hybrid method combining features of MAC and VOF. In this method, connected marker particles are placed at the interface, and are advected by the local velocity field. The material properties are calculated by using a function similar to VOF, which is reconstructed using the connected marker particles at the interface. This is a very attractive method as it produces superior results when the interface does not distort a lot. However, special treatment is necessary when interfaces merge or break apart.

The discontinuity of volume fraction at the interface, in the VOF method, led to inaccuracies in calculation of interface normals, which are necessary for computing surface tension force. Hence, the level set method for fluid flow problems was born. Even though the level-set method existed before, it was introduced for fluid flow problems by Sussman, *et al.* [16] in 1994. In this method a smooth and continuous function is used, unlike the VOF marker function which changes rapidly at the interface. The interface location in the level-set method is given by a fixed level of the function. The signed-distance function is a commonly used level-set function, where the level of zero identifies the interface. To calculate the material properties a smoothed Heaviside function is used, which is obtained from the level-set function. In the level-set method, the marker function used to represent the interface does not have any physical significance, hence its advection is susceptible to producing non-physical results. Such a numerical inaccuracy is manifested as non-conservation of mass in the level-set method [17].

A more recent approach, for capturing interfaces in two-phase flows, is the phase-field method. The phase-field method is widely used to study metal solidification in material science. The use of this method for simulating two-phase flows is more recent [18]. In this method an energy potential is defined using thermodynamic principles, such that the interface sharpness is maintained. The material properties are computed similar to the VOF method and all the calculations are performed on a fixed grid. Even though the method has its roots in metal solidification, where the energy potential has physical significance, the adopted method for two-phase flows uses artificially chosen parameters [18] to maintain a sharp interface.

The methods, which are briefly described above, are designed to do only one thing: which is to represent the interface, so that the fluid properties can be correctly computed in the different regions of the flow. Once the fluid properties are computed, all the above methods make use of the one-fluid model to evolve the solution. Presently available methods are deficient in one way or other. Therefore, accurate simulation of two-phase flows still remain a challenge. For example, the volume of fluid method conserves mass exactly, but it is difficult to use with non-Cartesian meshes [19–21]. The front tracking method is very accurate for simple interfaces, but becomes chal-

lenging and inaccurate when it comes to breaking or merging interfaces [22]. The level-set method is very easy to extend to non-Cartesian mesh, but it does not conserve mass [17]. The phase-field method modifies the marker function based on thermodynamic principles with artificially chosen parameters to maintain a sharp interface, therefore it is not clear if the added sophistication enhances any accuracy [23]. In short, the currently available methods for simulating two-phase flows still need improvements to obtain better accuracy and efficiency in dealing with more general real world problems.

1.2 Motivation and research objectives

The VOF method is a very popular method and it is implemented in many software, both commercial and open-source ones. The VOF method is a volume tracking method, where the volume fraction of one of the fluids is used as the marker function. The popularity of the VOF method is not only because of its maturity, but also because of the following attractive features:

1. The VOF method conserves mass "exactly" by design, as the volume fraction is analogous to mass fraction, in case of incompressible fluid.
2. The material properties can be directly computed using the volume fraction.
3. The VOF method automatically captures complex phenomena such as interface breaking and merging, without any need of additional treatment.
4. The volume fraction has a physical meaning and therefore allows for innovative techniques to be developed, based on intuition.

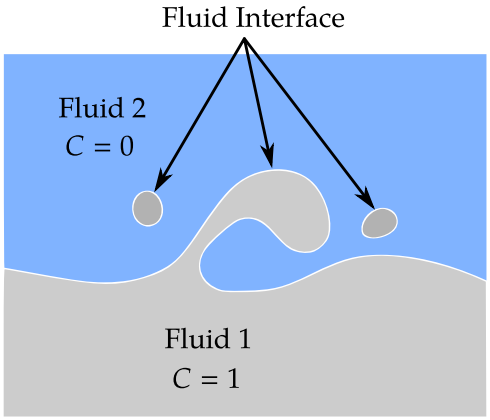


Fig. 1.1: Schematic of fluid interface and volume fraction, C, in a two-phase flow

Even though the VOF method has all the above appealing features, it also suffers from a few drawbacks. The drawbacks of the method are due to the fact that the volume fraction is discontinuous at the interface, as the value of volume fraction, C, changes from 1 to 0. This is schematically shown in Fig. 1.1. The sudden jump

in volume fraction across the interface cause the commonly used upwind methods to introduce large amount of diffusion. The diffusion error shows up as smeared interfaces over multiple cells, which is not acceptable in two-phase flow simulations. To overcome this problem the SLIC and PLIC methods have been used to reconstruct the interface, and the flux is computed using geometric advection of volume fraction. These methods accurately capture the interface and hence are widely used on structured Cartesian mesh. However, the geometric nature of these methods also mean that they are difficult to be extended to unstructured mesh. A few attempts, in literature, to define a geometric method for triangular and tetrahedral cells has led to very complicated algorithms [20, 21, 24, 25].

The aim of this research work is to improve the VOF method by retaining its good features and overcoming its limitations. The approach towards achieving these research objectives is summarized in the following two points:

1. The geometric method used for flux computation in the PLIC method, is the main reason for it being limited to Cartesian meshes. This limitation can be overcome by using a Riemann solver for convective flux in an artificial compressibility framework. Since Riemann solvers are not limited to Cartesian mesh, the method would automatically work for any arbitrarily shaped cells. Therefore, to fulfill this objective, a new Riemann solver needs to be designed for the flow equations coupled with the VOF equation. Riemann solvers are known to add numerical diffusion to ensure stability of the solver. The numerical diffusion manifests itself as smeared interfaces over multiple cells. The designed Riemann solver, therefore also needs to add minimum possible numerical diffusion and incorporate a mechanism to maintain sharp interfaces.

2. The need for interface compression in Riemann solvers can be overcome by designing a technique to compute very high-order accurate Riemann states. This can be achieved by sub-cell interface reconstruction with the discontinuity captured within the cell, by using the marching-cubes technique. As the marching-cubes technique easily extends to any convex hull, the method of reconstruction will be extensible to complex cell shapes. The method can be combined with the Riemann solver designed in the first part of the work to produce accurate two-phase flow simulations.

The first objectives is met by designing a novel Riemann solver, called HLLC-VOF, with interface compression. The second objective is achieved by developing a new technique for interface reconstruction and flux computation inspired from the marching-cubes technique. The newly developed solver and technique are thoroughly tested by solving several standard test problems reported in the literature. The test problems are chosen such that only a few aspects of the solver are examined at a time. The solutions are compared with theoretical, experimental, or numerical results available in the literature.

1.3 Organization of the thesis

In this chapter, the features and shortcomings of the currently available solvers for incompressible two-phase flows are briefly discussed, to identify clear research objectives. The next chapter discusses the literature, by logically classifying the different areas of research related to two-phase flows. Chapter 3 covers the mathematical and numerical background, which would be necessary for understanding the research work presented in the remaining part of the thesis. The chapter 4 presents the first set of contributions from this work, which are two contact preserving Riemann solvers of HLLC-type and Roe-type, respectively. The second contribution from this work is the interface reconstruction technique, which is presented in chapter 5. The final chapter of the thesis, chapter 6, draws important conclusions and presents the research contributions that resulted from this work.

Chapter 2

Review of Literature

"If I have seen further it is by standing on the shoulders of Giants" – Sir Isaac Newton, 1675.

Before going ahead with the present research work, the existing literature is studied to understand the current progress of science in the field of incompressible two-phase flows and related numerical methods. Numerical simulations and study of two-phase flows have a long history. Research in this area dates back to five or six decades, and hence there is an enormous amount of literature available in this field. This chapter is a sincere effort to cover the most relevant and important aspects of the presently available numerical methods in the field of incompressible two-phase flows. The scope of the review is limited to one-fluid models, where a single set of governing equations are solved in the complete domain, by locally changing the material properties of the fluid. On the contrary a two-fluid model, which is not reviewed here, uses two sets of governing equations in corresponding fluids and apply boundary condition at the interface surface.

The salient feature, in numerical simulation of two-phase flows, is the presence of an interface separating the two-fluids. There are two ways in which the fluids and the interface can be represented. The first group consists of methods that store the interface shape and location between the fluids, and obtain the distribution of fluids in the domain from the interface data. Such methods are called interface tracking methods, which include height function and front tracking method. In the second group are the methods which store data at every cell in the domain and infer the interface location from the cell data. Such methods are called interface capturing methods. These include the MAC method, the VOF method, and the level-set method. The methods from both groups are described in the next two sections.

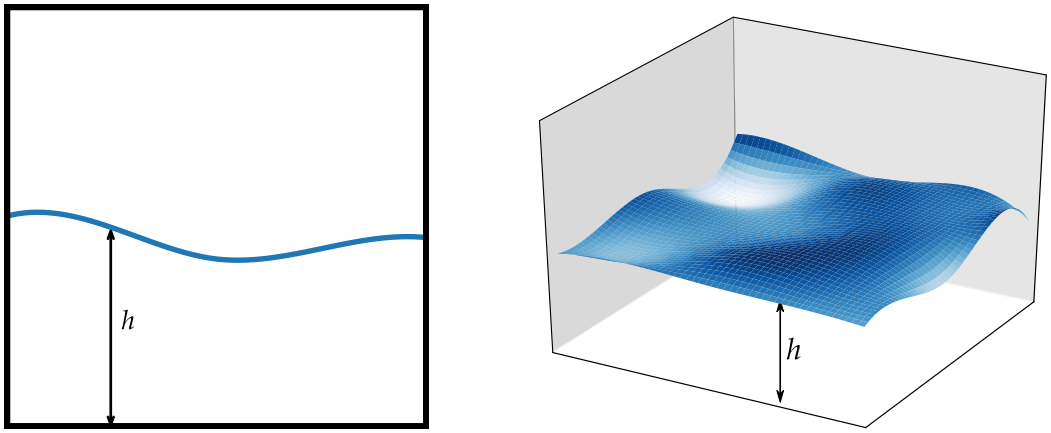
Another aspect of two-phase flows is the force due to surface tension at the fluid interface. A brief theory and literature related to numerical formulation of surface tension is described in section 2.3. Other important topics related to numerical simulation of incompressible two-phase flows includes: incompressible flow solvers, dual-time stepping, Riemann solvers and order of accuracy analysis. The theory and literature related to these topics is discussed in subsequent sections.

2.1 Interface tracking methods

In the interface tracking methods, markers are placed at the fluid interface and hence there is no need to place markers in the fluid. The curvature of the interface and therefore the surface tension forces can be very accurately computed. These methods however have a severe limitations when handling complicated interfaces, especially in case of three dimensional flows. The Interface tracking methods can be classified as height function method and front tracking method.

2.1.1 Height function method

In the height function method, the interface is defined by the distance along the y-axis from a reference plane. If we imagine a tank with floating weightless particles at the fluid interface, then the vertical distance of these marker particles from the base of the tank can be viewed as a height function, as illustrated in Fig. 2.1.



(a) two-dimensional (b) three-dimensional

Fig. 2.1: Height function for representing the interface.

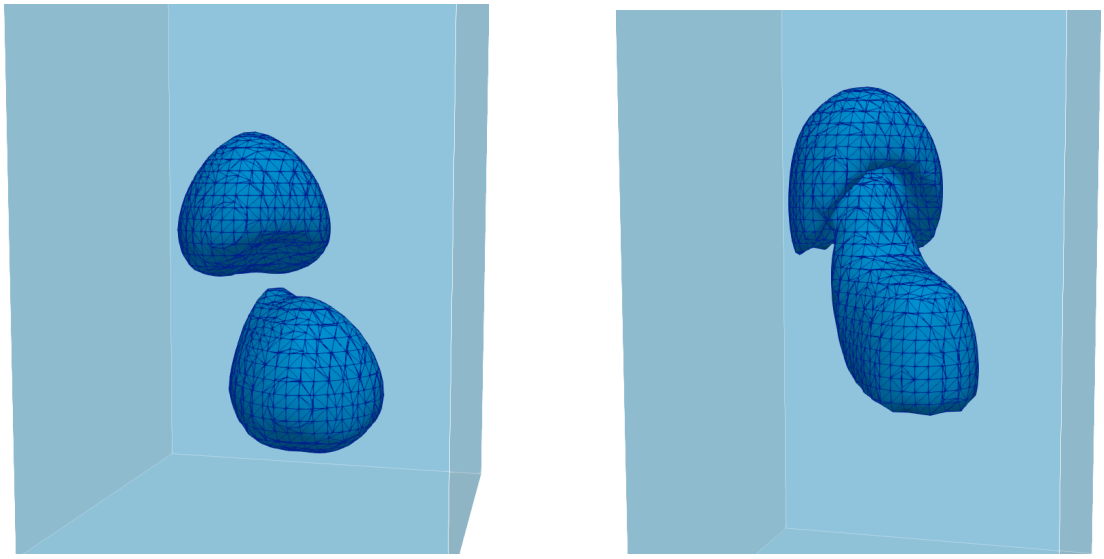
The height function, h, is evolved by using the kinematic equation,

$$\frac{\partial h}{\partial t} + u\frac{\partial h}{\partial x} = v\,, \tag{2.1}$$

for two-dimensional problems, where u and v are the Cartesian velocity components of the interface. A similar equation can be used for evolving the interface in three-dimensions. The paper by Hirt and Nichols [11], discusses the evolution procedure for the above equation. The paper also describes the limitations of this method, which include 1) It is restricted to smooth surfaces with slope, $\Delta h/\Delta x$, less then the cell aspect ratio, $\Delta y/\Delta x$, of the Eulerian grid used. 2) It cannot deal with interfaces with multi-valued height, and hence problems involving droplets or bubbles cannot be solved using this method. The height function method is also briefly discussed by Hyman [26].

2.1.2 Front tracking method

The front tracking method is a generalization of the height function method. In this method the marker particles are tracked by a combination of curves joining the marker particles forming the *interface front*. A interface front formed by joining marker particles using line segments is described by Hirt and Nichols [11]. In the paper by Hyman [26], a summary of existing methods for front tracking till that time were presented. Front tracking for gas dynamics in general has been presented by Chern, *et al.* [27], in which the application to Kelvin-Helmholtz instability has been studied. In 1992, the interface front tracking method was demonstrated by using a unstructured moving mesh for the front in an Eulerian stationary mesh by Unverdi and Tryggvason [15]. In this paper, the moving unstructured mesh was continuously restructured to maintain the front for a continuously deforming bubble. The complicated dynamics of a three-dimensional colliding drops was studied using front tracking method by Nobari and Tryggvason [28] in 1996. In 2001, similar work is extended by Tryggvason, *et al.* [22] to study interfaces in three-dimensional domain and axi-symmetric bubbles in two-dimensions. In three-dimensional problems the fronts need to be re-meshed using triangular mesh at every time step as shown in Fig. 2.2(a) with two bubbles.



(a) fronts tracking of two separated bubbles

(b) challenges faced due to merging surfaces

Fig. 2.2: Front tracking for three-dimensional problems with merging interface.

The major difficulty in front tracking methods arises in case of merging or breaking-off of interfaces. An example of such as challenge is shown in Fig 2.2(b), where due to merging bubbles the connectivity of the surface mesh needs to be recomputed. In such a case, the algorithm has to restructure the interface front mesh, which is computationally expensive and not trivial for complex fronts. This is probably the reason why this method has not gained a lot of attention in spite of the advantage of lesser memory requirement and accurate solutions when merging or separating interfaces are not involved.

2.2 Interface capturing methods

In the interface capturing methods, the information about the interface shape and location is inferred from cell data. The calculation of surface tension flux had been a challenge in these methods, until Brackbill [29] defined the mathematically equivalent surface tension as a volume force. The interface capturing methods are predominant, in the two-phase flow community. This is because the complex interactions between the fluid interfaces, such as breaking and coalescence of interfaces, are automatically taken care of by such methods. These methods date back to as early as 1965 and are still widely used. The methods in this group are explained below.

2.2.1 Marker and cell method

In the marker and cell (MAC) method massless marker particles are placed in the fluid, which advect along with the fluid at local fluid velocity. The flow is solved on a fixed Eulerian mesh with marker particles in the fluid as shown in Fig. 2.3. The presence of the interface is indicated by a non-empty cell in the neighborhood of empty cell. The traces of work done on multiphase flows can be seen since 1965 when Harlow, Shannon and Welch [10, 30] used the MAC method to simulate dam break problems at Los Alamos Scientific Laboratory. In their work only one fluid with free surface was simulated by placing massless marker particles in the fluid with necessary boundary conditions at the fluid interface. The staggered grid approach was also introduced in this paper which is extensively used even today for incompressible flows.

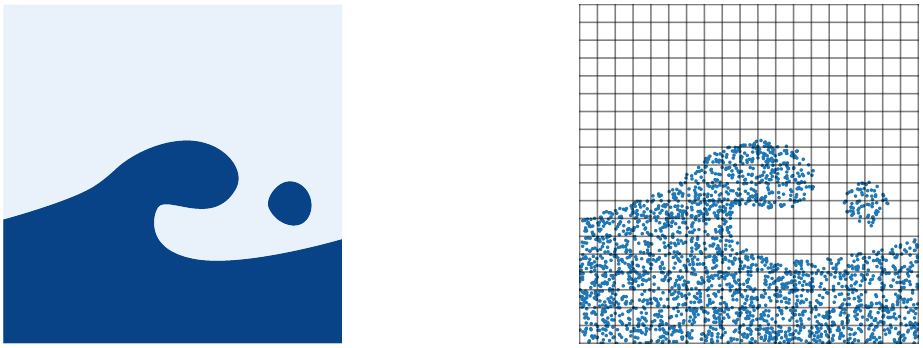


Fig. 2.3: An example of fluid interface with the corresponding marker particles and cells in the MAC method.

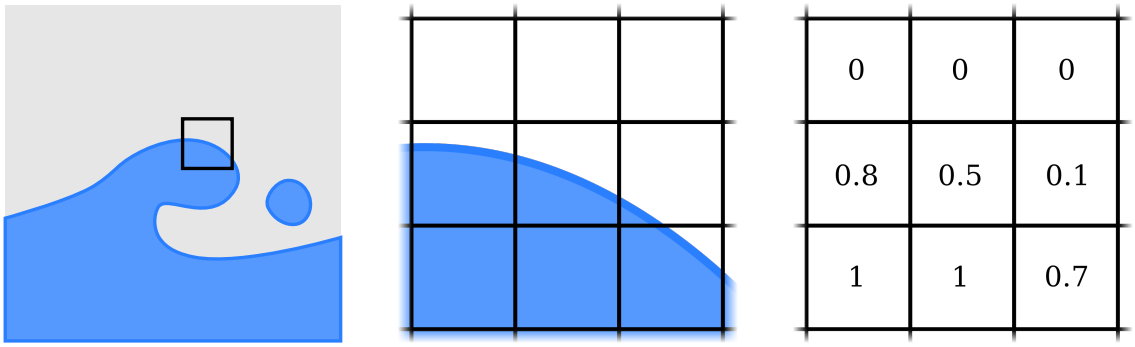
The original MAC method was extended for multiple fluids by Daly [12] in 1967 to simulate Rayleigh-Taylor instability of different fluids with density ratios of 1.1:1, 1.5:1, 2:1 and 10:1. The method was also used by Harlow and Shannon [31] to simulate splash of liquid drop. In 1970, Amsden and Harlow [32] presented a simplified version of the MAC method called simplified-MAC (SMAC). In the SMAC method marker particles are used as in case of MAC method, but the solution of fluid flow equations is simplified by defining a pseudo-pressure for the cells containing the surface. The

description of the SMAC program and implementation details are given in the Los Alamos Report LA4370 [33].

It was soon realized that the MAC method requires a lot of computer memory to store the marker particle positions, which can be reduced by using an Eulerian framework with a single marker function value stored for each cell. Also, the non-uniform distribution of marker particles in MAC method could lead to inaccuracies as the flow evolved. This led to the development of the volume of fluid method.

2.2.2 Volume of fluid method

In the volume of fluid (VOF) method, each cell stores the cell-averaged value of volume fraction of the primary fluid. This value varies between 0 and 1 indicating the absence or presence of the primary fluid respectively. The interface between the fluids is not explicitly saved, but can be inferred from the volume fraction data saved for each cell. The interface is located in the cells with a value of the volume fraction between 0 and 1. This is schematically shown in Fig. 2.4 for a subset of 3×3 cells. The volume fraction flux can be computed using an upwind method, however it would lead to smearing of the interfaces, due to excessive diffusion. The smearing of interfaces is not acceptable in practical applications, hence different ways have been developed to maintain a sharp interface.



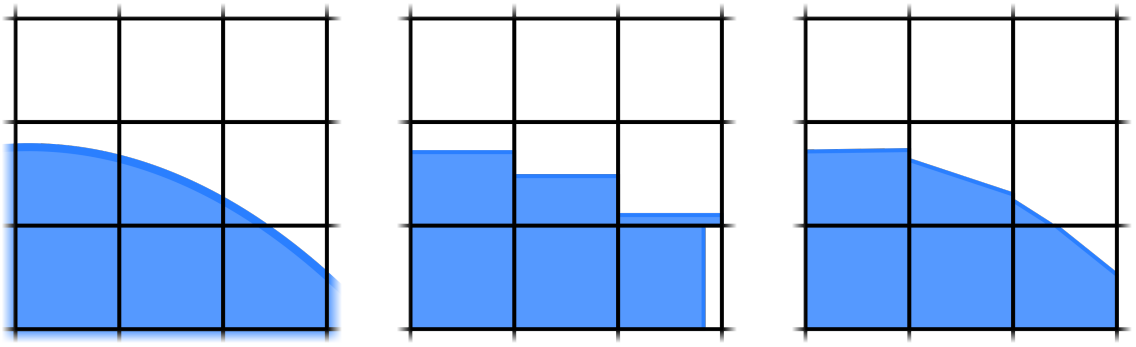
(a) larger view (b) local grid view (c) volume fractions

Fig. 2.4: Representation of interface as cell-averaged volume fraction in VOF method.

2.2.2.1 Maintaining sharp interfaces in VOF

The first approach, to overcome the smearing of the interfaces, is to reconstruct the interface geometrically, such that it accurately represents the real interface within the cell. Such methods are called interface reconstruction methods. The second approach, to overcome the smearing of the interfaces, is to use the upwind methods for volume fraction flux, followed by a procedure to maintain the interface sharpness. Such methods are referred to as interface compression methods. These two approaches are described below.

Interface reconstruction: A marker function based on volume fraction of the fluid has been used by Noh and Woodward [13] in 1976. The fluid interface was reconstructed by using a method called simple line interface calculation (SLIC). In SLIC the interface is constrained to be parallel to one of the coordinate direction (i.e horizontal or vertical), as shown in Fig. 2.5(b). This method was detailed in a paper by Hirt and Nichols [11], and the method was referred to as volume of fluid (VOF) method. In their paper, the VOF method was also demonstrated by solving various examples such as dam break, undular bore, breaking bore and Rayleigh-Taylor instability. The VOF method does not explicitly use marker particles on the interface, and the interface is constructed based on the values of volume fraction of neighboring cells. This causes a difficulty in calculation of surface tension flux which was defined only as a surface force. This problem has been addressed by Brackbill [29] in 1992 by defining a continuum method for calculating surface tension which can be used in VOF method. Many researchers have used VOF to solve various problems of varying complexities such as in the paper by Tomiyama [34] where the gas bubble rising in a liquid is studied and compared to experimental results. The SLIC reconstruction of the interface is computationally very cheap, but it is unable to tackle problems with high shear flows. In case of high shear in fluid flow, the SLIC method creates non-physical isolated fluid regions commonly known as jetsam and floatsam [13, 35].



(a) actual interface (b) SLIC reconstruction (c) PLIC reconstruction

Fig. 2.5: Interface reconstruction using SLIC and PLIC technique in the VOF method.

The need for high order accurate reconstruction of interface was felt with increase in computational power availability. In 1982 Youngs [14] developed the higher order reconstruction method called piecewise linear interface reconstruction (PLIC). In 1998, Rider and Kothe [36] provided a detailed algorithm for geometric flux calculation with methods to overcome problems such as overlapping areas during volume advection. In the PLIC method the interface is constructed as linear segments inside each cell like in case of SLIC, but without the constraint of orientation. Thus allowing from linear approximation to the interface, as shown in Fig. 2.5(c). This improvement makes the method very accurate but also computationally expensive. The PLIC method was extended by Scardovelli and Zaleski [37] in 2003, Pilliod and Puckett [38] in 2004 and Aulisa, *et al.* [39] in 2007, using least square fit for interface reconstruction, to obtain a

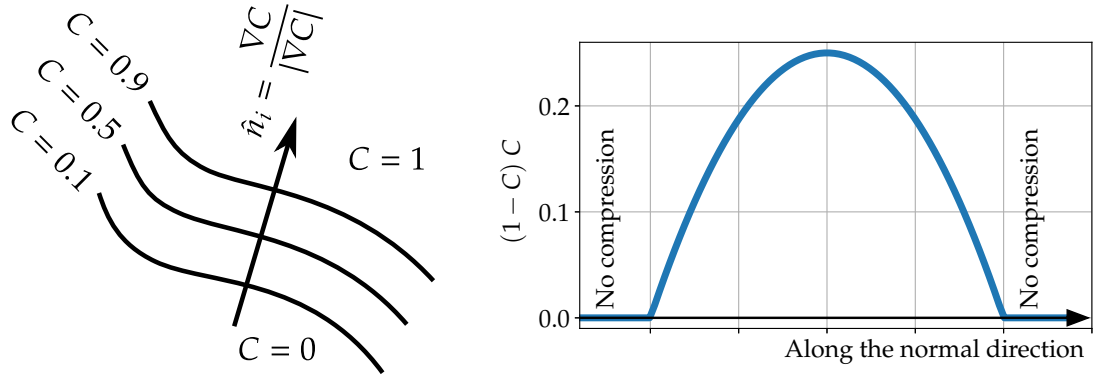
better converging and accurate method. The advantage of SLIC and PLIC type reconstruction methods is that they naturally maintain very sharp interfaces. The limitation however is that the extension of these methods to arbitrary unstructured meshes gets complicated. Nevertheless, such an extension is attempted by Ji and Shi [19] in 2008 and Huang, *et al.* [24] in 2012. Another limitation is that these methods may result in unphysical solutions called "jetsam" and "floatsam" [13, 35]. These are unphysical small pieces of fluid floating around which break-off in regions of high shear. Another method in this family is called the tangent of hyperbola interface capturing (THINC), which uses a non-linear function to represent the interface inside the cell. This method was introduced by Xiao, *et al*. [40] in 2005 and then improved upon by multiple researchers [41, 42].

Interface compression: As seen from the literature review above on VOF method, it is customary to reconstruct the interface inside the cell (using methods like SLIC and PLIC), for calculation of advection fluxes. The reason for the above approach is that upwind methods, which work well for compressible flows, are not acceptable for capturing fluid interfaces, due to high diffusion. There has been some research effort to overcome this limitation of upwind methods.

In 1997 Rudman [35] used the flux corrected transport method for high resolution capturing of the fluid interface. In this method a precise combination of upwind and downwind flux is used so that the spurious oscillations do not occur, and at the same time the diffusion is controlled, thus capturing the jump in volume fraction accurately at the interface. A very well recognized work of reducing the diffusion of interface was done by Ubbink [43] in 1999. Ubbink introduced a method called compressive interface capturing scheme for arbitrary meshes (CICSAM). This method makes use of the normalized variable diagram to tune the numerical differencing scheme to maintain the sharpness of the interface and also maintaining numerical stability.

If the mass, momentum and volume fraction equations are written as a hyperbolic system of partial differential equations (PDEs), then the methods from well understood hyperbolic solvers can be utilized for solving the multiphase flow problems. The artificial compressibility formulation by Chorin [44] can be used so that the PDEs are hyperbolic in nature. A Roe-type Riemann solver for artificial compressibility formulation has been developed by Pan and Chang [45] in 2000. In their work the interface sharpness is accomplished by using a modified slope limiter. A similar approach is also taken by Zhao, *et al.* [46] for solution of multiphase problems. This method has been demonstrated to be successful by solving various problems such as Rayleigh-Taylor instability, dam break problem, rising bubble problem of various densities, Weber numbers and Reynolds numbers and micro-drop injection problem. The interface capturing using high resolution method has also been studied by Yakovenko and Chang [47] in 2008 and Chen Price and Temarel [48] in 2010. The advantage of this methodology of flux calculation is that it is very straightforward to implement for arbitrary unstruc-

tured meshes.



(a) interface contours and normal

(b) scaling coefficient of compression velocity

Fig. 2.6: Behavior of the interface compression function.

Instead of using a modified slope limiter [45], the interface compression method can be used to maintain interface sharpness. The method of interface compression was introduced by Rusche [49] and was implemented in the OpenFOAM [50] software. In this method the compression is achieved by introducing a compressive flux, given as,

$$\mathbf{F}_c = \vec{V}_c \ (1 - C) \ C \ , \tag{2.2}$$

where, $\vec{V}_c$ is called the compression velocity. The compression is restricted to regions close to the interface due to the term $(1 - C)\ C$, as illustrated in Fig. 2.6. This method was further improved by using a dynamic interface compression by Lee [51], which modifies the compression coefficient locally based on interface orientation.

2.2.2.2 Calculation of flux in VOF

In the VOF method, the calculation of convective flux for evolving the interface in time has been achieved using three approaches:

1. geometric method
2. flux corrected transport method
3. upwind methods such as Riemann solvers

The flux corrected transport and the upwind methods, together, are sometimes referred to as algebraic methods in the literature [41, 42]. The three approaches for calculation of volume fraction flux are discussed below.

1. Geometric method: The oldest approach of flux calculation in VOF is to compute the flux geometrically. In this method, the the interface is reconstructed by using SLIC [13] or PLIC [14] and the computed cell volume is geometrically advected to

neighboring cells. The flux can be computed either by a split method [52] , or by an unsplit method [36]. In the split method, the flux is computed in one coordinate direction at a time. In the unsplit method, the volume fraction is updated by advecting the fluid volume depending on the local velocity vector field, as shown in Fig. 2.7. Due to the numerical nature of velocity distribution, even the velocity field in incompressible fluid flow can be slightly divergent. Therefore, the geometrically advected fluid can lead to overlapping volumes. This issue is addressed by Rider and Kothe [36] and they provide a detailed algorithm to deal with such problems. It is possible to extend this method to unstructured and mixed cell-types, which is attempted by a few researchers [19, 24], but the implementation is very cumbersome due to their geometric nature.

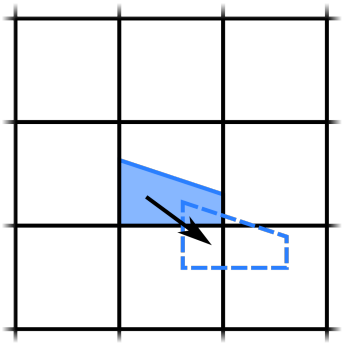


Fig. 2.7: Schematic of geometric volume advection in VOF.

2. Flux corrected transport method: The second approach for VOF flux calculation is to use a combination of upwind and downwind flux, such that the interface remains sharp. The contributions from the upwind and downwind fluxes has to be tuned properly to ensure stability of the solver. One such attempt is to use the flux corrected transport (FCT) method [53] for flux calculations. This method was incorporated in VOF by Rudman [35], and the method was called FCT-VOF. This method is simple to extend to multiple dimensions with unstructured and mixed cell-types. Another variation to this approach was proposed by Ubbink [43], in the method called compressive interface capturing scheme for arbitrary meshes (CICSAM). In CICSAM method the normalized variable diagram is used to tune the contributions from upwind and downwind fluxes. The CICSAM is available in commercial software such as Ansys Fluent.

3. Upwind methods: The third approach for VOF flux calculation is to simply use an upwind flux with a high-order solution reconstruction with a limiter to ensure oscillation free solution. In this approach, some type of additional treatment is necessary to ensure sharp interfaces. The important feature of this approach is that convective flux for the complete system of governing equations, including the VOF equation, can be computed in a closely-coupled manner using Riemann solvers. The Riemann solvers

can be designed to accurately compute the flux by considering the physics of the flow. A Roe-type Riemann solver has been developed, by Pan and Chang [45] in 2000, for calculation of VOF flux.

2.2.3 Level set method

In the level-set method (LSM), the marker function used is called a level-set function which is generally a signed-distance from the interface. Therefore, a level-set of zero corresponds to the interface, a value greater than zero corresponds to one fluid and a value smaller than zero corresponds to the other fluid. The level-set function can represent complex features such as merging or splitting of fluid interfaces very easily as shown in Fig. 2.8[†]. The level set function is advected at fluid velocity to track the interface as the solution evolves. For the calculation of fluid properties, a smoothed Heaviside function is constructed from the level-set function which changes from 1 to 0 at the interface between the two fluids. This operation can be viewed as transforming the level-set function into an approximate volume-fraction function.

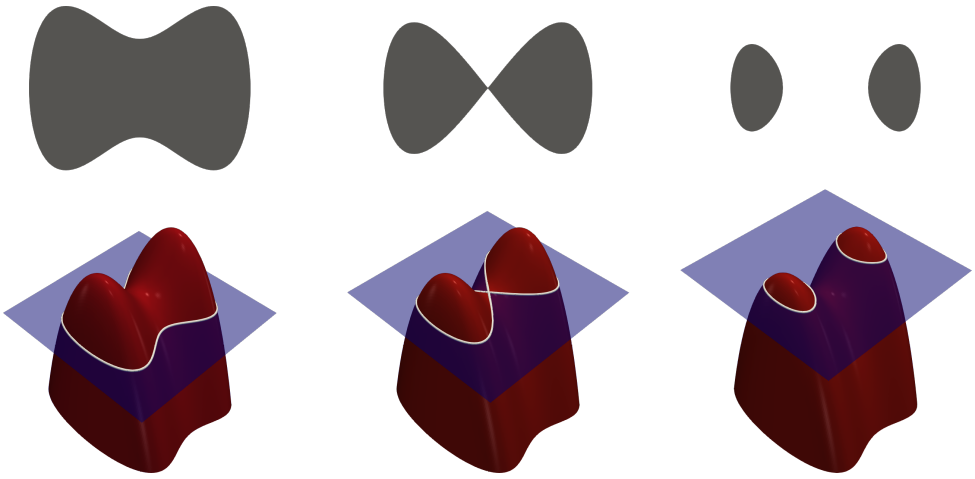


Fig. 2.8: Level set function with merging interfaces for two-dimensional problems.

The LSM was adopted from other fields of research to fluid flow in 1988 by Osher and Sethian [54] for propagating interfaces with curvatures. This was done to overcome the difficulties of VOF method due to sudden change in the volume fraction at the interfaces. The LSM was applied to compressible gas dynamics with multi-fluid flows consisting of air-air and air-helium gases in 1992 by Mulder, Osher and Sethian [55]. In 1994, Sussman, Smereka and Osher [16] applied the LSM to high density ratio (1000:1) flows consisting of water-air combinations and demonstrated with problems of air bubble rising in water and water drop in air. In this paper the advantage of choosing signed distance as level set function is also discussed.

The "fast" LSM was developed by Adalsteinsson and Sethian [56] in 1995 to reduce the computation time of LSM, by defining the signed-distance function only in the

[†]Level-set function graphics: Nicoguaro / CC BY (https://creativecommons.org/licenses/by/4.0)

vicinity of the interface. Numerical experiments were conducted by Chang, Hou, Merriman and Osher [57] in 1996, to deduce the feasibility of LSM for incompressible flows and analyzed the mass conservation properties of LSM. This paper also displayed an important limitation of LSM, which is its inability to conserve mass. Sussman, *et al.* [58] in 1998, used essentially non-oscillatory (ENO) method for space discretization and higher order Runge-Kutta time discretization with LSM for computing effects of stiff surface tension and steep density gradients in the flow.

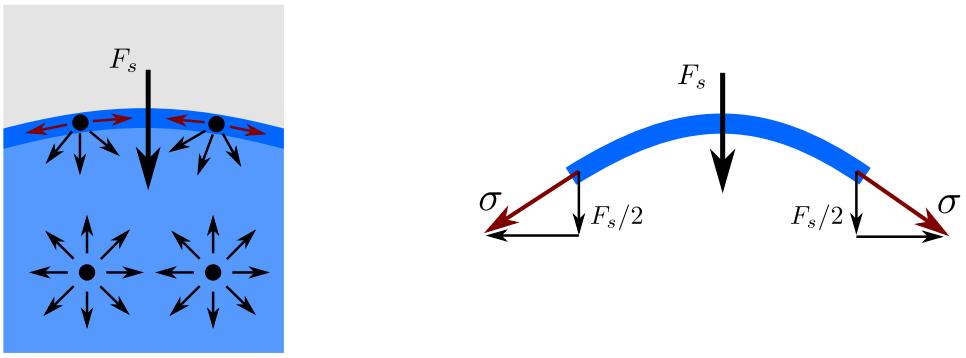
Coupling of LSM with VOF to overcome limitation of LSM (non-conservation of mass) was studied by Sussman and Puckett [59]. It has been shown in this paper that the coupling of LSM and VOF together produces superior results compared to either of the methods used alone. In 2001, Osher and Fedkiw [60] gave an overview of the state-of-art LSM at that time, covering incompressible and compressible flows. Again in 2001, a similar paper was written by Sethian [61] for presenting the evolution of LSM till then. In 2010, the coupling of VOF and LSM was again studied by Sun and Tao [62]. In their paper, an iterative geometric operation is used to calculate the level set function which is claimed to be efficient compared to earlier coupled method [59]. The coupled method introduces many complications, as two different methods are involved in the solver.

The conservative level-set method was introduced by Olsson, *et al.* [17, 63] as an attempt to reduce the mass error in the level-set method. In the conservative level-set method, the level-set function used is similar to the volume fraction smoothly changing at the interface. The level-set function used is a hyperbolic tangent function with the interface identified by the level of 0.5, instead of 0 which was used in the signed-distance function. Similar to the signed-distance function, the hyperbolic tangent function also needs to reinitialized frequently, so as to maintain its properties as the solution evolves in time.

2.3 Surface tension

One of the phenomena which is specific to multiphase fluid flows is surface tension. In the bulk of the fluid, the molecular cohesive forces are uniform and balanced in all the directions. However, close to the interface due to the presence of two separate fluids, these forces are not balanced. This imbalanced resultant force, F_s, acts in the direction normal to the interface, as shown in Fig. 2.9(a). At a macroscopic level such a force tries to minimize the surface area of the interface, as if the surface is being stretched (or under tension force), as shown in Fig. 2.9(b). The tangential force acting along the interface is called the surface tension force. The magnitude of the surface tension force depends on the molecular properties of the two fluids. For a given combination of fluids, the molecular properties of the interface are expressed by a single parameter called the coefficient of surface tension, commonly denoted by the symbol σ. The coefficient of surface tension has units of force-per-unit-length $[\![\mathrm{N/m}]\!]$. The value of σ can

be obtained experimentally or can be looked up from a catalog such as the Physical Chemistry of Surfaces, by Adamson and Gast [64], for required combination of fluids.



(a) (b)

Fig. 2.9: Schematic diagram for depicting the surface tension force at microscopic and macroscopic levels.

The surface tension force is defined only at the fluid interface separating the two fluids. Therefore, incorporating the force in interface capturing methods had been difficult, until Brackbill, *et al*. [29] defined the mathematically equivalent volume force model. This model is called the continuum surface force (CSF). In the CSF model, the surface tension term is introduced as a source term. On the contrary, in the continuum surface stress (CSS) model, by Lafaurie, *et al*. [65], the surface tension is introduced as a stress term. The CSS model is a conservative formulation allowing for spatial variation of surface tension coefficient. Different methods for numerical computation of surface tension are studied in [66]. The numerical treatment of surface tension using the CSS model is described in section 3.6 of this thesis.

2.4 Incompressible flow solvers

In case of incompressible flows, the divergence-free condition needs to be satisfied by the velocity field. This can be written as,

$$\nabla \cdot \vec{V} = 0 \,, \tag{2.3}$$

where $\vec{V}$ is the velocity vector. This is equivalent to the equation of mass conservation. Also, the conservation of momentum needs to be satisfied, which is given by the Newton's second law. For an incompressible flow, with uniform density, the conservation of momentum can be written as,

$$\frac{\partial \vec{V}}{\partial t} + \nabla \cdot \left(\vec{V} \otimes \vec{V} \right) = -\nabla p + \nu \nabla^2 \vec{V} + g \,, \tag{2.4}$$

where, $\otimes$ is the operator for outer product: $\vec{V} \otimes \vec{V} = \vec{V}\vec{V}^T$, p is the kinematic pressure, ν is the kinematic viscosity and g is the acceleration due to gravity. The above equa-

tions form a closed system, as the number of unknowns are equal to the number of equations. However, it is difficult to numerically solve for the unknowns using these equations in the current form. The numerical difficulties arise because the velocity can be evolved using the momentum equations, but there is no equation for evolving the pressure. This means that any arbitrary pressure distribution can be used to obtain a velocity field, by utilizing the momentum equations, but the obtained velocity field may not satisfy the divergence-free condition. In other words, the velocity and pressure are indirectly coupled through the divergence free condition. The governing equations therefore need to be modified to obtain a more direct pressure-velocity coupling. This is normally achieved by two different class of methods[‡] described below.

2.4.1 Pressure based method

In pressure based methods, an additional elliptic equation is derived to solve for pressure. The equation for pressure is obtained by combining the divergence-free condition with the momentum equations. The equations for velocity and the derived equation for pressure are iteratively solved until convergence is achieved to required degree of accuracy. Many important methods such as the SIMPLE [67], SIMPLEC [68] and the PISO [69] algorithms fall in this category of methods.

2.4.2 Artificial compressibility method

The artificial compressibility method was introduced, by Chorin [44] in 1967, for solving incompressible single-phase flows, and has since been used widely [70–74]. In this method, an artificial term is added to the mass equation (2.3) for evolving the pressure. The modified equation can be written as,

$$\frac{1}{\beta}\frac{\partial p}{\partial t} + \nabla \cdot \vec{V} = 0 \,, \tag{2.5}$$

where, β is called the artificial compressibility parameter. The artificial term can be viewed as a simple pre-conditioner for the system of partial differential equations (PDEs). A more elaborate pre-conditioner was defined in 1987 by Turkel [75]. This equation along with the momentum equations can be used to evolve pressure and velocity field. However, the solutions are valid only at the steady-state, as the artificial term vanishes. Nevertheless, the method can be used for solving unsteady flows by using the dual-time stepping algorithm [76].

Due to the modification of governing equations, the convective part of the equations behave like a hyperbolic system. Thus allowing the use of Riemann solvers, for

[‡]Other methods, such as the vorticity-stream function formulation, are commonly used for solving incompressible flows. However, such methods are not included here due to their limited use in two-phase flows.

calculation of the convective flux. An HLLC-type contact preserving Riemann solver for incompressible flows was developed by Mandal and Iyer [72].

The value of β plays a very important role in the rate of convergence of the solver. A few investigations, such as in [73], have been made to achieve an optimal value of β. The artificial compressibility method has also been widely used for two-phase flows by various researchers [46, 47, 77–82].

In two-phase flows using VOF, the artificial compressibility method provides a close-coupling between the pressure, velocity and the volume fraction. This close-coupling results in two major advantages when dealing with two-phase incompressible flows compared to other methods. Firstly, the artificial compressibility method is found to converge faster [83] compared to pressure-based methods. Secondly, there is no lag between the velocity field and the volume fraction, because the complete flux vector is computed at the same instant as the solution converges.

2.5 Dual-time stepping for time-accurate solutions

As discussed above, the artificial compressibility method is only useful to obtain steady state solutions. To apply the method to unsteady flows, a dual-time stepping approach needs to be used. The dual time-stepping method was first introduced by Jameson, *et.al* [84] for solving compressible flow over airfoils. Its adoption to artificial compressibility method was studied by Gaitonde [76]. Since then the method has been used for solving unsteady incompressible flows [73, 74, 82, 85, 86]. This method uses two time-like variables: the pseudo-time – τ, and the real-time – t, hence the name "dual-time".

In this method, pseudo-time derivatives are introduced in all the governing equations (i.e. mass and momentum equations). The real-time derivatives are computed by using high-order finite-difference methods. At every real-time step, the pseudo-time derivatives are converged to the required accuracy, thus obtaining a time accurate solution. The details of the dual-time stepping method are presented in section 3.3.

2.6 Riemann solvers

A Riemann problem is a initial value problem, defined for a system of hyperbolic partial differential equations,

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = 0 \,, \tag{2.6}$$

with the initial conditions,

$$\mathbf{U}\left(x,0\right) = \begin{cases} \mathbf{U}_L & x < 0 \,, \\ \mathbf{U}_R & x > 0 \,, \end{cases} \tag{2.7}$$

where, $\mathbf{U}$ is the solution vector with m components, $\mathbf{F} = \mathbf{F}(\mathbf{U})$ is called the flux vector, $\mathbf{U}_L$ is called the left state and $\mathbf{U}_R$ is called the right state. This is pictorially shown in Fig. 2.10.

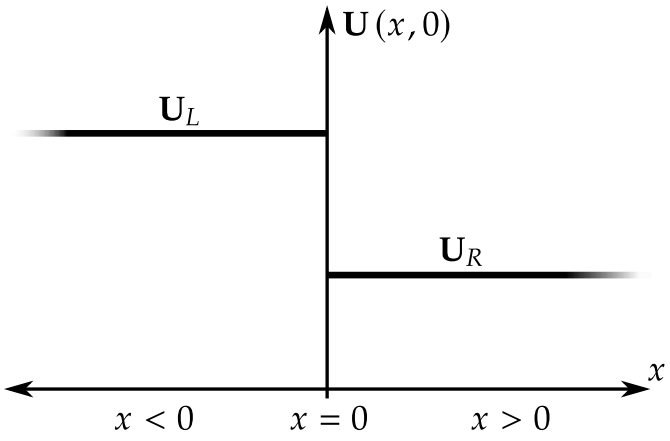


Fig. 2.10: Pictorial representation of a Riemann problem

Using the Jacobian and the chain-rule, the hyperbolic system of PDEs (2.6) can be re-written as,

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial x} = 0\,, \tag{2.8}$$

where, the Jacobian matrix is calculated as,

$$\frac{\partial \mathbf{F}}{\partial \mathbf{U}} = \begin{bmatrix} \partial F_1/\partial U_1 & \partial F_1/\partial U_2 & \cdots & \partial F_1/\partial U_m \\ \partial F_2/\partial U_1 & \partial F_2/\partial U_2 & \cdots & \partial F_2/\partial U_m \\ \vdots & \vdots & & \vdots \\ \partial F_m/\partial U_1 & \partial F_m/\partial U_2 & \cdots & \partial F_m/\partial U_m \end{bmatrix}. \tag{2.9}$$

A system of partial differential equations is said to be hyperbolic if all the eigenvalues of the Jacobian matrix, are real, with distinct eigenvectors. The exact solution to the Riemann problem, $\mathbf{U}(0,t)$, can be obtained using a Godunov solver [87]. However, for non-linear problems, where $\partial\mathbf{F}/\partial\mathbf{U}$ is a function of $\mathbf{U}$, the Godunov solver requires an iterative approach. In numerical methods for solving PDEs, such as the finite volume method, this can be computationally very expensive, as the flux $\mathbf{F}(\mathbf{U}(0,t))$ has to be computed for every cell face in the mesh. Therefore, approximate Riemann solvers are generally employed to obtain numerical flux [88].

An approximate Riemann solver computes solution to a linearized Riemann problem, where the eigenvalues and the eigenvectors are assumed to be constant over the evolution time. The evolution pattern of the Riemann problem, in the x-t plane, is shown in Fig. 2.11(a). As time evolves the initial single jump in solution splits into m distinct jumps with $m-1$ intermediate states, with each jump traveling at the speed corresponding to its eigenvalue. An illustrative representation is shown in Fig. 2.11(b), for one of the components of $\mathbf{U}$. The eigenvalues are denoted as $\lambda_1, \lambda_2, \ldots, \lambda_m$, with the corresponding eigenvectors given by $\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \ldots, \mathbf{R}^{(m)}$. Approximate Riemann

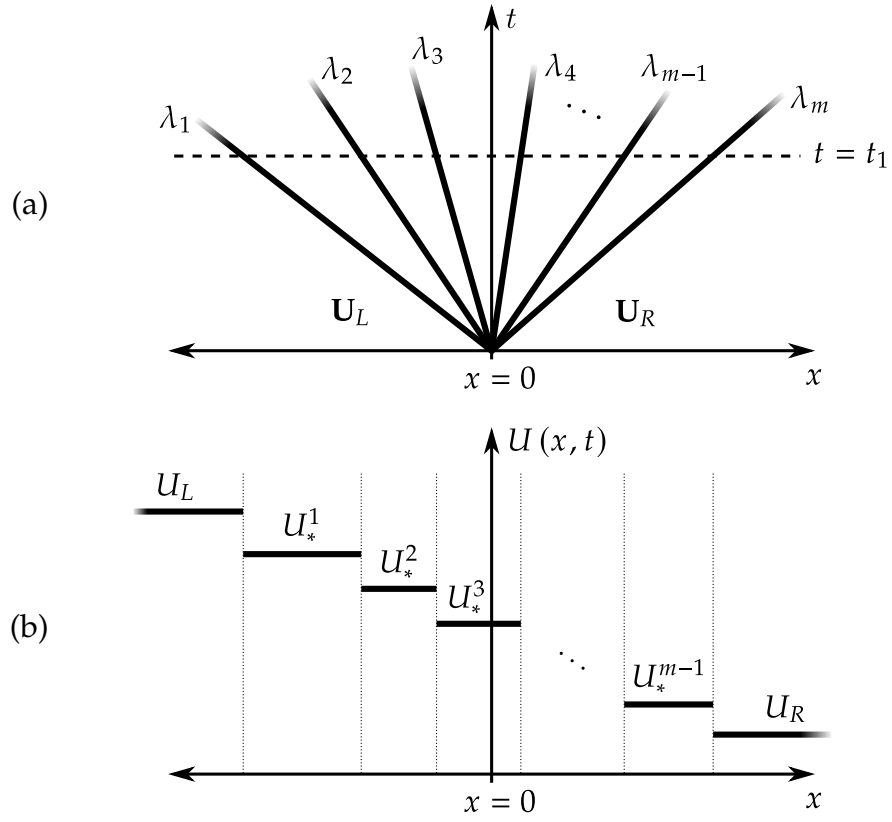


Fig. 2.11: Evolution pattern of a Riemann problem: (a) eigenvalues corresponding to waves, (b) illustration of corresponding Riemann states for a component of U.

solvers make use of the wave speeds (and the corresponding jumps in the solution) to efficiently obtain solution to the linearized Riemann problem.

Two of the most popular generic Riemann solvers are the Rusanov solver [89] and the Harten-Lax-vanLeer (HLL) solver [90]. In the Rusanov solver only the maximum absolute value of the eigenvalue is used. In case of HLL solver the minimum and maximum eigenvalues are used with a constant intermediate state. Other more accurate and PDE specific Riemann solvers make use of more waves, such as the Roe solver [91] and the contact preserving Harten-Lax-vanLeer (HLLC) solver [92]. In the Roe solver, a single intermediate average state is computed called the Roe-averaged state. The flux is computed using all the eigenvalues and eigenvectors of the Jacobian which are computed using the Roe-averaged state. The HLLC solver uses the Rankine-Hugoniot jump conditions across all the waves and generalized Riemann invariants to compute the flux.

Many of the Riemann solvers use one-dimensional concept, therefore, the rotational invariance property of the governing equations is used for calculation of the flux when solving two- and three-dimensional problems. The procedure for applying the Riemann solvers to multi-dimensions is described in section 3.4.2. The book on Riemann solvers by Toro [88] is a very useful reference on the subject.

2.7 Order of accuracy analysis

The Godunov theorem [87] states that it is not possible for a *linear* higher order scheme, (of order two or higher), to ensure a non-oscillatory solution for hyperbolic equations. A linear scheme is the one where a linear combination of cell averages is used for local reconstruction of the solution. It is however observed that, if there are no discontinuities in the solution then, the numerical solution obtained by a linear high order scheme is much superior compared to the first order upwind scheme. It has been a quest of many researches to circumvent the limitation imposed by the Godunov theorem to achieve an order of accuracy as high as possible. This effort has led to two classes of methods of adding non-linearity to the scheme. The two classes of methods are: the artificial viscosity methods and the total variation diminishing methods. In case of artificial viscosity methods [93] additional non-physical, viscous like terms, are added to the scheme such that the oscillations are damped out. The currently available methods in this class, have to define a coefficient which has to be tuned according to the problem. Therefore, these methods are not easily extensible to general problems. The total variation diminishing (TVD) methods use slope-limiters and flux-limiters to locally switch to a lower order scheme, based on the local solution gradients. This class of methods are more easily extensible to large variety of problems and therefore are more widely used. Many major contributions have been made to this class of methods over the years [53, 90, 94–100]. Various limiters have been designed to obtain high-order reconstructions in smooth regions but reduce the order of accuracy close to discontinuities. The limiter versus the gradient-ratio plot commonly known as Sweby plot [99] is used in these methods to define a limiter function. Some of the widely used limiter functions are Superbee [101], Vanleer [94], Vanalbada [102] and Minmod [103].

A hybrid class of schemes closely related to TVD methods are solution dependent methods. In these methods a MUSCL-type [104] solution strategy is used, with a linear combination of all possible solution reconstructions. MUSCL stands for Monotone Upstream-centered Scheme for Conservation Laws. These type of methods include a family of more recent methods like SDWLS [105, 106] and WENO [107–112] methods. The combination of weights is so chosen, such that it reduces the oscillations in the solutions, called Gibbs phenomenon. These methods do not enforce the TVD condition explicitly, and therefore can encounter finite oscillations in the solution.

The current trends in high order methods and their need are very well reviewed by Wang and group in their article [113]. The order of accuracy analysis of WENO methods for structured grids is performed by Shu and Balsara [107, 109]. The analysis on unstructured grid for WENO methods is carried out by Hu [108] and Liu [111]. Using the ADER approach (Arbitrary high-order DERivative Riemann problem) and discontinuous Galerkin finite element method, the analysis has been performed by Dumbser, *et al*. [114, 115].

2.8 Beyond existing literature

In this chapter, the existing literature related to numerical methods and incompressible two-phase flows is briefly discussed. Necessary references are provided, pointing to the original work, for further perusal of the literature. It is understood from the literature review that the currently available methods, for numerically solving two-phase flows, suffer from a few shortcomings. Among the currently available methods, the VOF method has an attractive feature that it accurately conserves mass by design. There are two variants of the VOF method depending on the reconstruction of the volume fraction and flux calculation — the ones using geometric method and others using algebraic method for capturing the interface. The geometric reconstruction methods, such as the PLIC method, accurately capture the fluid interface. However, it can be challenging to apply the geometric methods to real life problems using unstructured mesh due to complicated flux calculation involved in these methods. On the other hand, the algebraic methods make use of upwind solvers for flux calculation and therefore naturally extend to unstructured mesh. However, the algebraic methods fail to maintain a sharp fluid interface resulting in inaccurate two-phase flow simulations.

These problems with the current solvers are overcome in this thesis through two novel contributions. The first contribution is the development of a Riemann solver, which takes into account the physics of fluid interfaces by modeling the interface explicitly as a contact wave. The Riemann solver is named as “HLLC-VOF” solver. This solver easily extends to unstructured mesh and also substantially reduces the smearing of the interface. The second contribution is the development of an interface reconstruction technique which is easy to extend to arbitrarily shaped cells. This reconstruction technique combines easily with a Riemann solver to accurately capture sharp fluid interfaces in two-phase flow simulations.

Chapter 3

Mathematical and Numerical Background

It is difficult to numerically solve incompressible fluid flows, as there is no equation for evolving the pressure. The artificial compressibility formulation [44] is one of the ways of solving steady-state incompressible flows. In such a formulation, a pseudo-time derivative is introduced, to obtain an equation for pressure. However, the formulation is valid only for steady-state flows. To solve for unsteady flows, a dual-time stepping algorithm needs to be used. This ensures that at every real-time, the pseudo-time derivative term is driven to zero. In order to solve for two-phase incompressible flows, an equation for advection of volume fraction is added. The mathematical formulation and the numerical techniques for solving such flows are presented in this chapter. This chapter provides the numerical and mathematical background necessary for understanding the novel contributions presented in the later chapters.

3.1 Governing equations

The system of partial differential equations, governing the three-dimensional, unsteady, incompressible, two-phase flows, closely-coupled with the VOF function can be written using the artificial compressibility formulation as,

$$\frac{\partial \mathbf{U}}{\partial \tau} + \frac{\partial \mathbf{W}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} + \frac{\partial \mathbf{H}}{\partial z} + \nabla \cdot \mathbf{F}_c = \frac{\partial \mathbf{F_v}}{\partial x} + \frac{\partial \mathbf{G_v}}{\partial y} + \frac{\partial \mathbf{H_v}}{\partial z} + \nabla \cdot \mathbf{T} + \mathbf{S}; \qquad (3.1)$$

$$\mathbf{U} = \begin{bmatrix} p/\beta \\ \rho u \\ \rho v \\ \rho w \\ C \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} 0 \\ \rho u \\ \rho v \\ \rho w \\ C \end{bmatrix} \quad \mathbf{F} = \begin{bmatrix} u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ uC \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ vC \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ wC \end{bmatrix}$$

$$\mathbf{F_v} = \begin{bmatrix} 0 \\ 2\mu\frac{\partial u}{\partial x} \\ \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \\ \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \\ 0 \end{bmatrix} \quad \mathbf{G_v} = \begin{bmatrix} 0 \\ \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \\ 2\mu\frac{\partial v}{\partial y} \\ \mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \\ 0 \end{bmatrix} \quad \mathbf{H_v} = \begin{bmatrix} 0 \\ \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \\ \mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \\ 2\mu\frac{\partial w}{\partial z} \\ 0 \end{bmatrix} \quad \mathbf{S} = \begin{bmatrix} 0 \\ \rho g_x \\ \rho g_y \\ \rho g_z \\ 0 \end{bmatrix}$$

where, τ is a pseudo-time variable used for iterating to a converged solution at a particular real-time t. $\mathbf{U}$ is the conservative variable vector updated in pseudo-time, $\mathbf{W}$ is the conservative variable vector updated in real-time, $(\mathbf{F}, \mathbf{G}, \mathbf{H})$ are convective flux vectors, $\nabla \cdot \mathbf{F}_c$ is the interface compression term, $(\mathbf{F_v}, \mathbf{G_v}, \mathbf{H_v})$ are the viscous flux vectors, $\mathbf{S}$ is a vector containing source terms and $\mathbf{T}$ is a tensor for surface tension force. The variable p denotes pressure, β is the artificial compressibility parameter, (u, v, w) is the velocity vector in (x, y, z) Cartesian space co-ordinates, ρ is the density of the fluid, μ is the dynamic viscosity of the fluid and (g_x, g_y, g_z) is the acceleration due to gravity. It may be noted that the conservative variable p/β in the mass equation is an artificial term added for coupling the mass and momentum equations. Also, this modification converts the pseudo-time derivative along with the convective flux into a hyperbolic system. Hence, Riemann solvers can be used for calculation of convective flux.

The numerical simulation of two-phase flow, brings in an additional fifth equation for advection of volume fraction, C, and a tensor associated with surface tension $\mathbf{T}$. The details of surface tension are explained in section 3.6. An interface compression term, $\nabla \cdot \mathbf{F}_c$ is introduced to ensure sharp interfaces, the details of which are given in section 3.8. The density, ρ, and the dynamic viscosity, μ, are defined as a function of volume fraction as,

$$\rho = \rho(C) = \rho_1 C + \rho_2 (1 - C) = (\rho_1 - \rho_2) C + \rho_2, \tag{3.2}$$

$$\mu = \mu(C) = \mu_1 C + \mu_2 (1 - C) = (\mu_1 - \mu_2) C + \mu_2 \tag{3.3}$$

where, ρ_1 is the density of first fluid, ρ_2 is the density of the second fluid, μ_1 is the dynamic viscosity of the first fluid and μ_2 is the dynamic viscosity of the second fluid.

3.2 Finite volume method

The finite volume discretization of the governing equations (3.1) is obtained by integrating the equations over an arbitrary volume in space. The volume of a cell P is denoted as Ω_P. Different types of cells used in this work are shown in Fig. 3.1. Using the Gauss-divergence theorem, the volume integrals reduce to surface integrals resulting in the following discretized form of the governing equations.

$$\Omega_P \frac{\partial \overline{\mathbf{U}}}{\partial \tau} + \Omega_P \frac{\partial \overline{\mathbf{W}}}{\partial t} + \sum_{m=1}^{M} \left(\mathbf{F}\, n_x + \mathbf{G}\, n_y + \mathbf{H}\, n_z \right)_m \Gamma_m + \sum_{m=1}^{M} \left(\mathbf{F}_c \cdot \hat{n}\, \Gamma \right)_m$$
$$= \sum_{m=1}^{M} \left(\mathbf{F_v}\, n_x + \mathbf{G_v}\, n_y + \mathbf{H_v}\, n_z \right)_m \Gamma_m + \sum_{m=1}^{M} \left(\mathbf{T} \cdot \hat{n}\, \Gamma \right)_m + \Omega_P\, \overline{\mathbf{S}}\,. \quad (3.4)$$

The over-bar indicates cell averaged value of a variable inside cell P. The cell is enclosed by M planar faces, denoted by $m = 1 \ldots M$, with the m^{th} face having a surface area of Γ_m. The unit normal vector $\hat{n} = \left(n_x, n_y, n_z\right)$ for each of the planar faces points outward of the cell P. It may be noted that, in case of a second order accurate finite volume method, the cell-averaged values can be obtained by evaluating the variables at cell-centroid [116].

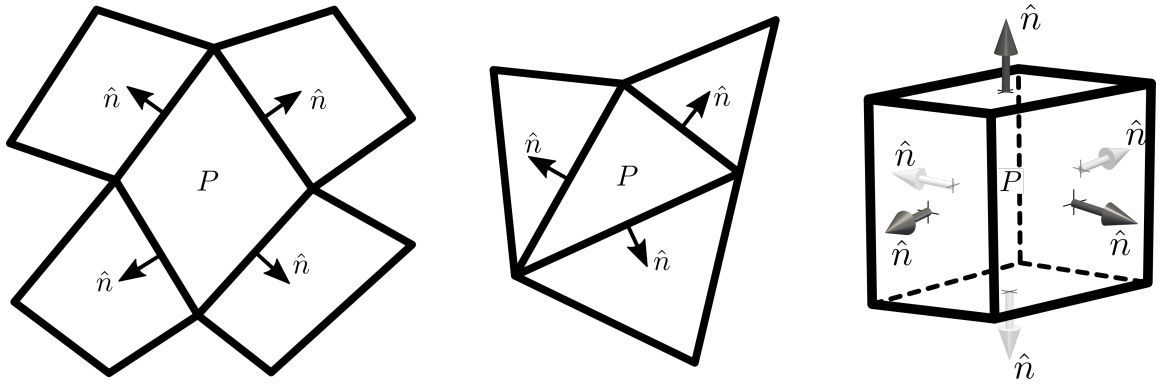


Fig. 3.1: Notations used in the finite volume formulation for different cell types.

3.3 Dual time stepping method

The dual-time stepping procedure proposed by Jameson [84] for compressible flows can be utilized along with the artificial compressibility formulation to obtain a time accurate solution for incompressible flows. The basic pseudo-compressibility procedure followed here is similar to the one described by Gaitonde [76].

3.3.1 Treatment of pseudo- and real-time derivatives

The finite volume formulation (3.4) can be re-written as an ordinary differential equation (ODE) in pseudo-time as,

$$\Omega_P \frac{d\overline{\mathbf{U}}}{d\tau} = -R\left(\overline{\mathbf{U}}\right)\,, \quad (3.5)$$

where, $R\left(\overline{\mathbf{U}}\right)$ is called the residual of cell P and can be written as,

$$R\left(\overline{\mathbf{U}}\right) = \Omega_P \frac{\partial \overline{\mathbf{W}}}{\partial t} + \sum_{m=1}^{M} \left(\mathbf{F}\, n_x + \mathbf{G}\, n_y + \mathbf{H}\, n_z\right)_m \Gamma_m + \sum_{m=1}^{M} \left(\mathbf{F}_c \cdot \hat{n}\, \Gamma\right)_m - \sum_{m=1}^{M} \left(\mathbf{F}_\mathbf{v}\, n_x + \mathbf{G}_\mathbf{v}\, n_y + \mathbf{H}_\mathbf{v}\, n_z\right)_m \Gamma_m - \sum_{m=1}^{M} \left(\mathbf{T} \cdot \hat{n}\, \Gamma\right)_m - \Omega_P\, \overline{\mathbf{S}}\,. \quad (3.6)$$

Based on the ODE given by equation (3.5), the solution is updated in pseudo-time using the two-stage stability preserving Runge-Kutta method [117], which can be written as,

$$\begin{aligned} \mathbf{U}^{(0)} &= \overline{\mathbf{U}}^k; \\ \mathbf{U}^{(1)} &= \mathbf{U}^{(0)} - \tfrac{\Delta\tau}{\Omega_P} R\left(\mathbf{U}^{(0)}\right); \\ \mathbf{U}^{(2)} &= \tfrac{1}{2}\mathbf{U}^{(0)} + \tfrac{1}{2}\mathbf{U}^{(1)} - \tfrac{1}{2}\tfrac{\Delta\tau}{\Omega_P} R\left(\mathbf{U}^{(1)}\right); \\ \overline{\mathbf{U}}^{k+1} &= \mathbf{U}^{(2)}; \end{aligned} \quad (3.7)$$

where, $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$ are the solutions at intermediate pseudo-time steps, $\overline{\mathbf{U}}^k$ is the solution at current pseudo-time and $\overline{\mathbf{U}}^{k+1}$ is the updated solution in pseudo-time. The pseudo-time iterations are carried out until the L_2-norm of the residual is reduced to a very small value, of 10^{-3}, for all the conservative variables. Thus recovering the original unsteady two-phase flow equations, such that $\partial\overline{\mathbf{U}}/\partial\tau \approx 0$. The L_2-norm of the residual is defined as given by Wang et. al [113]. It may be noted that all the above variable vectors are functions of $\overline{\mathbf{U}}$, i.e. $\overline{\mathbf{W}}, \mathbf{F}, \mathbf{G}, \mathbf{H}, \mathbf{F}_c, \mathbf{F}_\mathbf{v}, \mathbf{G}_\mathbf{v}, \mathbf{H}_\mathbf{v}, \mathbf{T}$ and $\overline{\mathbf{S}}$ are dependent on $\overline{\mathbf{U}}$. Hence, updating $\overline{\mathbf{U}}$ will modify the residual at every Runge-Kutta step.

The real-time derivative is numerically approximated using a three-point implicit backward difference formula, which can be easily derived to be,

$$\frac{\partial \overline{\mathbf{W}}}{\partial t} = \frac{3\,\overline{\mathbf{W}}^k - 4\,\overline{\mathbf{W}}^n + \overline{\mathbf{W}}^{n-1}}{2\Delta t}\,. \quad (3.8)$$

where, Δt is the chosen time step to evolve the solution in real-time (this time step is problem dependent and is defined for each problem), superscript k denotes the current solution in pseudo-time, superscript n denotes the current solution in real-time and superscript $n-1$ denotes the previous solution in real-time. The solution at previous real-time level is not available at the beginning of the simulation, hence, a two-point backward difference approximation is used. The two-point backward difference approximation can be written as,

$$\frac{\partial \overline{\mathbf{W}}}{\partial t} = \frac{\overline{\mathbf{W}}^k - \overline{\mathbf{W}}^n}{\Delta t}\,. \quad (3.9)$$

The pseudo-time step size, $\Delta\tau$, is limited by the stability requirements of the scheme. The pseudo-time step is dependent on the convective flux, viscous flux and the surface

tension flux. Therefore,

$$\Delta\tau \leq \min\left(\Delta\tau_{cv}, \Delta\tau_s\right) \tag{3.10}$$

where, $\Delta\tau_{cv}$ is the time step constraint due to convective and viscous flux, and $\Delta\tau_s$ is the time step constraint imposed by surface tension. The largest time step for a cell P due to convective and viscous fluxes may be estimated using [118–120]

$$\left(\Delta\tau_{cv}\right)_P = \frac{\Omega_P}{\left(\Lambda_c + K\Lambda_v\right)_P}\,, \tag{3.11}$$

where, Λ_c and Λ_v are estimates of the convective and viscous spectral radii for the cell respectively. The spectral radii are calculated as,

$$\Lambda_c = \sum_{m=1}^{M} \left(\max|\lambda|\ \Gamma\right)_m \quad \text{and} \quad \Lambda_v = \frac{4}{3\,\Omega_P} \sum_{m=1}^{M} \left(\frac{\mu}{\rho}\Gamma^2\right)_m\,, \tag{3.12}$$

where, $\max|\lambda|$ is the maximum absolute eigenvalue of the convective flux Jacobian at the face m and $K = 4$ is an empirically determined coefficient in [118]. The time constraint due to surface tension is estimated based on the expressions given in [23, 29] to be,

$$\Delta\tau_s = \left[\frac{\left(\rho_1 + \rho_2\right)\Omega_P}{\pi\,\sigma}\right]^{1/2}\,. \tag{3.13}$$

The pseudo-time step is further limited by the real-time step, Δt, which is estimated using linear stability analysis in [121] as,

$$\Delta\tau \leq \frac{2}{3}\Delta t\,. \tag{3.14}$$

Finally a scaling factor, similar to the Courant number, is used to obtain the following expression for pseudo-time step,

$$\Delta\tau = \mathrm{CFL}\left[\min\left(\Delta\tau_{cv}, \Delta\tau_s, \frac{2}{3}\Delta t\right)\right]\,. \tag{3.15}$$

The scaling factor of $\mathrm{CFL} = 1$ is used in all the simulations in this work. A local-time stepping algorithm is used to accelerate the convergence, where the value of $\Delta\tau$ may be different for each cell, based on the local flow field.

3.4 Calculation of convective flux

In the finite volume formulation, cell-averaged values of the conservative variable, $\mathbf{U}$, are available in each cell. Using this information, the flux needs to be calculated through cell faces. The flux is calculated in two distinct steps. In the first step, a smooth solution is reconstructed within each cell using the cell-averaged values of the neighboring cells, to obtain a second-order accurate scheme. In the second step,

the reconstructed solution at the faces is used to obtain the convective flux by using a Riemann solver. These two steps are described in detail in the next two sub-sections.

3.4.1 Solution reconstruction

The conservative variables are reconstructed, using the local cell averaged data, to obtain a smooth distribution inside each cell. This is done by using a truncated Taylor series approximation and a distance weighted least-square approach. Consider a subset of a computational grid, as shown in Fig. 3.2.

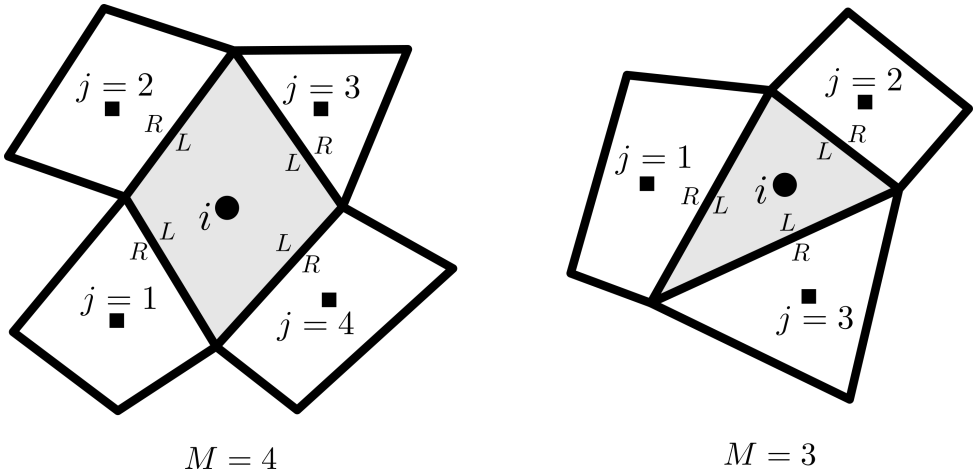


Fig. 3.2: Subset of computational grids consisting of mixed cell types. The centroid of the cell in which the solution is being reconstructed is represented using ● symbol and centroid of neighboring cells is represented using ■ symbol.

The cell, in which the solution reconstruction is carried out, is denoted as i and the face-based neighboring cells are denoted as $j = 1, 2 \ldots M$. Let a scalar ξ represent a single component from vector $\mathbf{U}$. A truncated Taylor series, about the centroid of cell i, can be written for approximating ξ at the centroid of cell j as,

$$\xi_j = \xi_i + \left.\frac{\partial \xi}{\partial x}\right|_i \Delta x_j + \left.\frac{\partial \xi}{\partial y}\right|_i \Delta y_j + \left.\frac{\partial \xi}{\partial z}\right|_i \Delta z_j \tag{3.16}$$

or the change in ξ can be written as,

$$\Delta\xi_j = \left.\frac{\partial \xi}{\partial x}\right|_i \Delta x_j + \left.\frac{\partial \xi}{\partial y}\right|_i \Delta y_j + \left.\frac{\partial \xi}{\partial z}\right|_i \Delta z_j \tag{3.17}$$

where, $\Delta\xi_j = (\xi_j - \xi_i)$, $\Delta x_j = (x_j - x_i)$, $\Delta y_j = (y_j - y_i)$, $\Delta z_j = (z_j - z_i)$. The derivatives of ξ are evaluated at the centroid of cell i. The second and higher order terms of the Taylor series are neglected, which results in a linear approximation that will lead to second order accurate scheme. Writing the equations for $j = 1 \ldots M$, produces M

linear equations and may be written as a over-determined system of equations,

$$\underbrace{\begin{bmatrix} \Delta x_1 & \Delta y_1 & \Delta z_1 \\ \Delta x_2 & \Delta y_2 & \Delta z_2 \\ \vdots & \vdots & \vdots \\ \Delta x_M & \Delta y_M & \Delta z_M \end{bmatrix}}_{S} \underbrace{\begin{bmatrix} (\partial\xi/\partial x)_i \\ (\partial\xi/\partial y)_i \\ (\partial\xi/\partial z)_i \end{bmatrix}}_{\nabla\xi_i} = \underbrace{\begin{bmatrix} \Delta\xi_1 \\ \Delta\xi_2 \\ \vdots \\ \Delta\xi_M \end{bmatrix}}_{\Delta\xi} . \tag{3.18}$$

An inverse-distance based weights, $w_j = 1/\left\|\Delta r_j\right\|_2$, are used to increase the influence of the nearby neighbors, where $\left\|\Delta r_j\right\|_2$ is the Euclidean distance between the points i and j, calculated as,

$$\left\|\Delta r_j\right\|_2 = \sqrt{\Delta x_j^2 + \Delta y_j^2 + \Delta z_j^2} \, . \tag{3.19}$$

The resulting weighted matrix may be written as,

$$W\, S\, \nabla\xi_i = W\, \Delta\xi \, , \tag{3.20}$$

where, $W = \text{diag}\,(w_1, w_2, \ldots, w_M)$. This over-determined system of equations is solved for $\nabla\xi_i$ using the singular value decomposition (SVD) method, using an open-source software library [122]. The SVD method has an advantage that its formulation remains same for 2D and 3D grids and it can also deal with highly skewed grids very well. The method is, however, computationally expensive compared to other direct methods. This is of a less concern here, as the decomposition needs to be computed only once for problems involving stationary grids. The computed derivatives are modified using the Venkatakrishnan slope limiter [116] to avoid spurious oscillations near the interface, due to the discontinuous nature of the volume fraction. The final linearly reconstructed variable, ξ, inside the cell, may be written as,

$$\xi\,(x, y, z) = \xi_i + \mathbf{\Phi}_i \nabla\xi_i \cdot \Delta r \, , \tag{3.21}$$

where, (x, y, z) is a point within or on the boundary of cell i, $\mathbf{\Phi}_i$ is called the limiter for cell i and $\Delta r = (\Delta x, \Delta y, \Delta z)$ with $\Delta x = (x - x_i)$, $\Delta y = (y - y_i)$, $\Delta z = (z - z_i)$. The limiter is defined as [116],

$$\mathbf{\Phi}_i = \min\left(\Phi_1, \Phi_2, \ldots \Phi_M\right) \, , \tag{3.22}$$

where, for each of the face,

$$\Phi_j = \begin{cases} \phi\left(\frac{\xi_i^{\max} - \xi_i}{\xi_j - \xi_i}\right), & \text{if } \xi_j - \xi_i > 0 \\ \phi\left(\frac{\xi_i^{\min} - \xi_i}{\xi_j - \xi_i}\right), & \text{if } \xi_j - \xi_i < 0 \\ 1, & \text{if } \xi_j - \xi_i = 0 \, . \end{cases} \tag{3.23}$$

The smooth function, $\phi(\eta)$, is given by [116],

$$\phi(\eta) = \frac{\eta^2 + 2\eta}{\eta^2 + \eta + 2} \,. \tag{3.24}$$

The reconstruction of solution data within each cell can potentially create discontinuities at the cell faces, resulting in a Riemann problem at each cell face. The reconstructed solution at a face, therefore has two values, which are commonly called as the left state (interpolated value for cell i) and the right state (interpolated value from the neighboring cell). This is depicted in Fig. 3.2 using symbols L and R respectively. The left and the right states are computed at the centroid of the face to obtain a second-order accurate solution. Upon reconstruction of all the components of vector $\mathbf{U}$, using the above procedure, the left and right states, $\mathbf{U}_L$ and $\mathbf{U}_R$, are obtained at the centroid of each face. The convective flux is calculated by using a Riemann solver based on the jump condition, as described in the upcoming sub-section.

3.4.2 Convective flux calculation using Riemann solver

The convective flux in a finite volume formulation is computed using the following three steps:

1. Given the reconstructed conservative variables for a face, $(\mathbf{U}_L, \mathbf{U}_R)$, the transformed variables, in a locally rotated coordinate system, $\left(\hat{\mathbf{U}}_L, \hat{\mathbf{U}}_R\right) = (T\,\mathbf{U}_L, T\,\mathbf{U}_R)$ are calculated, which is elaborated below.

2. Using $\left(\hat{\mathbf{U}}_L, \hat{\mathbf{U}}_R\right)$, the convective flux, $\hat{\mathbf{F}}_f$, for each face of the cell is calculated.

3. The convective flux required in the three-dimensional finite volume formulation, $T^{-1}\hat{\mathbf{F}}_f$ is then calculated, which can be added over the faces of the cell to obtain total flux for the cell.

The above steps are elaborated in the upcoming text.

As the interest in this subsection is to obtain the convective flux, let us consider a simplified system with only the pseudo-time derivative and the convective flux vectors. Additionally, consider a rotated coordinate system $(\hat{x}, \hat{y}, \hat{z})$, as shown schematically in Fig. 3.3, where $\hat{x}$ is the direction normal to the face and $\hat{y}$ and $\hat{z}$ are tangential to the face. In such a coordinate system, only the $\hat{x}$-split three-dimensional governing equations are needed for calculation of convective flux, as the other two flux components will not contribute in the finite volume flux computation. Thus, to compute the convective flux using a Riemann solver the following simplified system of governing equations is sufficient,

$$\frac{\partial \hat{\mathbf{U}}}{\partial \tau} + \frac{\partial \hat{\mathbf{F}}}{\partial \hat{x}} = 0 \,, \tag{3.25}$$

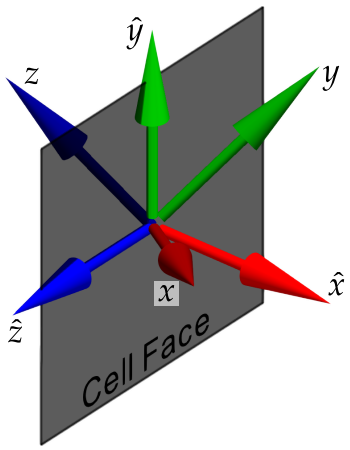


Fig. 3.3: Schematic to display the rotated coordinate system $(\hat{x}, \hat{y}, \hat{z})$ in which the convective flux is computed using a Riemann solver. The coordinate axis (x, y, z) represents the physical Cartesian coordinate system.

where, $\hat{\mathbf{F}} = \mathbf{F}\left(\hat{\mathbf{U}}\right)$ and $\hat{\mathbf{U}} = T\,\mathbf{U}$, with the expression for $\mathbf{U}$ same as in equation (3.1). The transformation matrix T can be obtained to be,

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & n_x & n_y & n_z & 0 \\ 0 & t_{1x} & t_{1y} & t_{1z} & 0 \\ 0 & t_{2x} & t_{2y} & t_{2z} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.26}$$

where, $\hat{n} = \left(n_x, n_y, n_z\right)$ is the unit vector normal to the face, pointing from left state to right state (along $\hat{x}$ direction), and, $\hat{t}_1 = \left(t_{1x}, t_{1y}, t_{1z}\right)$ and $\hat{t}_2 = \left(t_{2x}, t_{2y}, t_{2z}\right)$ are unit orthogonal vectors tangent to the face. The transformed conservative variable vector, $\hat{\mathbf{U}}$, can be written as,

$$\hat{\mathbf{U}} = \begin{bmatrix} p/\beta \\ \rho\,\left(u\,n_x + v\,n_y + w\,n_z\right) \\ \rho\,\left(u\,t_{1x} + v\,t_{1y} + w\,t_{1z}\right) \\ \rho\,\left(u\,t_{2x} + v\,t_{2y} + w\,t_{2z}\right) \\ C \end{bmatrix} = \begin{bmatrix} p/\beta \\ \rho\,\hat{u} \\ \rho\,\hat{v} \\ \rho\,\hat{w} \\ C \end{bmatrix},$$

where, $\hat{u} = u\,n_x + v\,n_y + w\,n_z$; $\hat{v} = u\,t_{1x} + v\,t_{1y} + w\,t_{1z}$ and $\hat{w} = u\,t_{2x} + v\,t_{2y} + w\,t_{2z}$. The Riemann solver for calculation of convective flux takes the left and the right states at the face, $\left(\hat{\mathbf{U}}_L, \hat{\mathbf{U}}_R\right)$, as inputs and returns the flux through the face, $\hat{\mathbf{F}}_f$ as an output.

$$\hat{\mathbf{F}}_f = \mathbf{F}_f\left(\hat{\mathbf{U}}_L, \hat{\mathbf{U}}_R\right), \tag{3.27}$$

where, $\hat{\mathbf{F}}_f$ is the numerical flux at the cell face. Finally, to compute the required flux in the finite volume formulation in the original coordinate system (x, y, z), the inverse

transform can be applied,

$$\mathbf{F}\, n_x + \mathbf{G}\, n_y + \mathbf{H}\, n_z = T^{-1}\hat{\mathbf{F}}_f \,. \tag{3.28}$$

Since the transformation matrix, T, is orthonormal, its inverse can be calculated by simply transposing the matrix,

$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & n_x & t_{1x} & t_{2x} & 0 \\ 0 & n_y & t_{1y} & t_{2y} & 0 \\ 0 & n_z & t_{1z} & t_{2z} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} . \tag{3.29}$$

The different ways of obtaining an accurate convective flux, $\hat{\mathbf{F}}_f$, using a Riemann solver is presented in chapter 4.

3.5 Calculation of viscous flux

In the finite volume discretization, the viscous flux needs to be calculated at the cell faces. Hence, the velocity gradients, ∇V, and the dynamic coefficient of viscosity, μ, need to be computed at the cell faces. The velocity gradients can be written as,

$$\nabla V = \begin{bmatrix} \partial u/\partial x & \partial u/\partial y & \partial u/\partial z \\ \partial v/\partial x & \partial v/\partial y & \partial v/\partial z \\ \partial w/\partial x & \partial w/\partial y & \partial w/\partial z \end{bmatrix} . \tag{3.30}$$

For numerical computation of derivatives at the face, cell values in the vicinity of the face are needed. Consider the cell i shown in Fig. 3.4. The set of neighbors of a face, say $j = 1$, is constructed by collecting the neighbors of all the nodes of the face. In the example shown in Fig. 3.4, this will result in a set of 6 neighbors i.e. cells $n = 1 \ldots 5$ and cell i. Here a quadrilateral cell is considered, however the same strategy can be followed for all types of cells, in two- and three-dimensions.

In a finite volume formulation, as we know the cell averaged values of the conservative variables, the required derivatives of velocity are calculated in two steps. The first step is to compute the gradients of the conservative variables at the face; and the second step is to convert the gradients of conservative variables to gradients of velocity by using the Jacobian of transformation. The gradients of the conservative variables are calculated by using a truncated Taylor series. Consider a variable, ξ, to be a component of the conservative variable vector $\mathbf{U}$. A truncated Taylor series to obtain a linear

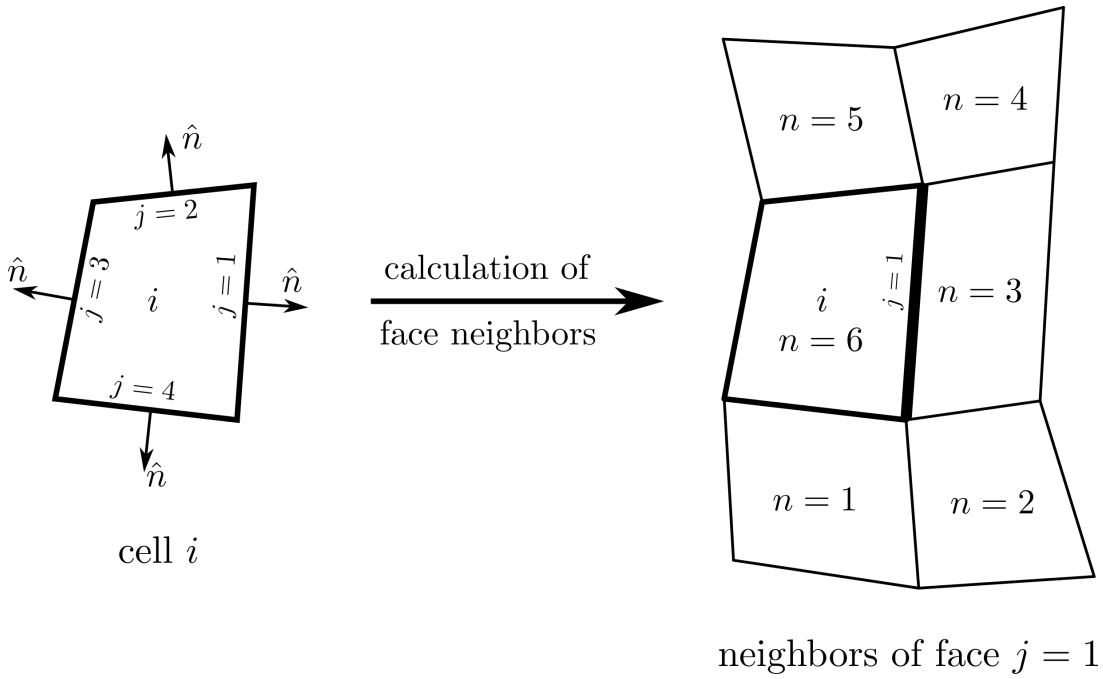


Fig. 3.4: An example of face neighbors for numerical calculation of gradients at the cell face $j=1$.

approximation of ξ can be written as,

$$\xi_n = \xi_j + \left.\frac{\partial \xi}{\partial x}\right|_j \Delta x_n + \left.\frac{\partial \xi}{\partial y}\right|_j \Delta y_n + \left.\frac{\partial \xi}{\partial z}\right|_j \Delta z_n \,, \tag{3.31}$$

where, the derivatives are evaluated at the centroid of face j. $(\Delta x_n, \Delta y_n, \Delta z_n)$ is the vector from the centroid of face j to the centroid of neighbor cell n. Writing the above equation for all the neighbors, $n = 1, 2 \ldots N$, of a face will result in the following system:

$$\underbrace{\begin{bmatrix} 1 & \Delta x_1 & \Delta y_1 & \Delta z_1 \\ 1 & \Delta x_2 & \Delta y_2 & \Delta z_2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \Delta x_N & \Delta y_N & \Delta z_N \end{bmatrix}}_{S} \cdot \underbrace{\begin{bmatrix} \xi \\ \partial\xi/\partial x \\ \partial\xi/\partial y \\ \partial\xi/\partial z \end{bmatrix}_j}_{\xi'_j} = \underbrace{\begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_N \end{bmatrix}}_{\xi_n} . \tag{3.32}$$

The equations are weighted using inverse-distance based weights, $w_n = 1/\left\|\Delta r_n\right\|$, where $\left\|\Delta r_n\right\| = \sqrt{\Delta x_n^2 + \Delta y_n^2 + \Delta z_n^2}$. The system of equations with weights can be written as,

$$W\, S\, \xi'_j = W\, \xi_n \,, \tag{3.33}$$

with $W = \text{diag}\,(w_1, w_2 \ldots w_N)$. This results in an over-determined system of equations with four unknowns, ξ, $\partial\xi/\partial x$, $\partial\xi/\partial y$ and $\partial\xi/\partial z$ at the centroid of face j. The system is solved by using singular value decomposition (SVD) method using an open source mathematical library [122].

The above procedure can be followed for each component of the conservative vari-

able vector, **U**, to obtain its value and gradient at the cell face. The gradients of conservative variables at a face j, can be packed together as,

$$\nabla \mathbf{U}_j = \begin{bmatrix} \partial U_1/\partial x & \partial U_1/\partial y & \partial U_1/\partial z \\ \partial U_2/\partial x & \partial U_2/\partial y & \partial U_2/\partial z \\ \partial U_3/\partial x & \partial U_3/\partial y & \partial U_3/\partial z \\ \partial U_4/\partial x & \partial U_4/\partial y & \partial U_4/\partial z \\ \partial U_5/\partial x & \partial U_5/\partial y & \partial U_5/\partial z \end{bmatrix}, \tag{3.34}$$

where, $U_1 = p/\beta$, $U_2 = \rho\, u$, $U_3 = \rho\, v$, $U_4 = \rho\, w$, $U_5 = C$.

Once the gradients of the conservative variables at the cell face are computed, the second step is to calculate the velocity gradients by using the transformation,

$$\nabla V_j = J\, \nabla \mathbf{U}_j\,, \tag{3.35}$$

where J is the Jacobian, obtained as,

$$J = \frac{\partial V}{\partial \mathbf{U}} = \begin{bmatrix} 0 & 1/\rho_j & 0 & 0 & \frac{(\rho_2-\rho_1)u_j}{\rho_j} \\ 0 & 0 & 1/\rho_j & 0 & \frac{(\rho_2-\rho_1)v_j}{\rho_j} \\ 0 & 0 & 0 & 1/\rho_j & \frac{(\rho_2-\rho_1)w_j}{\rho_j} \end{bmatrix}. \tag{3.36}$$

As a part of the solution of system (3.33), the interpolated values of conservative variables, $(\rho u)_j$, $(\rho v)_j$, $(\rho w)_j$ and C_j, are also obtained at the cell face. The density, ρ_j, at the cell face can be calculated using equation (3.2). The velocity components at the cell face can be calculated as,

$$u_j = (\rho u)_j / \rho_j, \quad v_j = (\rho v)_j / \rho_j, \quad w_j = (\rho w)_j / \rho_j\,, \tag{3.37}$$

which can be used for calculation of the Jacobian, J. The value of the coefficient of dynamic viscosity, μ_j, at the cell face can be calculated using equation (3.3). The viscous flux can be calculated using the obtained gradients of velocity and coefficient of dynamic viscosity.

3.6 Calculation of surface tension flux

Surface tension is a force which acts only at the interface between two fluids, which is a result of inter-molecular forces. This force acts in such a way so as to minimize the surface area of the interface. The magnitude of the resultant force is proportional to surface tension coefficient, σ, corresponding to the fluids. The value of σ can be obtained experimentally for different fluid combinations from literature [64]. This force is commonly modeled in numerical simulations, using interface capturing methods,

by the continuum surface force (CSF) [29] or the continuum surface stress (CSS) [65] models. In this work, the CSS method is used for modeling surface tension as it does not require explicit calculation of interface curvature. Also, based on numerical experiments it was observed in [65], that the CSS model does not require smoothing of VOF function. Hence, it is very straightforward and quite natural to combine the CSS model within the VOF framework in a conservative finite volume formulation. The CSS model describes the surface tension force in a momentum equation as,

$$\mathbf{F_s} = \sigma \left(|\nabla C| \; I - \frac{\nabla C \otimes \nabla C}{|\nabla C|} \right) , \tag{3.38}$$

where, ∇C is the gradient of volume fraction, $|\nabla C|$ is the magnitude of the gradient vector and I is a 3×3 identity matrix. The surface tension force in the governing equations (3.1) therefore becomes,

$$\nabla \cdot \mathbf{T} = \begin{bmatrix} 0 \\ \nabla \cdot \mathbf{F_s} \\ 0 \end{bmatrix} .$$

The surface tension force is set to zero away from the interface, explicitly, when the value of $|\nabla C| < 10^{-6}$ to avoid division by zero. The tensor $\nabla C \otimes \nabla C$ is defined as,

$$\nabla C \otimes \nabla C = \nabla C^T \nabla C = \begin{bmatrix} \left(\frac{\partial C}{\partial x}\right)^2 & \frac{\partial C}{\partial x}\frac{\partial C}{\partial y} & \frac{\partial C}{\partial x}\frac{\partial C}{\partial z} \\ \frac{\partial C}{\partial x}\frac{\partial C}{\partial y} & \left(\frac{\partial C}{\partial y}\right)^2 & \frac{\partial C}{\partial y}\frac{\partial C}{\partial z} \\ \frac{\partial C}{\partial x}\frac{\partial C}{\partial z} & \frac{\partial C}{\partial y}\frac{\partial C}{\partial z} & \left(\frac{\partial C}{\partial z}\right)^2 \end{bmatrix} . \tag{3.39}$$

For incorporating the CSS model into finite volume formulation, the value of $\mathbf{F_s}$ has to be computed at the cell face. The gradients at the face,$\nabla \mathbf{U}_j$, are computed during calculation of viscous flux, which is described in sub-section 3.5. The gradient of volume fraction for a face j is calculated as,

$$\nabla C_j^T = J \, \nabla \mathbf{U}_j \, , \tag{3.40}$$

where J can be obtained to be,

$$J = \frac{\partial C}{\partial \mathbf{U}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix} . \tag{3.41}$$

3.7 Treatment of source term

The source term in the finite volume formulation is given by,

$$\Omega_P \, \overline{\mathbf{S}}, \quad \text{with} \quad \overline{\mathbf{S}} = \left[0, \overline{\rho}\, g_x, \overline{\rho}\, g_y, \overline{\rho}\, g_z, 0\right]^T ,$$

where, Ω_P is the volume of the cell. The average value of the density, $\overline{\rho}$, can be computed simply by using equation (3.2) and substituting the average value of volume fraction, $\overline{C}$, which is available in the finite volume formulation (3.5).

3.8 Interface compression

Approximate Riemann solvers inherently introduce some numerical dissipation to ensure stability. This is acceptable in case of compressible fluid flows, where Riemann solvers are extensively used, as the governing equations itself aids compression of discontinuities, such as shocks, to produce correct solutions. The passively advected volume fraction, on the other hand, will keep spreading out the discontinuous volume fraction at the interface, if not corrected artificially. To ensure sharp interfaces, an interface compression term, $\nabla \cdot \mathbf{F}_c$, is added to the governing equations (3.1). This technique is borrowed from [49], which is also implemented in OpenFOAM [50], and modified based on later advancements in the technique [51, 123]. The compression term can be written as,

$$\nabla \cdot \mathbf{F}_c = \Big[0, 0, 0, 0, \nabla \cdot \vec{V}_c \; (1 - C) \; C\Big]^T .$$

The resulting VOF equation becomes,

$$\frac{\partial C}{\partial \tau} + \frac{\partial C}{\partial t} + \nabla \cdot \Big(\vec{V}\, C + \vec{V}_c \; (1 - C) \; C\Big) = 0 \, , \tag{3.42}$$

where, $\vec{V} = (u, v, w)$ is the velocity field of the fluids and $\vec{V}_c$ is the compressive velocity field. The term $(1 - C)\; C$, ensures that the interface compression is active only near the interface.

The advection part of the flux, due to the term $\vec{V}\, C$, is calculated by the HLLC-VOF Riemann solver, therefore only the compressive part of the flux, due to the term $\vec{V}_c \; (1 - C) \; C$, needs to be computed. The remaining equation, without $\vec{V}\, C$, can be viewed like an advection equation of C, with $\vec{V}_c \; (1 - C)$ as the advection velocity field. The required compressive flux can therefore be calculated using a simple scalar upwind solver using the left and right states of volume fraction, C_L and C_R,

$$F_{fc} = \begin{cases} \lambda_c \, C_L & \lambda_c > 0 \, ; \\ \lambda_c \, C_R & \text{otherwise} \, . \end{cases} \tag{3.43}$$

The wave speed is calculated as $\lambda_c = \left(1 - C_f\right) \vec{V}_c \cdot \hat{n}$. The volume fraction at the cell

face is denoted as C_f. The compressive velocity field is computed based on literature [49] as,

$$\vec{V}_c = \zeta\, \eta \left| \vec{V}_f \cdot \hat{n} \right| \hat{n}_i \,, \tag{3.44}$$

where, ζ is the compressive coefficient which can be chosen in the range of 0 to 1. In this work, a value of $\zeta = 0.3$ is used. η is called the dynamic interface compression coefficient which is given as $\eta = \sqrt{|\hat{n} \cdot \hat{n}_i|}$. The unit vector normal to cell face is $\hat{n}$ and $\hat{n}_i$ is the unit vector perpendicular to fluid interface, calculated as,

$$\hat{n}_i = \begin{cases} \nabla C_f / \left| \nabla C_f \right| & \text{if } \left| \nabla C_f \right| > 10^{-6} \,, \\ 0 & \text{otherwise} \,, \end{cases} \tag{3.45}$$

to avoid division by zero. The velocity at the cell face, $\vec{V}_f$; the volume fraction at the cell face, C_f; and the gradient of volume fraction at the cell face, ∇C_f, are calculated as a part of viscous flux calculation (explained in section 3.5) and are reused for interface compression. The value obtained by equation (3.43) is simply added to the fifth component of flux obtained by the Riemann solver to avoid smearing of the interface.

3.9 Application of initial conditions

An accurate calculation of the initial cell-averaged values of conservative variables is important in two-phase flow problems, as the solution is sensitive to initial position of the fluid interface. To obtain the initial cell-averaged values, an integration of the initial distribution needs to be computed over each cell. A numerical integration is carried out, for each cell, by subdividing the cell into smaller parts and then performing a Riemann sum. In case of quadrilateral (and hexahedral) cells, the cell is divided by using transfinite interpolation with 20 divisions (4 divisions for hexahedral) along each computational coordinate direction. This results in 20×20 sub-quadrilaterals ($4 \times 4 \times 4$ sub-hexahedrals) per cell. In case of triangular cells, the sub-triangles are generated by using 5 levels of recursive sub-division with each level diving a triangle into 4 equal sub-triangles. This results in 1024 sub-triangles per cell.

3.10 Application of boundary conditions

The boundary conditions are implemented using ghost cells (also known as dummy cells). The ghost cells are additional finite volume cells added outside the computational domain so that gradients at the boundary can be specified or computed. There are three types of boundary conditions used in this work: the stationary slip wall (also called inviscid wall), stationary no-slip wall (also called viscous wall) and symmetry boundary condition. In all these boundary conditions the normal component of velocity is zero, therefore the convective flux is given by $\left[0, p\, n_x, p\, n_y, p\, n_z, 0\right]^T$. The calcula-

tion of viscous and the surface tension flux at the boundary depend on the gradients of variables at the boundary face, therefore the ghost cells have to be set appropriately.

In case of the slip wall boundary, the inside velocity vector is mirrored about the boundary plane, while the pressure and volume fraction are extrapolated using corresponding values and gradients from the inside cell. In case of the no-slip boundary, the negative of velocity vector is copied to the ghost cell, while the pressure and volume fraction are extrapolated. In case of the symmetry boundary, the inside velocity vector is mirrored about the boundary plane, while the pressure and volume fraction are copied as it is to the ghost cell.

Chapter 4

Novel Riemann Solvers for Incompressible Two-phase Flows

In numerical simulation of multi-fluid flows, it is very crucial to accurately capture the interface separating the fluids, which has the properties of a contact wave [45]. As the HLLC (Harten Lax van-Leer with contact) formulation has the ability to model contact waves exactly, a novel HLLC-type Riemann solver is developed for calculation of convective flux in a system where the incompressible Navier–Stokes equations are tightly coupled with the volume of fluid (VOF) equation (3.1). The new Riemann solver is named as HLLC-VOF Riemann solver.

The HLLC-VOF solver is very promising as it has been proven in [90] that the HLL family of solvers will converge to the weak solutions of the conservation laws. Moreover, in the same paper, it has been shown to produce physical and entropy satisfying solutions of the governing equations. Also, the need of an additional wave in the HLL solver for correctly capturing the contact-like phenomena has been pointed out in the literature [88, 90, 92]. The additional wave was included by Toro, *et al*. [92] for compressible flows and by Mandal, *et al*. [72] for single-phase incompressible flows. The current work develops an accurate Riemann solver by including the missing contact wave for two-phase incompressible flows.

The efficacy of the new formulation is evaluated by solving various standard test problems, in two- and three-dimensions, involving two-fluid flows. In the second section of this chapter, the Roe-type Riemann solver is extended by coupling it with an interface compression algorithm designed for unstructured mesh. The results from this solver are compared with the HLLC-VOF solver by solving a few standard test problems. The calculation of convective flux is a two step process: the solution reconstruction and the flux computation using a Riemann solver. These two steps were discussed in detail in section 3.4 in the previous chapter. In this chapter, the formulation of Riemann solvers are presented in detail.

4.1 HLLC-VOF Riemann solver

In the HLLC-VOF solver, a three-wave system is considered with the left moving, right moving and intermediate wave. It is observed, with the help of generalized Riemann invariant analysis (see Appendix B), that across the intermediate wave there are jumps in the tangential component of velocity and volume fraction, while the normal component of velocity and pressure remains continuous. This is similar to the behavior displayed by the contact wave in compressible fluid flows. It is well established in case of compressible flows that Riemann solvers considering the intermediate contact wave, such as HLLC [92], produce superior results. Hence, a Riemann solver for incompressible two-phase flow, which is derived considering the intermediate wave is expected to produce more accurate results in flows involving fluid interfaces. A similar argument is presented in the book by Toro [88] for compressible multi-component flows. Going ahead with this philosophy, a Riemann solver is proposed for incompressible two-phase flows.

4.1.1 Formulation

The procedure followed for the calculation of convective flux using a Riemann solver was discussed in sub-section 3.4.2. The missing piece of the discussion was the calculation of the convective flux, $\hat{\mathbf{F}}_f$ using a Riemann solver. In case of HLLC-VOF Riemann solver, the two initial states, $\left(\hat{\mathbf{U}}_L, \hat{\mathbf{U}}_R\right)$, at the cell face, evolve over pseudo-time, τ, into multiple solution states spreading out in space and time, as shown in Fig. 4.1. These solution states are separated by discontinuities across the waves. The speeds at which the waves move are associated with the eigenvalues of the Jacobian, $\partial\hat{\mathbf{F}}/\partial\hat{\mathbf{U}}$, which can be calculated to be,

$$\frac{\partial\hat{\mathbf{F}}}{\partial\hat{\mathbf{U}}} = \begin{bmatrix} 0 & 1/\rho & 0 & 0 & (\rho_2-\rho_1)\ \hat{u}/\rho \\ \beta & 2\,\hat{u} & 0 & 0 & (\rho_2-\rho_1)\ \hat{u}^2 \\ 0 & \hat{v} & \hat{u} & 0 & (\rho_2-\rho_1)\ \hat{u}\,\hat{v} \\ 0 & \hat{w} & 0 & \hat{u} & (\rho_2-\rho_1)\ \hat{u}\,\hat{w} \\ 0 & C/\rho & 0 & 0 & \rho_2\,\hat{u}/\rho \end{bmatrix} \tag{4.1}$$

The eigenvalues of the Jacobian can be calculated to be,

$$\lambda = [\hat{u}_C - a, \hat{u}, \hat{u}, \hat{u}, \hat{u}_C + a] \tag{4.2}$$

where,

$$\hat{u}_C = \frac{(\rho+\rho_2)\ \hat{u}}{2\,\rho} = \frac{\hat{u}}{2}\left(1+\frac{\rho_2}{\rho}\right); \quad a = \sqrt{\hat{u}_C^2 + \beta/\rho}\,. \tag{4.3}$$

The right eigenvectors of the Jacobian, written as columns of a matrix, are,

$$R_{ev} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ \frac{\rho\,\lambda_1(\hat{u}-\lambda_5)}{\lambda_1-\hat{u}} & 0 & 0 & 1 & \frac{\rho\,\lambda_5\,(\hat{u}-\lambda_1)}{\lambda_5-\hat{u}} \\ \frac{\rho\,\hat{v}\,\lambda_1}{\lambda_1-\hat{u}} & 1 & 0 & 0 & \frac{\rho\,\hat{v}\,\lambda_5}{\lambda_5-\hat{u}} \\ \frac{\rho\,\hat{w}\,\lambda_1}{\lambda_1-\hat{u}} & 0 & 1 & 0 & \frac{\rho\,\hat{w}\,\lambda_5}{\lambda_5-\hat{u}} \\ \frac{C\,\lambda_1}{\lambda_1-\hat{u}} & 0 & 0 & \frac{1}{(\rho_1-\rho_2)\,\hat{u}} & \frac{C\,\lambda_5}{\lambda_5-\hat{u}} \end{bmatrix} . \tag{4.4}$$

There are five waves corresponding to each of the eigenvalues. Since three of the eigenvalues are repeating, it will result in four distinct states separated by three waves as shown in Fig. 4.1. The solution states are denoted by $\hat{\mathbf{U}}_L$, $\hat{\mathbf{U}}_{*L}$, $\hat{\mathbf{U}}_{*R}$, $\hat{\mathbf{U}}_R$ and the wave speeds are denoted by S_L, S_* and S_R.

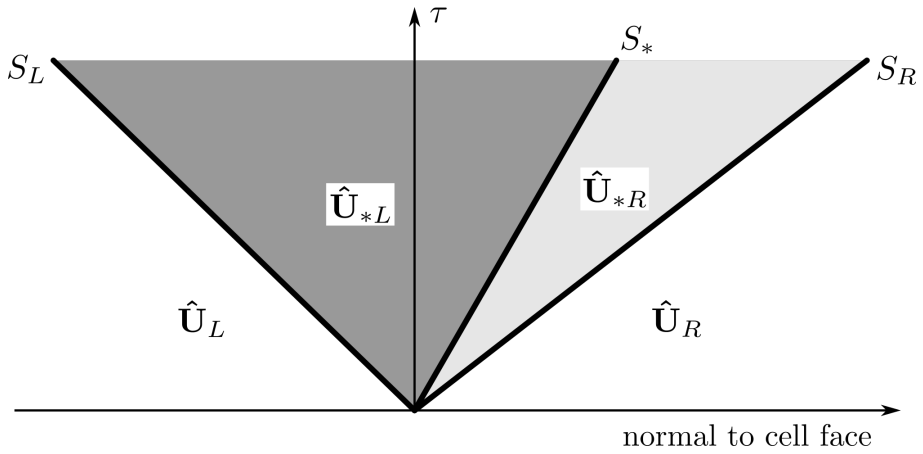


Fig. 4.1: Schematic diagram depicting the wave structure of a Riemann solver with three distinct waves.

This results in the following definition of flux,

$$\hat{\mathbf{F}}_f = \begin{cases} \hat{\mathbf{F}}_L & S_L \geq 0 \\ \hat{\mathbf{F}}_{*L} & S_L \leq 0 \leq S_* \\ \hat{\mathbf{F}}_{*R} & S_* \leq 0 \leq S_R \\ \hat{\mathbf{F}}_R & S_R \leq 0 \end{cases} \tag{4.5}$$

which depends on the magnitude of the wave speed. Here,

$$\hat{\mathbf{F}}_L = \mathbf{F}\left(\hat{\mathbf{U}}_L\right) \;; \quad \hat{\mathbf{F}}_R = \mathbf{F}\left(\hat{\mathbf{U}}_R\right) \; . \tag{4.6}$$

The evaluation of fluxes at intermediate states satisfy the Rankine-Hugoniot jump conditions across the left and right wave,

$$\hat{\mathbf{F}}_{*L} = \hat{\mathbf{F}}_L + S_L\left(\hat{\mathbf{U}}_{*L} - \hat{\mathbf{U}}_L\right) \; ; \tag{4.7}$$

$$\hat{\mathbf{F}}_{*R} = \hat{\mathbf{F}}_R + S_R \left(\hat{\mathbf{U}}_{*R} - \hat{\mathbf{U}}_R \right) \ . \tag{4.8}$$

The intermediate states, $\hat{\mathbf{U}}_{*L}$, $\hat{\mathbf{U}}_{*R}$, and the intermediate wave speed, S_*, can be calculated by using the Rankine-Hugoniot jump conditions and imposing additional conditions obtained from generalized Riemann invariant analysis (please refer to Appendix B for details). After a considerable amount of mathematical rigor all the above unknowns can be estimated (the detailed derivation is provided in Appendix A). All the expressions necessary for calculation of convective flux are concisely given below[†].

$$\hat{\mathbf{F}}_L = \mathbf{F}\left(\hat{\mathbf{U}}_L\right) = \begin{bmatrix} \hat{u} \\ \rho\,\hat{u}^2 + p \\ \rho\,\hat{u}\,\hat{v} \\ \rho\,\hat{u}\,\hat{w} \\ C\,\hat{u} \end{bmatrix}_L ; \qquad \hat{\mathbf{F}}_R = \mathbf{F}\left(\hat{\mathbf{U}}_R\right) = \begin{bmatrix} \hat{u} \\ \rho\,\hat{u}^2 + p \\ \rho\,\hat{u}\,\hat{v} \\ \rho\,\hat{u}\,\hat{w} \\ C\,\hat{u} \end{bmatrix}_R \tag{4.9}$$

$$\boxed{\rho_L = (\rho_1 - \rho_2)\,[C]_L + \rho_2} \qquad \boxed{\rho_R = (\rho_1 - \rho_2)\,[C]_R + \rho_2} \tag{4.10}$$

$$\hat{u}_{CL} = [\hat{u}]_L \, \frac{(\rho_L + \rho_2)}{2\,\rho_L} \qquad \hat{u}_{CR} = [\hat{u}]_R \, \frac{(\rho_R + \rho_2)}{2\,\rho_R} \tag{4.11}$$

$$a_L = \sqrt{\hat{u}_{CL}^2 + \beta/\rho_L}; \qquad a_R = \sqrt{\hat{u}_{CR}^2 + \beta/\rho_R} \tag{4.12}$$

$$\boxed{S_L = \min\left(\hat{u}_{CL} - a_L, \hat{u}_{CR} - a_R\right)} \qquad \boxed{S_R = \max\left(\hat{u}_{CL} + a_L, \hat{u}_{CR} + a_R\right)} \tag{4.13}$$

$$\boxed{[p/\beta]_{*L} = [p/\beta]_{*R} = [p/\beta]_* = \frac{[\hat{u}]_L - [\hat{u}]_R + S_R\,[p/\beta]_R - S_L\,[p/\beta]_L}{S_R - S_L}} \tag{4.14}$$

$$\boxed{\hat{u}_{*L} = \hat{u}_{*R} = \hat{u}_* = S_* = \frac{S_R\,[\rho\,\hat{u}]_R - S_L\,[\rho\,\hat{u}]_L - \left[\rho\,\hat{u}^2 + p\right]_R + \left[\rho\,\hat{u}^2 + p\right]_L}{S_R\rho_R - S_L\rho_L - (\rho_1 - \rho_2)\left([C\,\hat{u}]_R - [C\,\hat{u}]_L\right)}} \tag{4.15}$$

$$\boxed{[C]_{*L} = \frac{S_L\,[C]_L - [C\,\hat{u}]_L}{S_L - S_*}} \qquad \boxed{[C]_{*R} = \frac{S_R\,[C]_R - [C\,\hat{u}]_R}{S_R - S_*}} \tag{4.16}$$

$$\boxed{\rho_{*L} = (\rho_1 - \rho_2)\,[C]_{*L} + \rho_2} \qquad \boxed{\rho_{*R} = (\rho_1 - \rho_2)\,[C]_{*R} + \rho_2} \tag{4.17}$$

$$\boxed{[\rho\,\hat{u}]_{*L} = \rho_{*L}\hat{u}_*} \qquad \boxed{[\rho\,\hat{u}]_{*R} = \rho_{*R}\hat{u}_*} \tag{4.18}$$

$$\boxed{[\rho\,\hat{v}]_{*L} = \frac{S_L\,[\rho\,\hat{v}]_L - [\rho\,\hat{u}\,\hat{v}]_L}{S_L - S_*}} \qquad \boxed{[\rho\,\hat{v}]_{*R} = \frac{S_R\,[\rho\,\hat{v}]_R - [\rho\,\hat{u}\,\hat{v}]_R}{S_R - S_*}} \tag{4.19}$$

$$\boxed{[\rho\,\hat{w}]_{*L} = \frac{S_L\,[\rho\,\hat{w}]_L - [\rho\,\hat{u}\,\hat{w}]_L}{S_L - S_*}} \qquad \boxed{[\rho\,\hat{w}]_{*R} = \frac{S_R\,[\rho\,\hat{w}]_R - [\rho\,\hat{u}\,\hat{w}]_R}{S_R - S_*}} \tag{4.20}$$

[†]The terms in square brackets [ ] indicate either conservative variables or convective flux variables, which are stored in corresponding vectors and can be used without additional computation.

The intermediate states are obtained by assembling the above terms as,

$$\mathbf{U}_{*L} = \begin{bmatrix} p/\beta \\ \rho\,\hat{u} \\ \rho\,\hat{v} \\ \rho\,\hat{w} \\ C \end{bmatrix}_{*L} \qquad \mathbf{U}_{*R} = \begin{bmatrix} p/\beta \\ \rho\,\hat{u} \\ \rho\,\hat{v} \\ \rho\,\hat{w} \\ C \end{bmatrix}_{*R} . \tag{4.21}$$

The convective flux in the rotated coordinate system, $\hat{\mathbf{F}}_f$, can then be computed using equations (4.5), (4.7) and (4.8). Please refer to Appendix A for a detailed derivation of HLLC-VOF Riemann solver.

4.1.2 Test problems

To test the efficacy and practical applicability of the HLLC-VOF convective flux solver, several two-dimensional problems are solved using structured and unstructured mesh. Also, a three-dimensional problem involving complex merging of interfaces is solved to demonstrate the extensibility and robustness of the solver. The results are compared with theoretical, experimental and numerical results available in literature. The problems are also solved using the HLL solver, which is a generic solver and hence used for comparison of results. The mathematical expressions for calculation of convective flux using HLL solver are included in Appendix C. In all the problems the artificial compressibility parameter, β, is chosen to be 1000. The velocity field of the flow is displayed in some figures, using arrows. The fluid interface is displayed by plotting two levels at 0.3 and 0.7, in all the two-dimensional contour plots. In three-dimensions, a contour level of 0.5 is used to display the fluid interface.

Low amplitude sloshing

This is a classic two-phase flow problem used for testing the convective flux and the effect of gravity on the two-phase fluid system. This problem consists of a tank of dimensions 0.1 m $\times$ 0.1 m having a denser fluid (water with density, $\rho_1 = 998\,\mathrm{kg/m^3}$) at the bottom and a lighter fluid (air with density $\rho_2 = 1.2\,\mathrm{kg/m^3}$) on the top. The interface is at a mean height of 0.05 m and displaced initially by a magnitude of 0.005 m. The initial interface location is given by equation,

$$y' = 0.05 + 0.005\,\cos\left(2\,\pi\,x/0.2\right) . \tag{4.22}$$

which results in an initial volume fraction of,

$$C = \begin{cases} 1 & y \le y' , \\ 0 & \text{otherwise} . \end{cases} \tag{4.23}$$

The pressure and velocity are initialized to zero. Under the influence of gravity the interface starts sloshing back and forth about the mean position. An important feature of this test problem is that the first mode of sloshing frequency of the free surface can be estimated analytically, for an inviscid fluid, using simplified potential flow theory [124]. The analytical time period of oscillation can be calculated to be [124, 125],

$$P = 2\pi \left[g\, k\, \tanh\left(k\, h\right)\right]^{-1/2} \approx 0.373723 \text{ s}. \tag{4.24}$$

where, $g = 9.81\,\text{m/s}^2$ is the downward acceleration due to gravity, $k = 2\pi/0.2\,\text{m}^{-1}$ is the wave number and $h = 0.05\,\text{m}$ is the mean height of the free interface. The analytical solution is available only for simplified inviscid fluid, hence in this problem, the viscosity is set to zero ($\mu_1 = \mu_2 = 0\,\text{Pa-s}$) and the surface tension is set to zero ($\sigma = 0\,\text{N/m}$). A boundary condition of slip-wall is imposed on all the walls of the tank. The simulation is carried out for $t = 2.5\,\text{s}$ with a time step of $\Delta t = 0.001\,\text{s}$. The problem is solved on Cartesian mesh with different cell sizes and unstructured mesh with triangular cells. A few snapshots of the simulation using the HLLC-VOF on a structured mesh of size 160×160 is shown in Fig. 4.2. The location of the interface at the left wall of the tank is plotted as a function of time in Fig. 4.3. It can be observed that the new solver results in superior results compared to HLL solver. Also, as seen in Fig. 4.3, the results obtained on the 80×80 mesh and the 160×160 mesh are very similar (almost overlapping) in case of HLLC-VOF solver, indicating that an accurate solution is obtained at a coarser mesh level itself. It may be noted that the alternate higher peaks, compared to theoretical first mode, seen in the numerical solution obtained by HLLC-VOF is due to the second and higher modes of oscillation and not due to error. This sloshing behavior is also observed in the results presented in other works [81, 125].

To demonstrate the capability of the solver on arbitrary mesh types, the simulation is also performed on unstructured mesh consisting of 23394 triangular cells, which is slightly less than the number of cells in 160×160 mesh. The unstructured mesh is generated using an open source code called "Triangle" [126], with an area constraint of $6.77 \times 10^{-7}\,\text{m}^2$ and angle constraint of 30°. The results obtained using the unstructured mesh are displayed in Fig. 4.4. It is not possible to perform a direct comparison of results obtained on structured and unstructured mesh. Nevertheless, the location of the water-air interface at the left wall of the tank is plotted in Fig. 4.5, along with the theoretical first mode of oscillation. It can be observed that the solution produced on the unstructured mesh is similar to the solution produced on the structured mesh, hence indicating that the solver is capable of handling unstructured mesh without a need for any modifications.

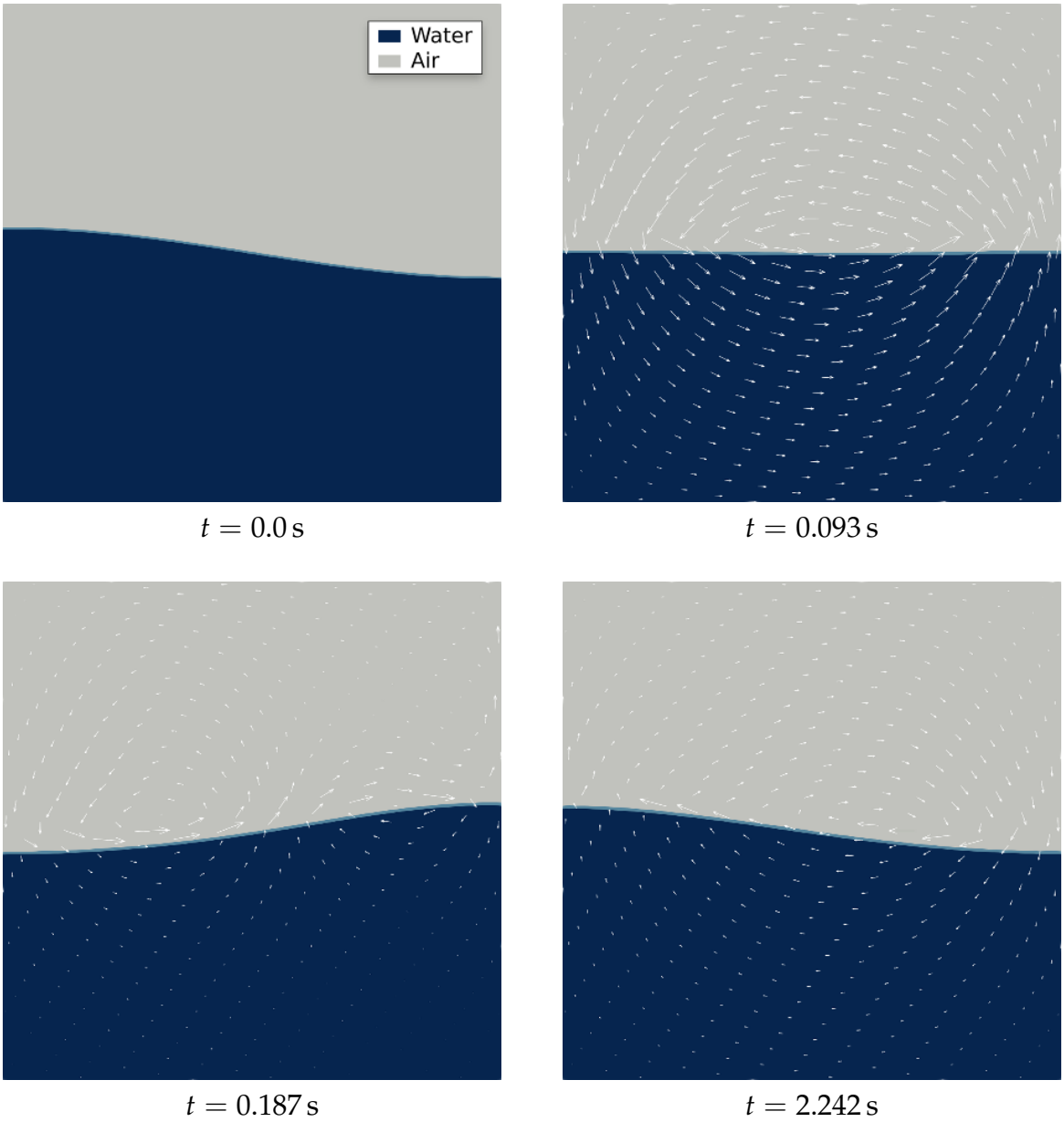


Fig. 4.2: A few snapshots of the low amplitude sloshing simulation using 160×160 Cartesian mesh at different time levels computed using HLLC-VOF Riemann solver.

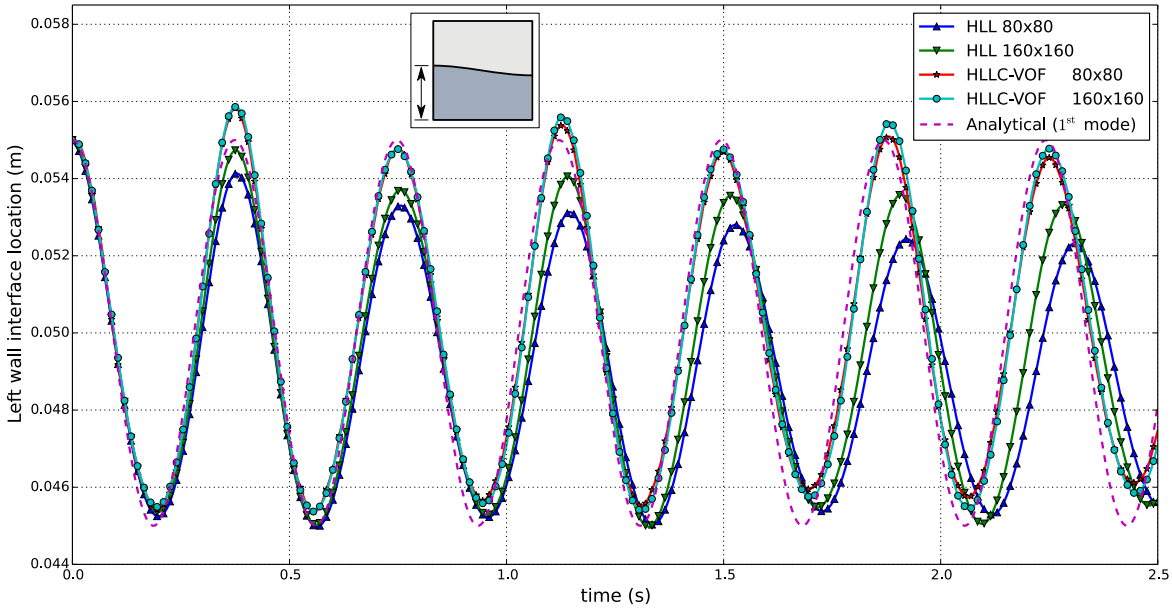


Fig. 4.3: Influence of the grid sizes and Riemann solvers on transient location of the water-air interface at the left wall of the tank.

Dam break

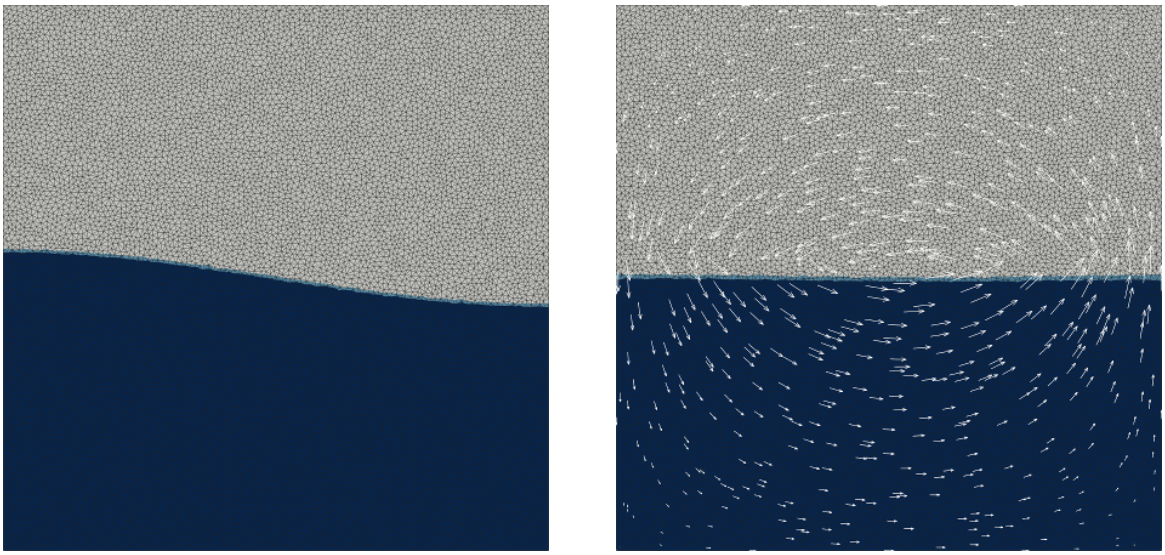
In this problem, a column of water is placed at the left-bottom corner of a tank containing air. The properties of water are taken as, $\rho_1 = 998\,\mathrm{kg/m^3}$, $\mu_1 = 1.002 \times 10^{-3}\,\mathrm{Pa\text{-}s}$ and the properties of air are taken as, $\rho_2 = 1.2\,\mathrm{kg/m^3}$, $\mu_2 = 1.825 \times 10^{-5}\,\mathrm{Pa\text{-}s}$. The surface tension coefficient is taken as $\sigma = 72.94 \times 10^{-3}\,\mathrm{N/m}$. The acceleration due to gravity is $g = (0, -9.81, 0)\ \mathrm{m/s^2}$. Under the influence of gravity the water column collapses and surges towards the right wall of the tank. The tank has a width of 2 m and a height of 1 m, with the initial water column having a width of $a = 0.45\,\mathrm{m}$ and a height of $b = 0.9\,\mathrm{m}$. An attractive feature of this test case is that experimental results are available [127, 128]. Therefore, in this test case, the complete governing equations (3.1) are solved, including convective, viscous, gravity and surface tension fluxes. In the available experimental results, the location of water-air interface at the bottom- and the left-wall is tracked and tabulated at various time intervals.

To solve the problem numerically, the domain is initialized by setting the volume fraction in the region of water as 1 and air as 0. The initial volume fraction is therefore given by,

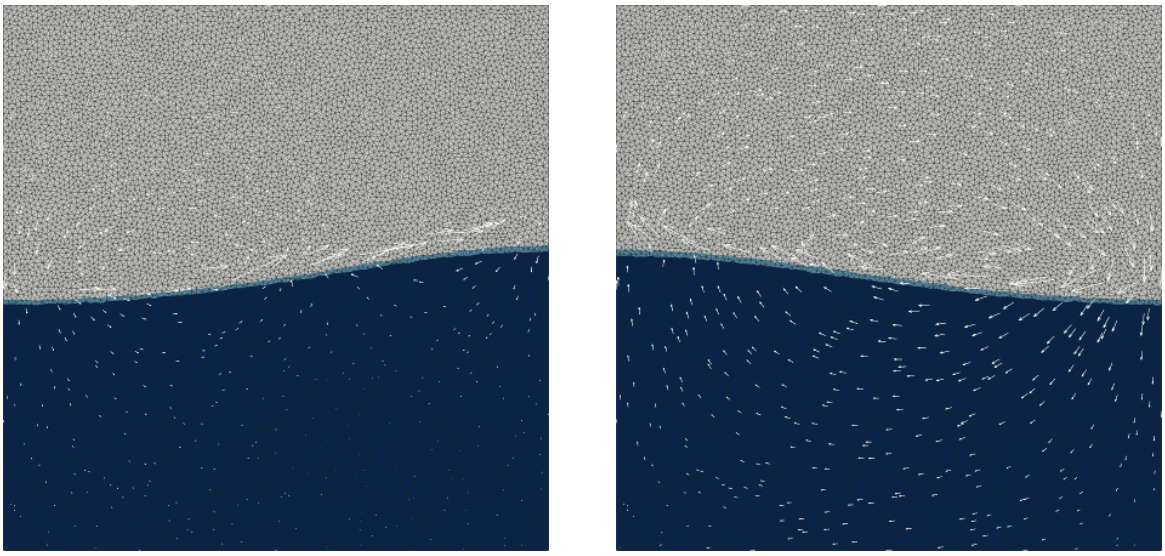
$$C = \begin{cases} 1 & x \leq 0.45 \text{ and } y \leq 0.9, \\ 0 & \text{othewise.} \end{cases} \tag{4.25}$$

The initial volume fraction is shown in Fig. 4.6. The initial pressure and velocity are set to zero. The boundaries of the tank are set as stationary no-slip walls.

The dam break problem is simulated, using a Cartesian mesh of size 160×80 cells and an unstructured mesh consisting of 11728 triangular cells, up to a time of $t = 2.5\,\mathrm{s}$ using time steps of $\Delta t = 0.001\,\mathrm{s}$. The unstructured mesh is created using Triangle code [126] with an area constraint of $2.7 \times 10^{-4}\,\mathrm{m}$ and an angle constraint of 30°. A few snap-



$t = 0.0\,\mathrm{s}$ $t = 0.093\,\mathrm{s}$



$t = 0.187\,\mathrm{s}$ $t = 2.242\,\mathrm{s}$

Fig. 4.4: A few snapshots of the low amplitude sloshing simulation using an unstructured mesh consisting of 23394 triangular cells at different time levels computed using HLLC-VOF Riemann solver.

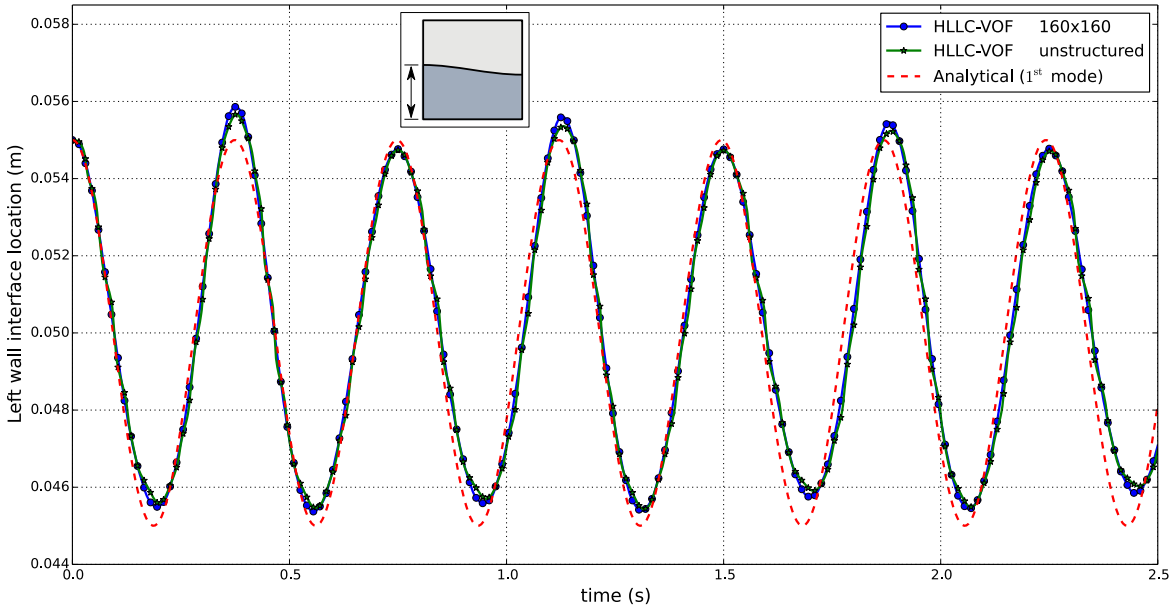


Fig. 4.5: Transient location of the water-air interface at the left wall of the tank obtained on structured and unstructured mesh computed using the HLLC-VOF Riemann solver.

shots of the numerical solution obtained using the new HLLC-VOF Riemann solver on structured mesh and unstructured mesh are shown in Fig. 4.7 and Fig. 4.8, respectively. These results are visually comparable with the experimental results available in literature [128]. The location of water front is denoted by l and the height of the water column at the left wall is denoted by h. The normalized location of the water-air interface at the bottom wall is plotted in Fig. 4.9, comparing the results of HLLC-VOF solver with that of HLL solver and experimental. The normalized location of the water-air interface at the left wall is plotted in Fig. 4.10, comparing the HLLC-VOF solver with HLL and experimental results. It can be observed in Fig. 4.9 and Fig. 4.10 that both the HLL and HLLC-VOF solvers capture the trend as seen in the experimental results during the short-time evolution of the flow. However during long-time evolution, due to extensive numerical dissipation of the non-contact capturing HLL solver, the water-air interface is not captured as sharply by the HLL solver as seen by comparing Fig. 4.7 and Fig. 4.11.

Rayleigh-Taylor instability

In this test problem, a heavier fluid is supported against gravity by a lighter fluid, hence the fluid system is unstable. The growth rate of instability, n, under the influence of gravity and surface tension can be obtained using linear theory [29] to be,

$$n^2 = k\,g\left[A - \frac{k^2\sigma}{g\,(\rho_1+\rho_2)}\right]. \tag{4.26}$$

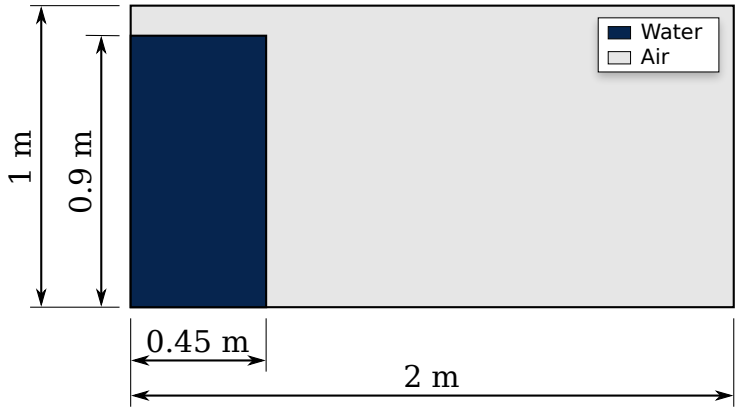


Fig. 4.6: Initial volume fraction for the dam break test problem.

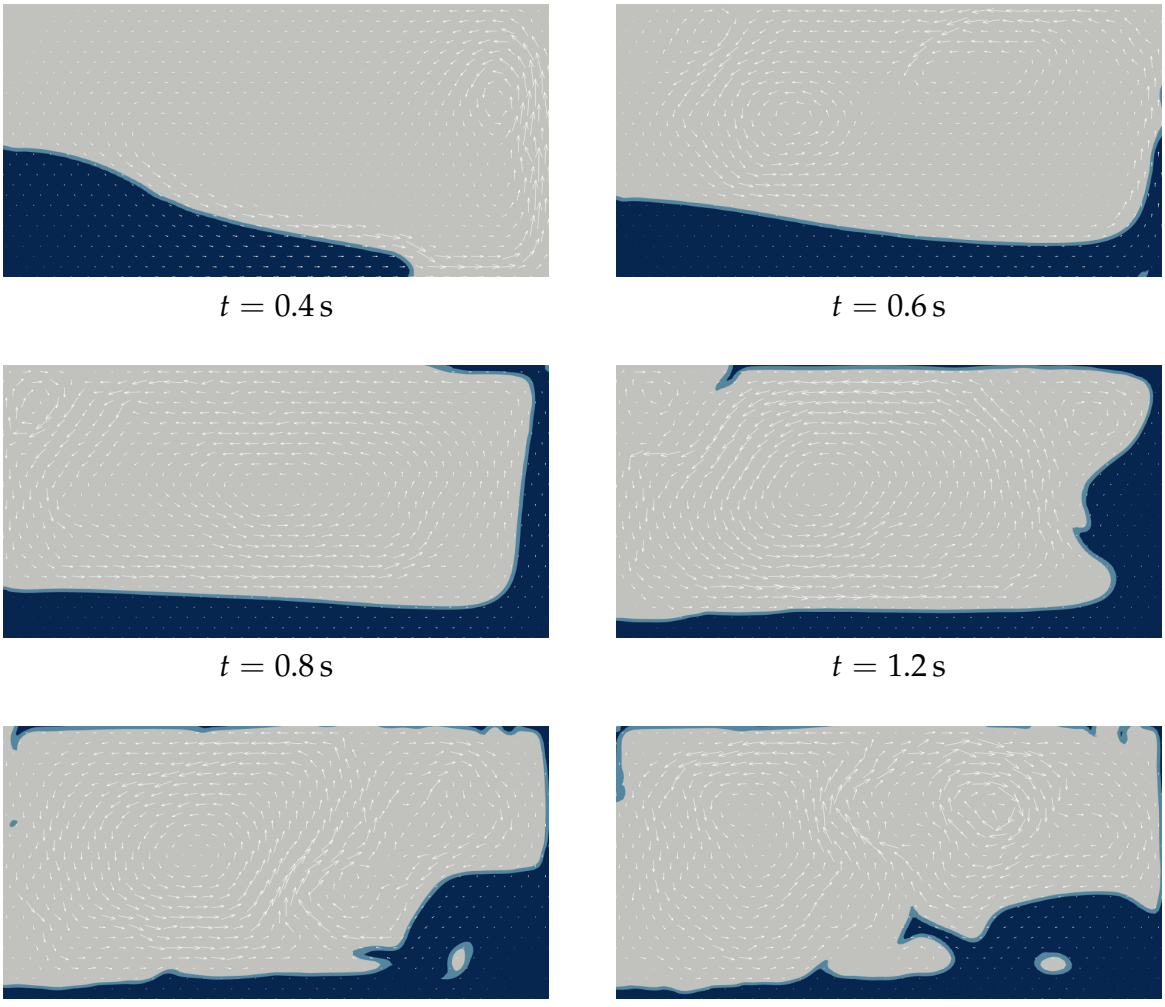


Fig. 4.7: A few snapshots of the dam break simulation using 160×80 Cartesian mesh at different time levels computed using HLLC-VOF Riemann solver.

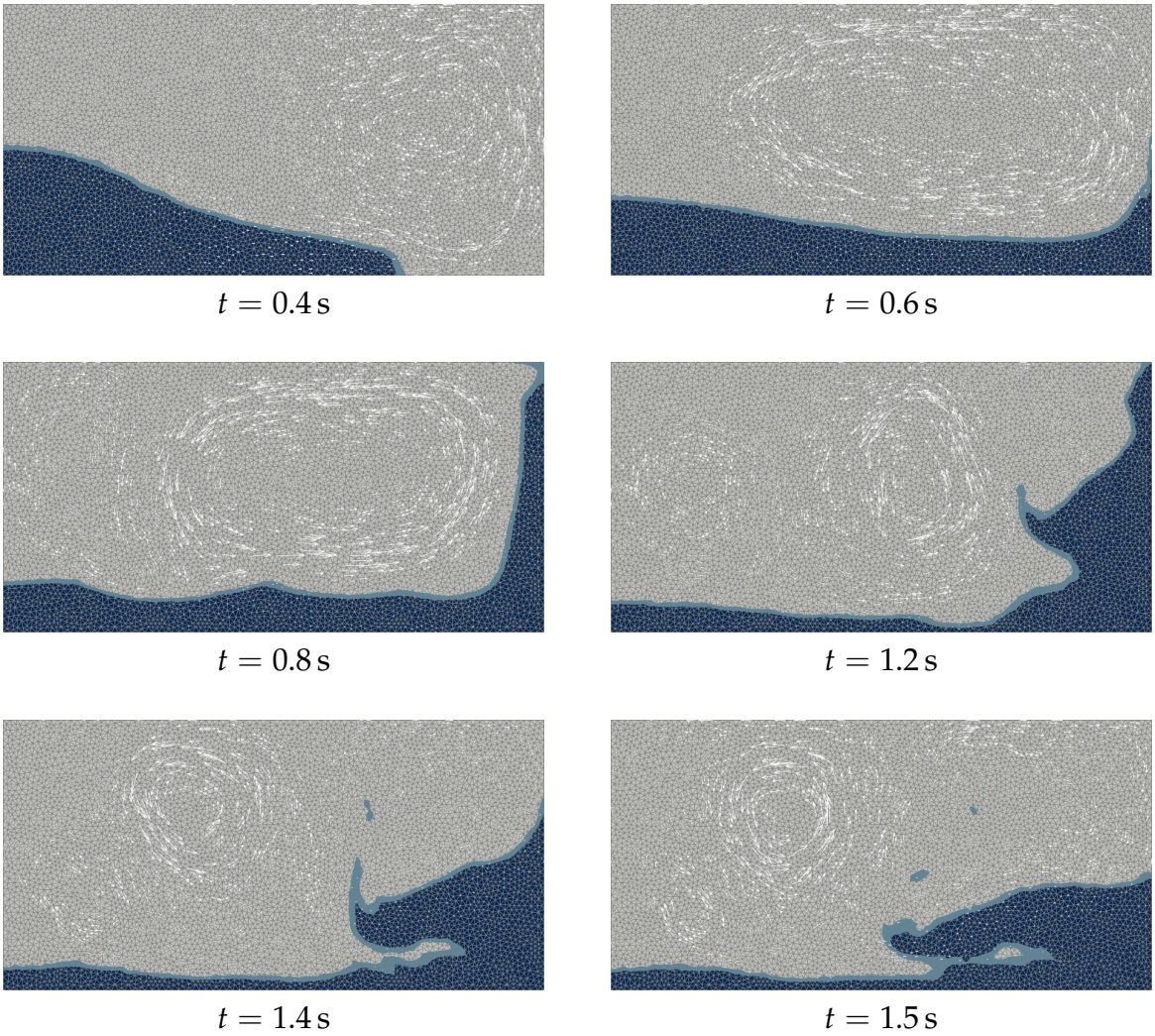


Fig. 4.8: A few snapshots of the dam break simulation using unstructured mesh consisting of 11728 triangular cells, at different time levels computed using HLLC-VOF Riemann solver.

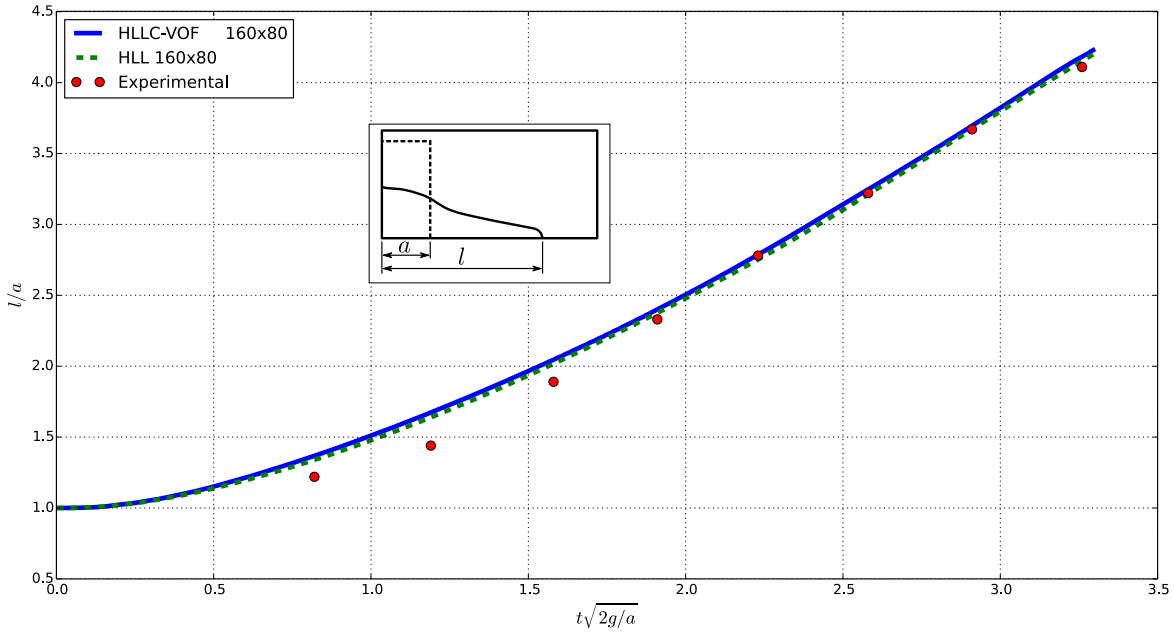


Fig. 4.9: Location of the water-air interface at the bottom wall in dam break problem.

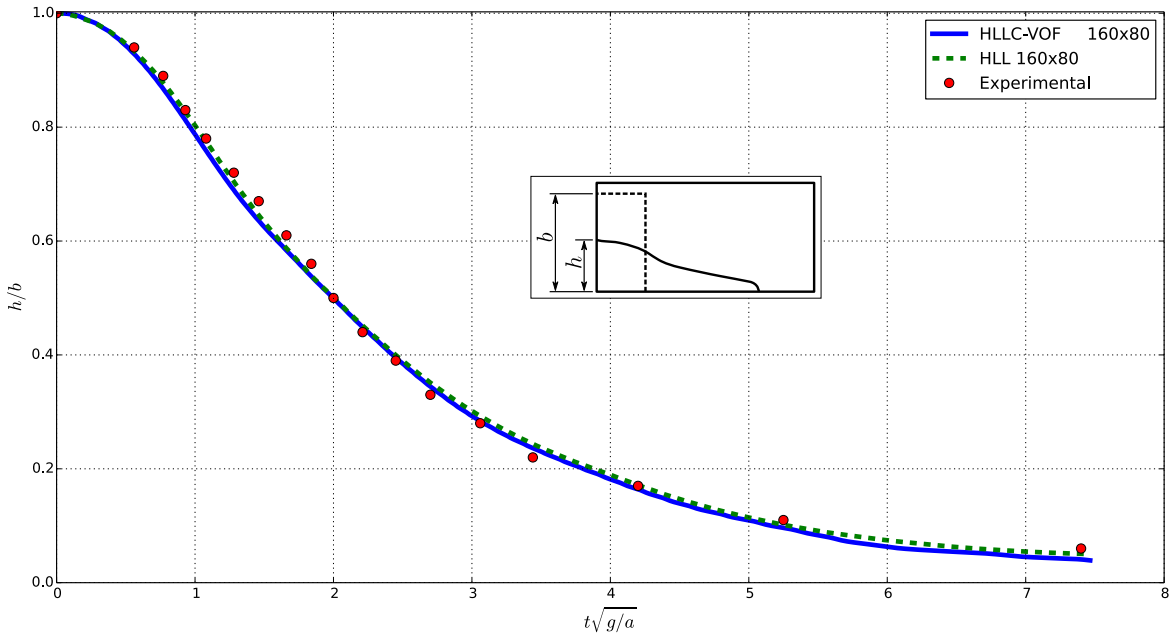


Fig. 4.10: Height of the water column at the left wall in dam break problem.

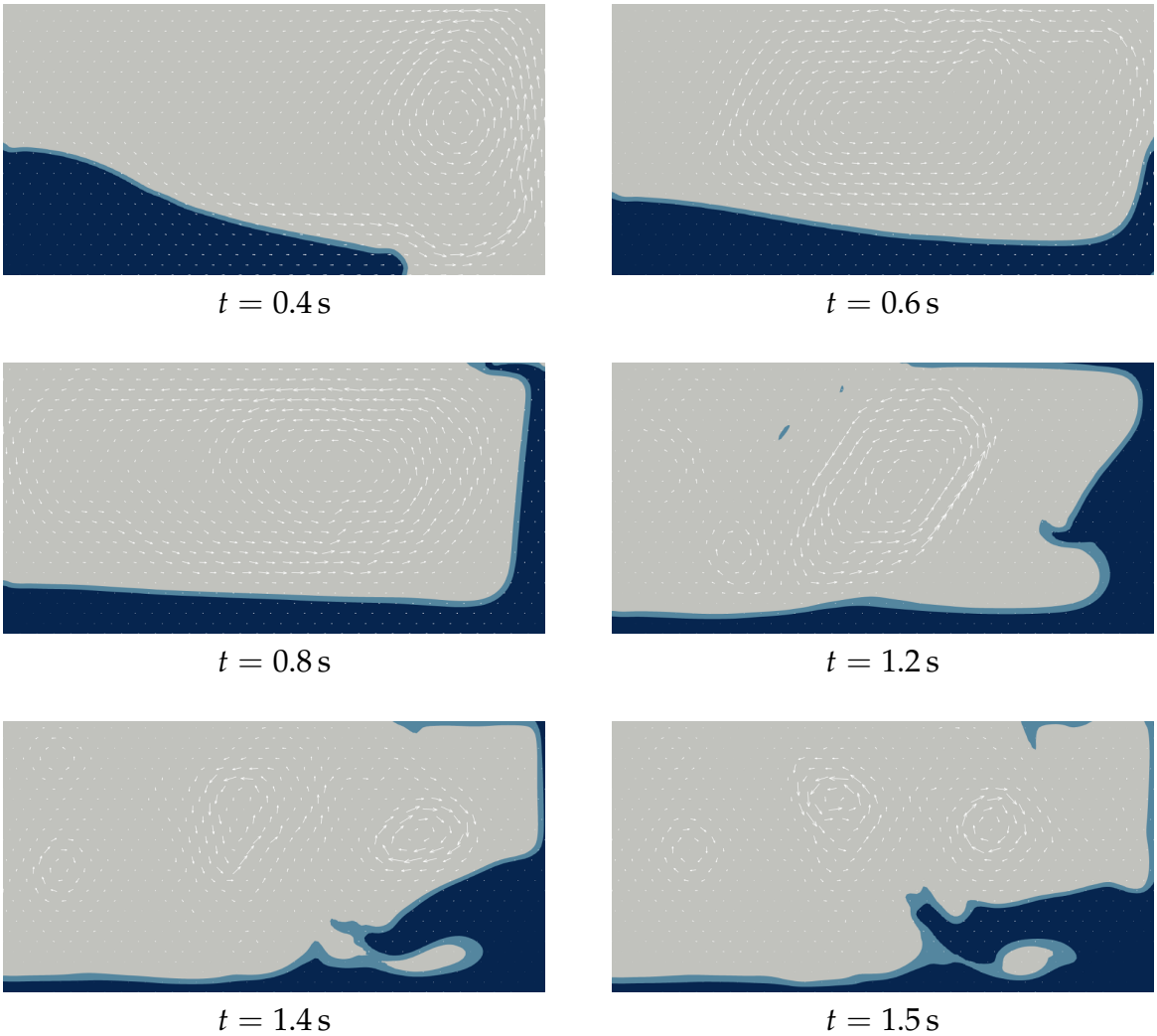


Fig. 4.11: A few snapshots of the dam break simulation using 160×80 Cartesian mesh at different time levels computed using HLL Riemann solver.

In the above equation, $A = (\rho_2 - \rho_1) / (\rho_1 + \rho_2)$ is called the Atwood number; $g = 1\,\mathrm{m/s^2}$ is the gravity; σ is the surface tension coefficient – value of which is decided by the stability parameter described later; ρ_1 is density of the lighter fluid and ρ_2 is density of the heavier fluid. The densities are chosen as, $\rho_1 = 0.3\,\mathrm{kg/m^3}$ and $\rho_2 = 1.2\,\mathrm{kg/m^3}$, which results in $A = 0.6$. The interface, separating the two fluids, is smoothly perturbed at the beginning so that the evolution of the instability is predictable. The initial perturbation of the interface is defined by a smooth cosine function,

$$y_0 = 0.035\ \cos(k\,x)\ , \tag{4.27}$$

where, k, is the wave number defined as $k = 2\pi/L$, and L is the wave length of perturbation. The wave number is chosen as $k = 5/3$. The perturbed initial interface location is defined as,

$$y' = y_{mean} + y_0\ . \tag{4.28}$$

The mean height of the interface, $y_{mean} = 2\,L$, and the total height of the domain is $3\,L$. As the problem is symmetrical about vertical plane, only half of the problem is solved by using symmetry boundary condition. Therefore, the width of the computational domain, is half wave length, $= L/2$. Based on the linear theory (which is only valid for short initial period), the perturbation grows exponentially [12, 29]. This can be mathematically expressed as,

$$y_c = y_0 \exp(n\,t)\ . \tag{4.29}$$

The initial vertical component of velocity at the interface, at $t = 0$, can be obtained by differentiating the above equation, which gives, $v_0 = n_0\,y_0$. The initial growth rate, n_0, can be obtained by using equation (4.26). The initial velocity field in the entire domain is defined as a smooth distribution, such that the velocity at the interface becomes v_0,

$$v = v_0\ \exp\left(-(k\,\Delta y)^2\right)\ , \tag{4.30}$$

where, Δy is the vertical distance from the interface, $\Delta y = y - y'$. The initial pressure field is defined using the hydrostatic pressure law, $dp = -\rho\,g\,dy$. The initial conditions are shown in Fig. 4.12. A symmetry boundary condition is applied at the left and right of the domain and a slip boundary condition is applied at the top and bottom.

The two fluids are considered to be inviscid, hence $\mu_1 = \mu_2 = 0$. In the absence of any other forces, the rate at which the instability grows is determined by the balance between the surface tension flux and the convective flux generated by gravity. A critical surface tension coefficient, σ_c, is defined as the one when the growth rate is zero. Therefore, from equation (4.26), the critical surface tension coefficient can be obtained as,

$$\sigma_c = \frac{A\,g\ (\rho_1 + \rho_2)}{k^2}\ . \tag{4.31}$$

The surface tension coefficient is defined using a stability parameter, ϕ, as $\sigma = \phi\,\sigma_c$.

The value of ϕ varies between 0 and 1, such that, when $\phi = 0$ surface tension is zero therefore the instability grows rapidly; and when $\phi = 1$ the surface tension is balanced by gravity and therefore the instability does not grow.

The Rayleigh-Taylor instability problem is numerically solved using the HLLC-VOF Riemann solver for various values of ϕ and compared with the results obtained using HLL solver and the growth rate predicted by linear theory. A Cartesian mesh of size 30×180 is used and the computation is performed using time steps of $\Delta t = 0.01\,\text{s}$ for a duration of $t = 10\,\text{s}$. The volume fraction at time $t = 10\,\text{s}$, for various values of ϕ, using HLLC-VOF solver, is plotted in Fig. 4.13. The growth rate of instability is given by the slope of displacement-versus-time on a semi-log plot, as shown in Fig. 4.14 (plotted only for few selected values of stability parameter to avoid clutter). In practice, the growth rate is estimated by fitting equation (4.29) to the interface displacement at symmetry line between time $t = 2\,\text{s}$ and $t = 4\,\text{s}$. The comparison of the growth rate, obtained using theoretical equation (4.26) and numerical computation, as a function of stability parameter is shown in Fig. 4.15. It can be observed that the non-contact preserving HLL Riemann solver is not capable of capturing the interface movement correctly, due to excessive smearing of interfaces, resulting from numerical diffusion. The HLLC-VOF Riemann solver produces much superior results in this case, as the smearing of the interface is well controlled by the contact preserving Riemann solver.

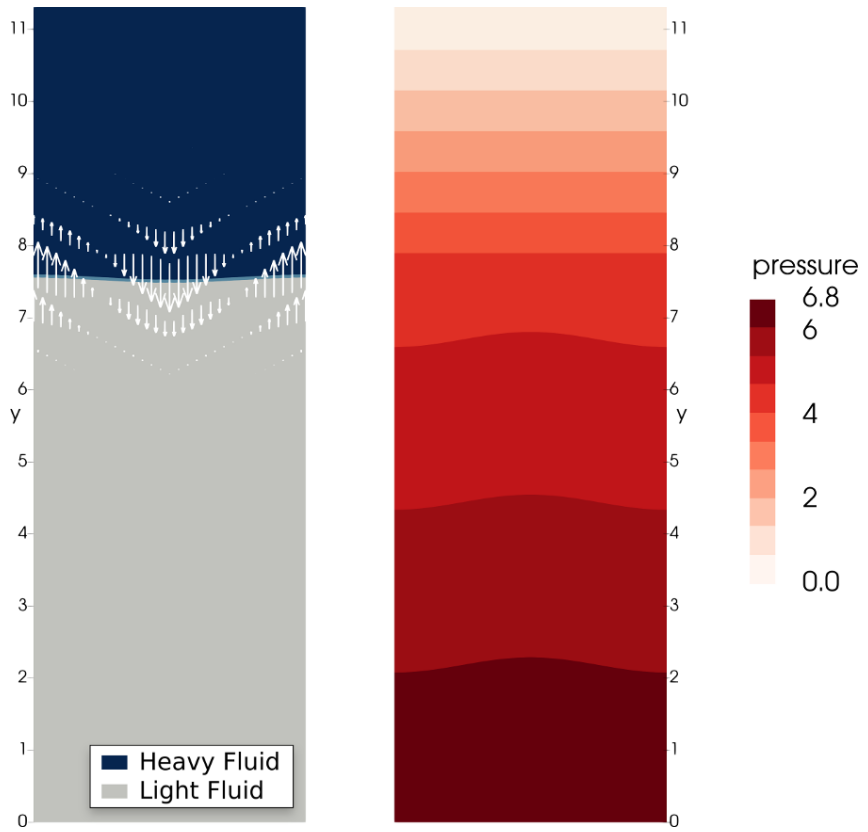


Fig. 4.12: Initial volume fraction, velocity field and pressure distrubution for Rayleigh-Taylor instability problem.

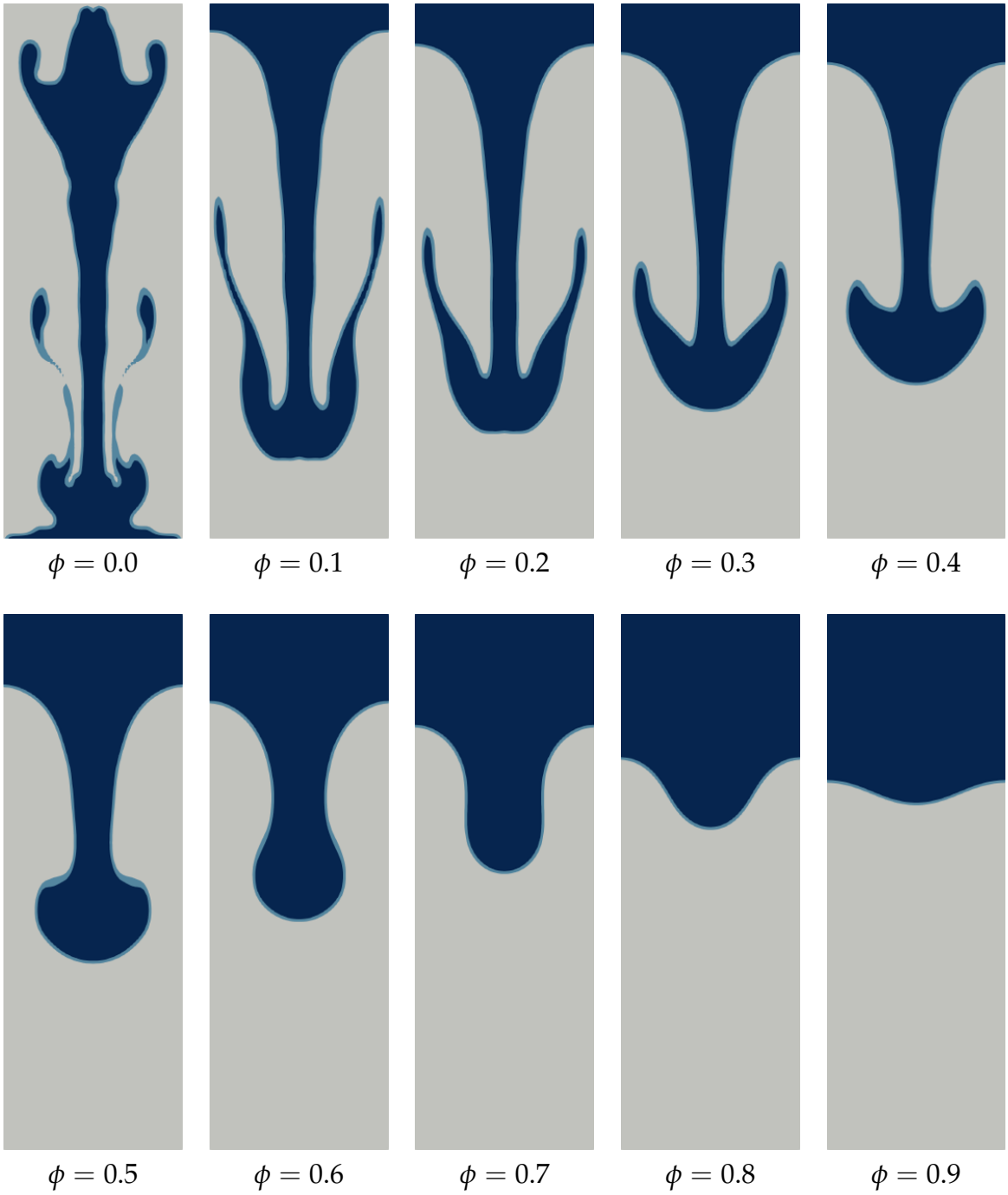


Fig. 4.13: Growth of Rayleigh-Taylor instability at time $t = 10\,\mathrm{s}$ with increasing values of stability parameter computed using HLLC-VOF solver.

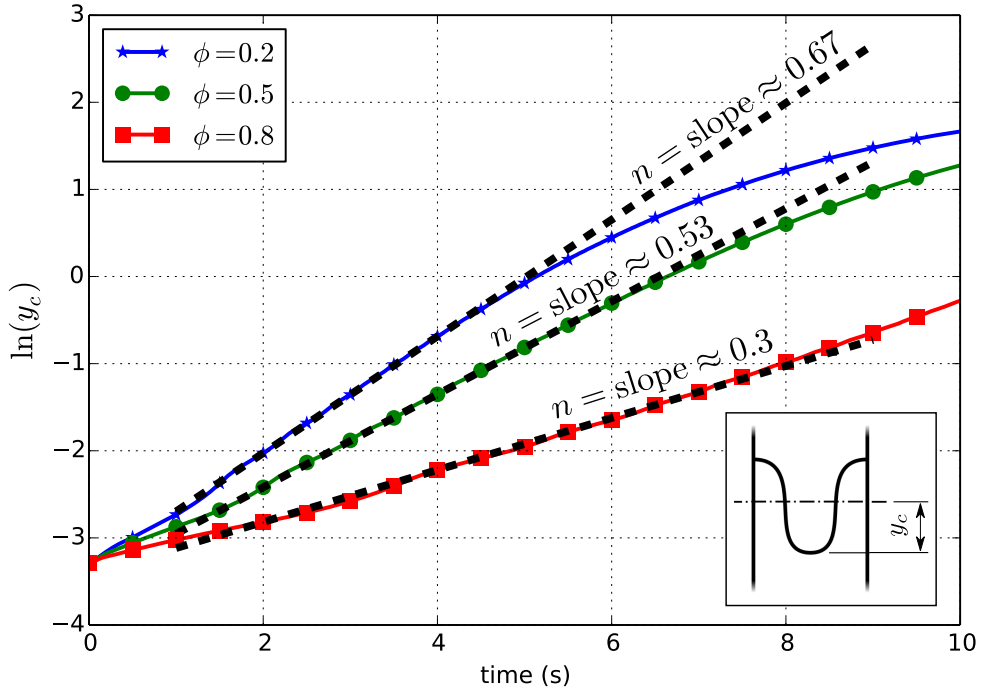


Fig. 4.14: Measurement of growth rate of Rayleigh-Taylor instability. Solutions are computed using HLLC-VOF solver for several values of ϕ. Here only a few values of ϕ are displayed to avoid clutter.

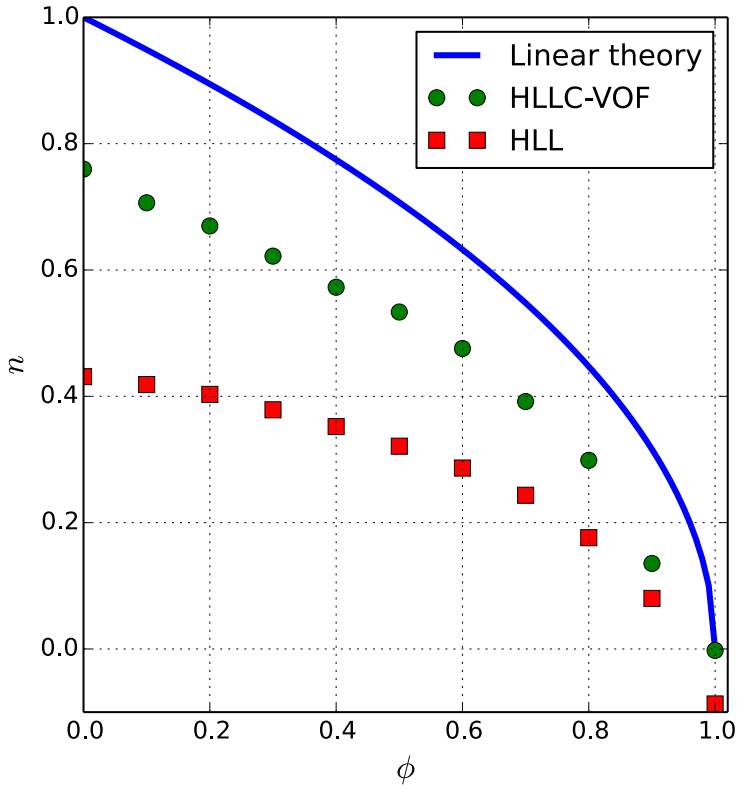


Fig. 4.15: Growth rate of Rayleigh-Taylor instability as a function of stability parameter.

Droplet splash

In this problem, a two-dimensional circular water droplet is placed over bulk water body separated by air. Numerical results of this problem are available in literature [129], where high order pressure-based, projection method is used. The effect of surface tension is not considered in the said paper, hence in this problem the surface tension coefficient is set to zero. The properties of the two fluids considered are water ($\rho_1 = 998\,\mathrm{kg/m^3}, \mu_1 = 1.002 \times 10^{-3}\,\mathrm{Pa\text{-}s}$) and air ($\rho_2 = 1.2\,\mathrm{kg/m^3}, \mu_2 = 1.825 \times 10^{-5}\,\mathrm{Pa\text{-}s}$). The acceleration due to gravity is $g = (0, -9.81, 0)\ \mathrm{m/s^2}$. The domain is of size $0.007\,\mathrm{m} \times 0.014\,\mathrm{m}$ with water filled up to $0.0088\,\mathrm{m}$. The water droplet is placed above the water surface with its center at $(0.0035\,\mathrm{m},\ 0.0105\,\mathrm{m})$ and having a diameter of $2.8 \times 10^{-3}\,\mathrm{m}$. The domain is discretized using a Cartesian mesh of size 80×160. The initial velocity and pressure in the entire domain is zero. The problem is solved up to a time of $t = 0.025\,\mathrm{s}$ using steps of $\Delta t = 10^{-5}\,\mathrm{s}$. The initial distribution of volume fraction along with the snapshots of the solution at various other time steps (same as [129]) using HLLC-VOF and HLL are shown in Fig. 4.16 and Fig. 4.17 respectively. It can be seen that the results obtained by the non-contact capturing HLL solver produces excessive numerical dissipation and the tiny air bubbles trapped inside the water are not captured by the solution. On the other hand, the results obtained using HLLC-VOF solver have very close resemblance with the results given in the literature [129].

Three-dimensional non-axisymmetric merging of bubbles

In this problem, two spherical bubbles are placed slightly offset to one another in a tank of size $1\,\mathrm{m} \times 2\,\mathrm{m} \times 1\,\mathrm{m}$. The diameter of the bubbles is $0.4\,\mathrm{m}$. The lower bubble is centered at $(0.6\,\mathrm{m}, 0.25\,\mathrm{m}, 0.6\,\mathrm{m})$; and the upper bubble is centered at $(0.4\,\mathrm{m}, 0.65\,\mathrm{m}, 0.4\,\mathrm{m})$, as measured from the bottom corner. Due to buoyancy the bubbles start moving up, and merge together as time evolves. A similar problem is solved numerically, in literature [15], using the advancing front method.

The bubbles and the outer fluid can be characterized by using the Eötvös number $Eo = \rho_1 g d_e^2/\sigma$, the Morton number $M = g\mu_1^4/(\rho_1 \sigma)$, and the ratio of fluid properties ρ_1/ρ_2 and μ_1/μ_2. Subscript 1 is used for heavier outer fluid and subscript 2 is used for the fluid contained inside the bubbles. The acceleration due to gravity in the downward direction is chosen as $g = 1\,\mathrm{m/s^2}$; the density of the outer fluid is chosen as $\rho_1 = 1\,\mathrm{kg/m^3}$; and the effective diameter of the bubbles is $d_e = 0.4\,\mathrm{m}$. The simulation is performed for $Eo = 50$, $M = 1$, $\rho_1/\rho_2 = 20$ and $\mu_1/\mu_2 = 26$. The remaining fluid properties can be easily evaluated using the above equations. The domain is discretized using a mesh of $32 \times 64 \times 32$ cells. The pressure and velocity are initialized to zero in the entire domain. All the boundaries of the domain are set as slip-walls. The simulation is carried out for a time duration of $t = 10\,\mathrm{s}$ with a time step

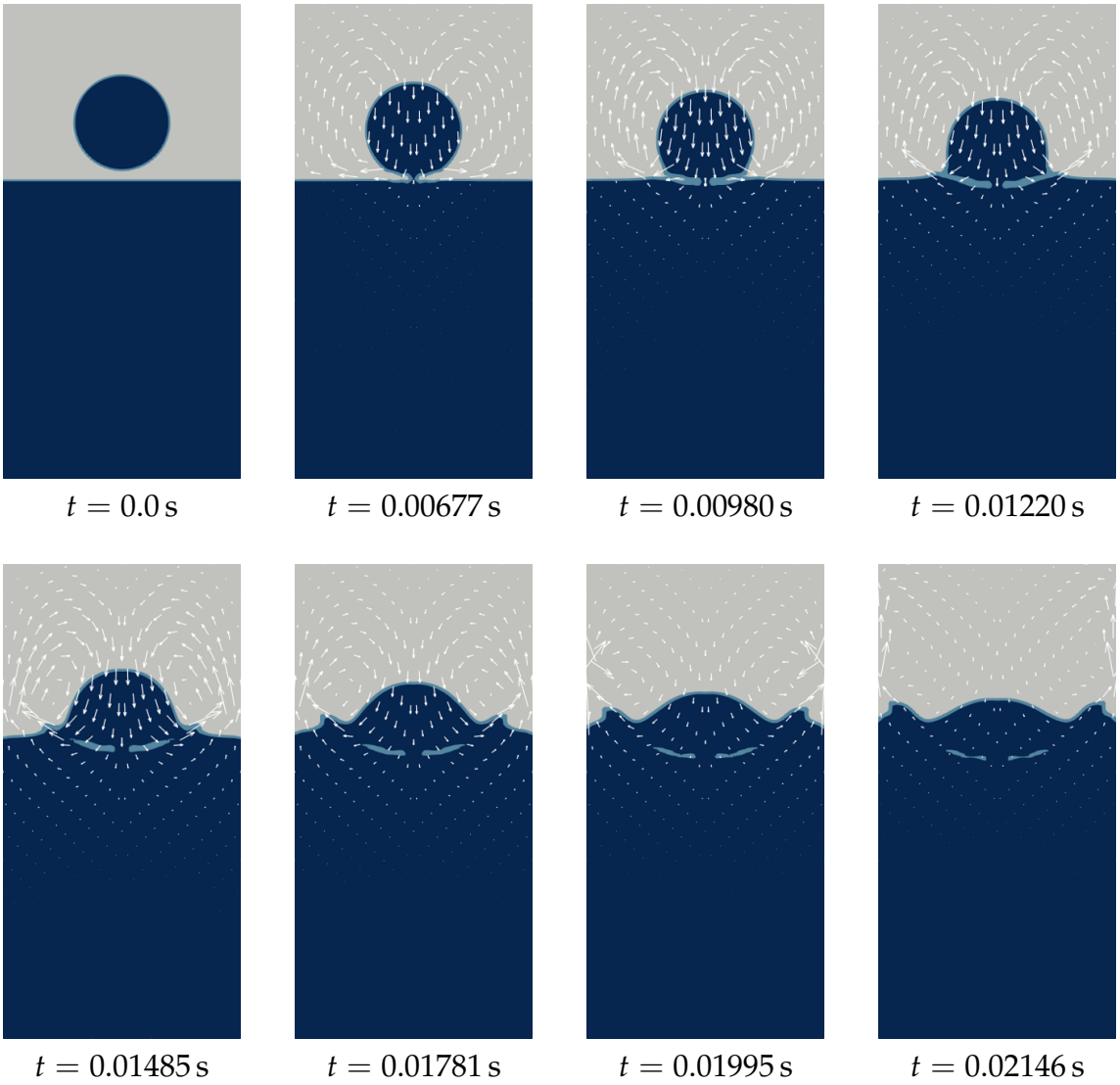
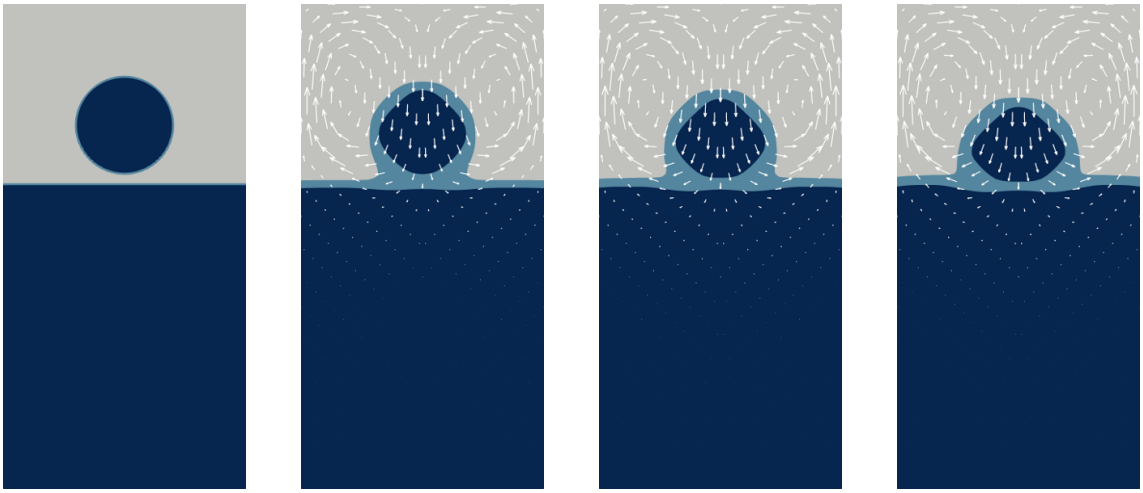


Fig. 4.16: Snapshots of the drop splash problem at various time levels computed using HLLC-VOF solver.



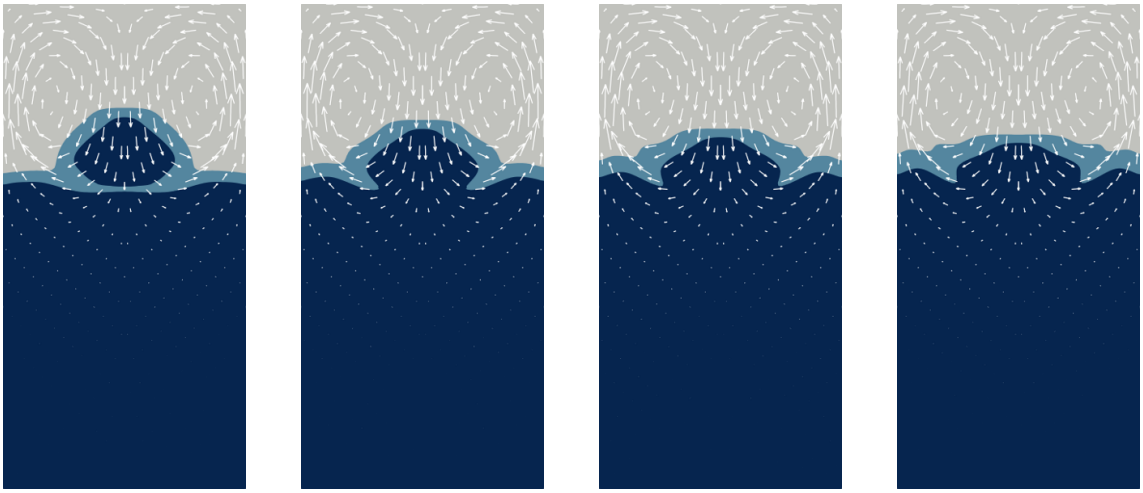


Fig. 4.17: Snapshots of the drop splash problem at various time levels computed using HLL solver.

of $\Delta t = 0.001\,\mathrm{s}$. Various snapshots of the HLLC-VOF solutions at different times of evolution are shown in Fig. 4.18. The evolution patterns are similar to the ones in the literature [15] for the same mesh size. Clearly the tail of the bubble is under resolved after a time of 6 s, which is also the case in the referred paper [15]. A finer mesh is necessary to compute the solution accurately for this problem over long time of evolution. Nevertheless, the capability of the HLLC-VOF solver in three-dimensions, including merging of complex interfaces, is established by solving this problem.

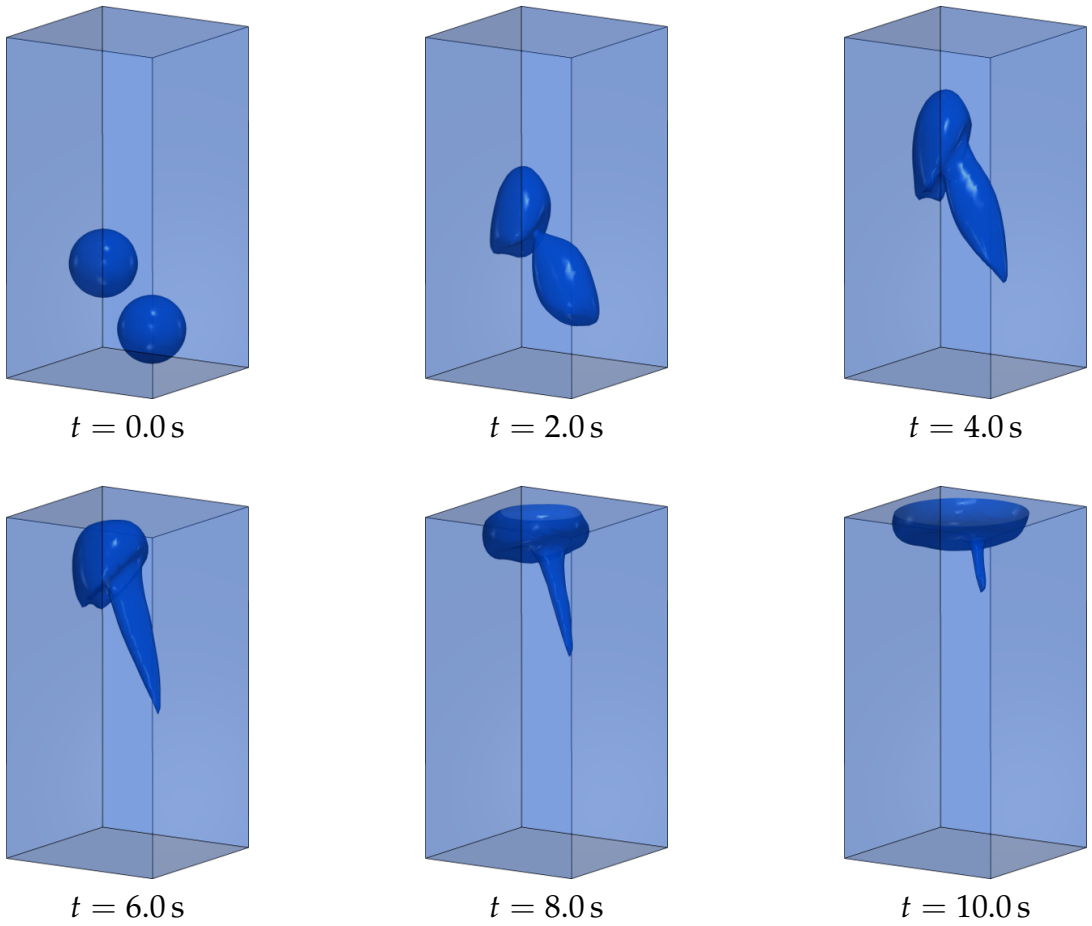


Fig. 4.18: Snapshots of two non-axisymmetric rising bubbles at different times computed using HLLC-VOF Riemann solver. Mesh = $32 \times 64 \times 32$ **cells,** $Eo = 50$, $M = 1$, $\rho_1/\rho_2 = 20$ **and** $\mu_1/\mu_2 = 26$.

4.1.3 Concluding remarks

An improved HLLC-type Riemann solver, called HLLC-VOF, is developed for three-dimensional, incompressible two-phase fluid flow. The convective flux is derived for the system of governing equations using the Rankine-Hugoniot jump conditions. The generalized Riemann invariant analysis is used to apply the necessary additional constraints. The convective flux computation using an upwind type solver inherently introduces numerical dissipation, which causes smearing of the fluid interface. Therefore, an interface compression mechanism, defined in section 3.8, is used to overcome

the unphysical dissipation effectively. The solver is generic and can be used to solve incompressible two-phase flow problems using structured and unstructured meshes in two- and three-dimensions. The new Riemann solver is found to be robust and performs well in all the selected test problems with a single chosen value of artificial compressibility parameter, β, unlike other formulations in the literature [81], where the value of β is determined based on trial-and-error. Various two- and three-dimensional test problems are solved on different mesh types to demonstrate the efficacy and robustness of the new solver.

4.2 Roe-type Riemann solver

The contact wave capturing ability of a Riemann solver is crucial for accurate flux estimation in incompressible two-phase fluid flows, as the fluid-interface behaves similar to a contact wave. This has already been demonstrated in the context of HLLC-type solvers when it is coupled with the volume of fluid (VOF) method in our paper [82] and preceding sections of this chapter. The Roe-type Riemann solver is another such method, where all the waves of the hyperbolic system are considered for flux calculation. Hence, a Roe-type solver has a potential of accurately calculating the flux in two-phase incompressible flows. However, unlike the HLLC-type solvers, the Roe-type solvers impose the requirement of linearity in its derivation. Therefore, it stands to be seen whether the Roe-type solver performs well in two-phase incompressible flow computation. In the present work, the Roe-type solver for VOF [45] is improved to work with unstructured meshes. The formulation is similar to the one developed for level-set method [130], which is extended here to three-dimensions. Additionally, an interface compression algorithm is used in VOF to ensure sharp fluid interfaces, hence eliminating the need for re-initialization which is necessary in case of the level-set method.

4.2.1 Formulation

In case of the Roe-type Riemann solver, the convective flux for a cell face can be written as,

$$\hat{\mathbf{F}}_f = \frac{1}{2}\left(\mathbf{F}\left(\hat{\mathbf{U}}_L\right) + \mathbf{F}\left(\hat{\mathbf{U}}_R\right)\right) - \frac{1}{2}\sum_{k=1}^{5} \tilde{\alpha}_k \left|\tilde{\lambda}_k\right| \tilde{\mathbf{K}}_k \,. \tag{4.32}$$

Here, tilde ($\tilde{}$) represents the Roe-averaged value of the variable. The left state (subscript L) and right state (subscript R) conservative variable vectors in the face-oriented

coordinate system can be written as,

$$\hat{\mathbf{U}}_{L,R} = \begin{bmatrix} p/\beta \\ \rho\ \left(u\, n_x + v\, n_y + w\, n_z\right) \\ \rho\ \left(u\, t_{1x} + v\, t_{1y} + w\, t_{1z}\right) \\ \rho\ \left(u\, t_{2x} + v\, t_{2y} + w\, t_{2z}\right) \\ C \end{bmatrix}_{L,R} = \begin{bmatrix} p/\beta \\ \rho\, \hat{u} \\ \rho\, \hat{v} \\ \rho\, \hat{w} \\ C \end{bmatrix}_{L,R} ,$$

The unit vectors $\hat{t}_1$ and $\hat{t}_2$ are tangent to the face. The Roe-averaged quantities are obtained as,

$$\tilde{\rho} = \sqrt{\rho_L\, \rho_R},\ \tilde{C} = \frac{C_L + C_R}{2} ,$$

$$\tilde{u} = \frac{\sqrt{\rho_L}\, \hat{u}_L + \sqrt{\rho_R}\, \hat{u}_R}{\sqrt{\rho_L} + \sqrt{\rho_R}},\ \tilde{v} = \frac{\sqrt{\rho_L}\, \hat{v}_L + \sqrt{\rho_R}\, \hat{v}_R}{\sqrt{\rho_L} + \sqrt{\rho_R}},\ \tilde{w} = \frac{\sqrt{\rho_L}\, \hat{w}_L + \sqrt{\rho_R}\, \hat{w}_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} ,$$

where, ρ_L and ρ_R are computed using equation (3.2) using C_L and C_R respectively. The Roe-averaged eigenvalues in equation (4.32) can be defined as,

$$\tilde{\lambda}_1 = \tilde{u},\ \tilde{\lambda}_2 = \tilde{u},\ \tilde{\lambda}_3 = \tilde{u},\ \tilde{\lambda}_4 = \tilde{u}_C + \sqrt{\tilde{u}_C^2 + \beta/\tilde{\rho}},\ \tilde{\lambda}_5 = \tilde{u}_C - \sqrt{\tilde{u}_C^2 + \beta/\tilde{\rho}} ,$$

where, $\tilde{u}_C = \tilde{u}\ \left(\rho_2 + \tilde{\rho}\right) / \left(2\, \tilde{\rho}\right)$. The Roe-averaged eigenvectors in equation (4.32) are computed as,

$$\tilde{\mathbf{K}}_1 = \begin{bmatrix} 0 \\ \left(\rho_1 - \rho_2\right)\ \tilde{u} \\ 0 \\ 0 \\ 1 \end{bmatrix} ,\ \tilde{\mathbf{K}}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} ,\ \tilde{\mathbf{K}}_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} ,$$

$$\tilde{\mathbf{K}}_4 = \begin{bmatrix} 1 \\ \frac{\tilde{\lambda}_4\left(\tilde{\lambda}_4\tilde{\rho} - \tilde{u}\rho_2\right)}{\tilde{\lambda}_4 - \tilde{u}} \\ \frac{\tilde{\lambda}_4\tilde{\rho}\tilde{v}}{\tilde{\lambda}_4 - \tilde{u}} \\ \frac{\tilde{\lambda}_4\tilde{\rho}\tilde{w}}{\tilde{\lambda}_4 - \tilde{u}} \\ \frac{\tilde{\lambda}_4\tilde{C}}{\tilde{\lambda}_4 - \tilde{u}} \end{bmatrix} ,\ \tilde{\mathbf{K}}_5 = \begin{bmatrix} 1 \\ \frac{\tilde{\lambda}_5\left(\tilde{\lambda}_5\tilde{\rho} - \tilde{u}\rho_2\right)}{\tilde{\lambda}_5 - \tilde{u}} \\ \frac{\tilde{\lambda}_5\tilde{\rho}\tilde{v}}{\tilde{\lambda}_5 - \tilde{u}} \\ \frac{\tilde{\lambda}_5\tilde{\rho}\tilde{w}}{\tilde{\lambda}_5 - \tilde{u}} \\ \frac{\tilde{\lambda}_5\tilde{C}}{\tilde{\lambda}_5 - \tilde{u}} \end{bmatrix} .$$

The Roe-averaged wave strengths are computed as,

$$\tilde{\alpha}_4 = \frac{1}{\tilde{\rho}\left(\tilde{\lambda}_4 - \tilde{\lambda}_5\right)} \left[\Delta\left(\rho\, \hat{u}\right) - \tilde{\rho}\, \Delta\left(p/\beta\right)\ \tilde{\lambda}_5 - \Delta C\, \tilde{u}\ \left(\rho_1 - \rho_2\right)\right] ,$$

$$\tilde{\alpha}_5 = \frac{1}{\tilde{\rho}\left(\tilde{\lambda}_5 - \tilde{\lambda}_4\right)} \left[\Delta\left(\rho\, \hat{u}\right) - \tilde{\rho}\, \Delta\left(p/\beta\right)\ \tilde{\lambda}_4 - \Delta C\, \tilde{u}\ \left(\rho_1 - \rho_2\right)\right] ,$$

$$\tilde{\alpha}_1 = \Delta C - \tilde{\alpha}_4 \left[\frac{\tilde{\lambda}_4\, \tilde{C}}{\tilde{\lambda}_4 - \tilde{u}}\right] - \tilde{\alpha}_5 \left[\frac{\tilde{\lambda}_5\, \tilde{C}}{\tilde{\lambda}_5 - \tilde{u}}\right] ,\quad \tilde{\alpha}_2 = \Delta\left(\rho\, \hat{v}\right) - \tilde{\alpha}_4 \left[\frac{\tilde{\lambda}_4\, \tilde{\rho}\tilde{v}}{\tilde{\lambda}_4 - \tilde{u}}\right] - \tilde{\alpha}_5 \left[\frac{\tilde{\lambda}_5\, \tilde{\rho}\tilde{v}}{\tilde{\lambda}_5 - \tilde{u}}\right] ,$$

$$\tilde{\alpha}_3 = \Delta\left(\rho\,\hat{w}\right) - \tilde{\alpha}_4\,\left[\frac{\tilde{\lambda}_4\,\tilde{\rho}\tilde{w}}{\tilde{\lambda}_4 - \tilde{u}}\right] - \tilde{\alpha}_5\,\left[\frac{\tilde{\lambda}_5\,\tilde{\rho}\tilde{w}}{\tilde{\lambda}_5 - \tilde{u}}\right]\,,$$

where, $\Delta\left(\cdot\right) = \left(\cdot\right)_R - \left(\cdot\right)_L$ is the jump across the cell face.

4.2.2 Test problems

To ascertain the capability of the Roe-type Riemann solver, two test problems are presented in this section.

Rayleigh-Taylor instability

This problem is identical to the one presented in section 4.1.2 for testing HLLC-VOF solver. A few snapshots of the solution at different values of stability parameter at time $t = 10\,\mathrm{s}$ are shown in Fig. 4.19. The numerical growth rate is compared with the theoretical growth rate based on linear theory [29] in Fig. 4.20.

Three-dimensional non-axisymmetric merging bubbles

This problem is identical to the one presented in section 4.1.2 for testing HLLC-VOF solver. A few snapshots of the solution are displayed in Fig. 4.21, at different time levels. This problem is solved using the front tracking method in literature [15] which compares well with our results.

4.2.3 Concluding remarks

A three-dimensional Roe-type Riemann solver, combined with generic interface compression algorithm, is defined for incompressible two-phase flows. A two-dimensional Rayleigh-Taylor problem is solved with different values of surface tension. The results of Roe-type solver are compared with the HLLC-VOF solver and linearized theory. It is observed that the Roe-type solver produces equally good results as the HLLC-VOF solver. Also, a three-dimensional problem involving non-axisymmetric merging of bubbles is solved using the Roe-type solver. It is found that the results are visually identical to the HLLC-VOF solver. It can be concluded from the results that preserving the intermediate contact wave is very important for producing accurate convective flux in incompressible two-phase flows. This conclusion is based on the observation that the newly developed HLLC-VOF and the Roe-type solvers, where the contact wave is preserved, produce superior results compared to the HLL Riemann solver, where the contact wave is not preserved. Thus, both the HLLC-VOF and the Roe-type Riemann solvers can be used for practical problems, however the interface compression algorithm, defined in section 3.8, needs to be used to maintain sharpness of interfaces.

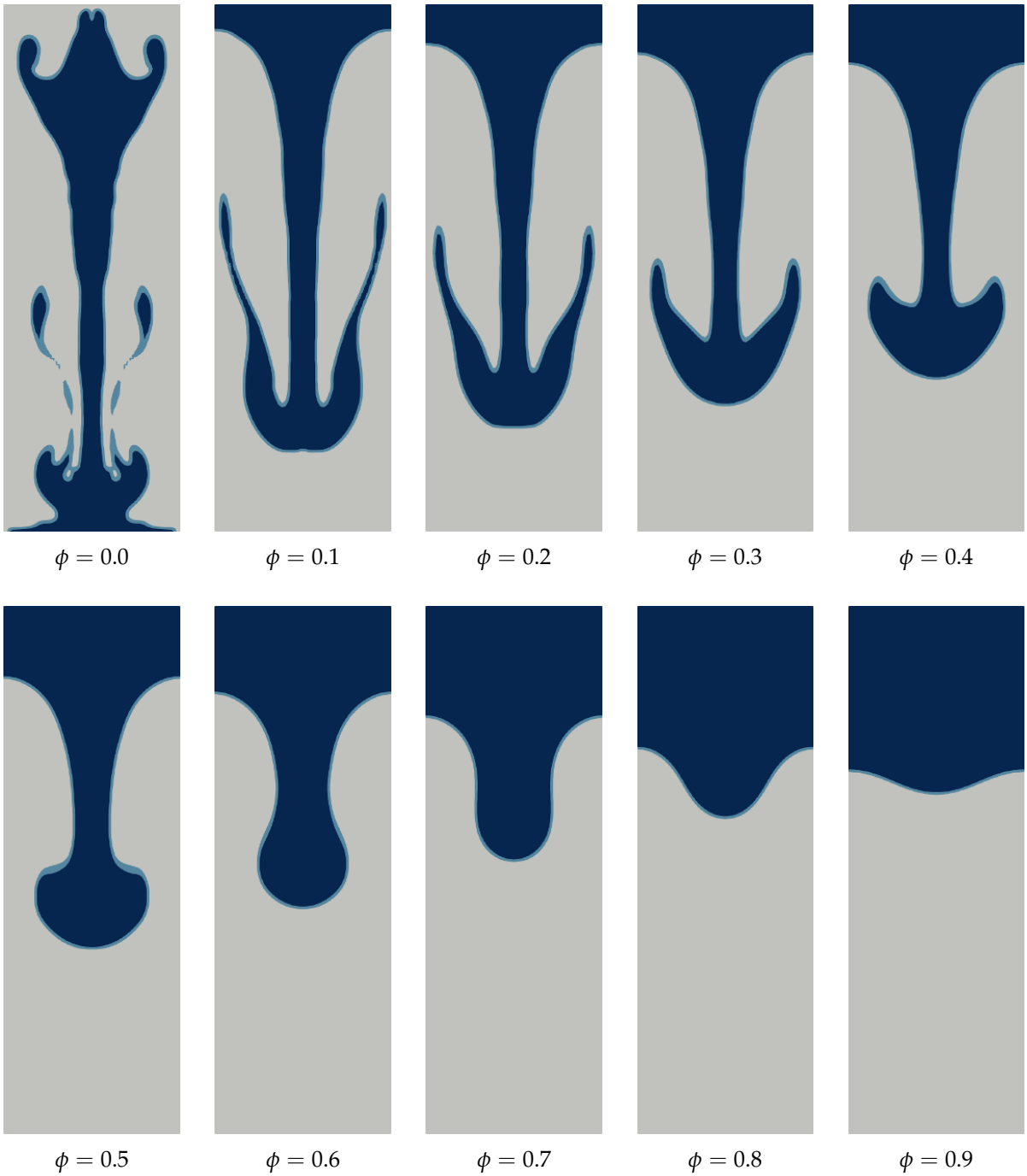


Fig. 4.19: Growth of Rayleigh-Taylor instability at time $t = 10\,\mathrm{s}$ with increasing values of stability parameter computed using Roe-type Riemann solver.

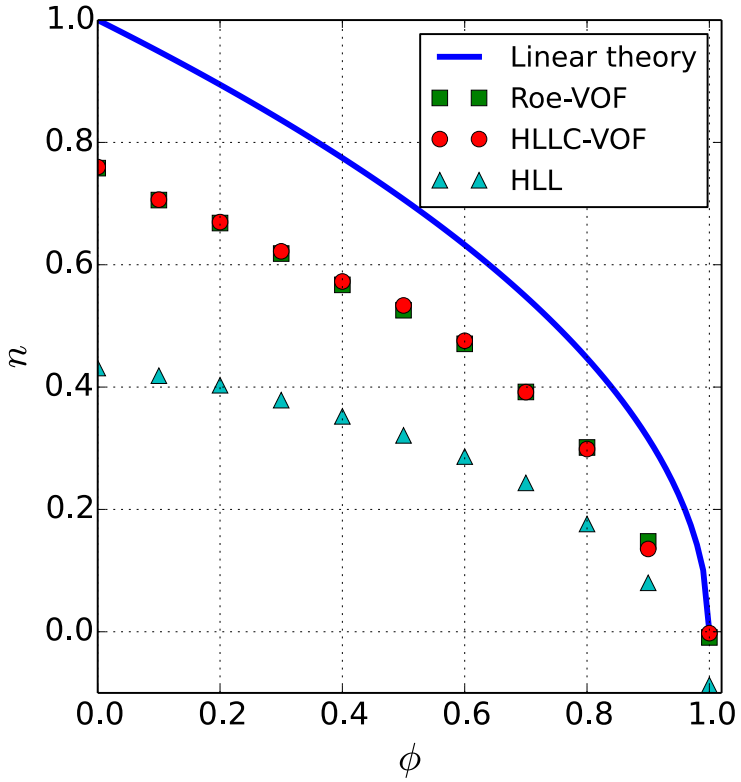


Fig. 4.20: Growth rate of Rayleigh-Taylor instability as a function of stability parameter computed using Roe-type Riemann solver.

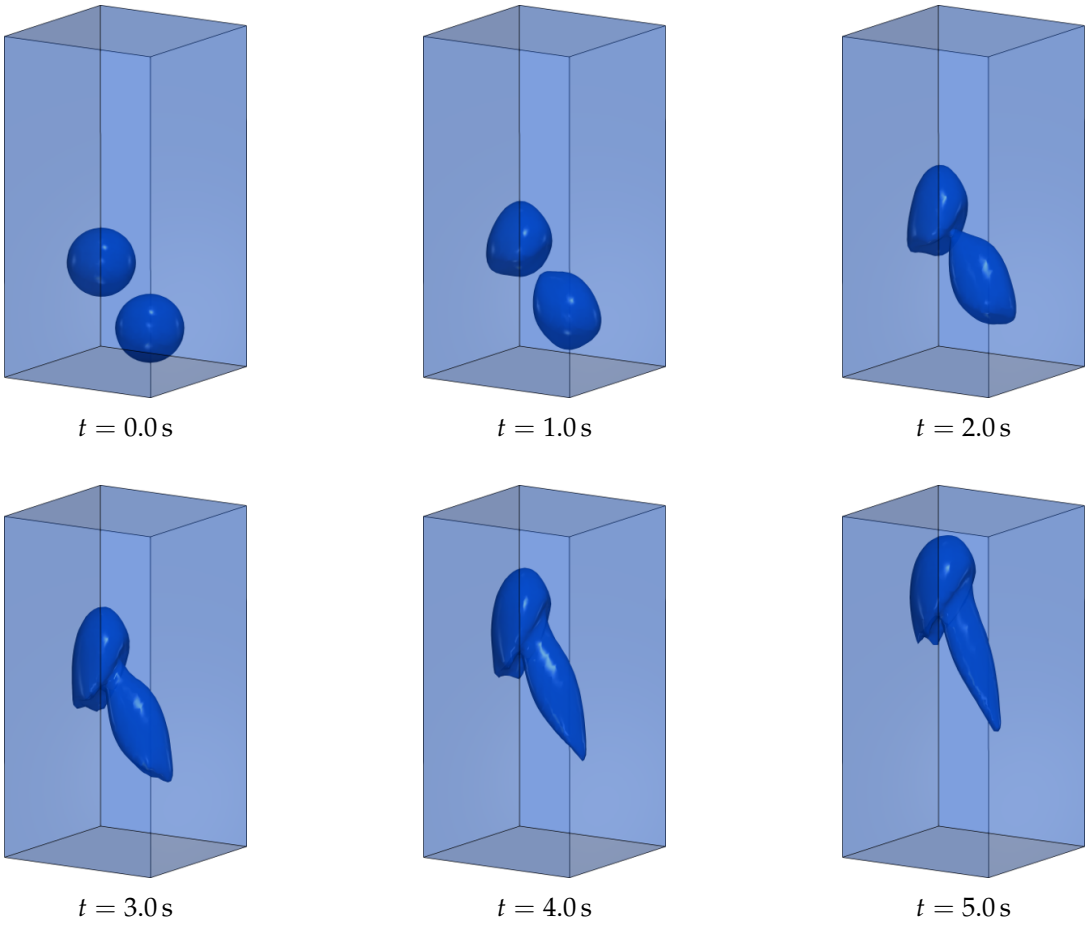
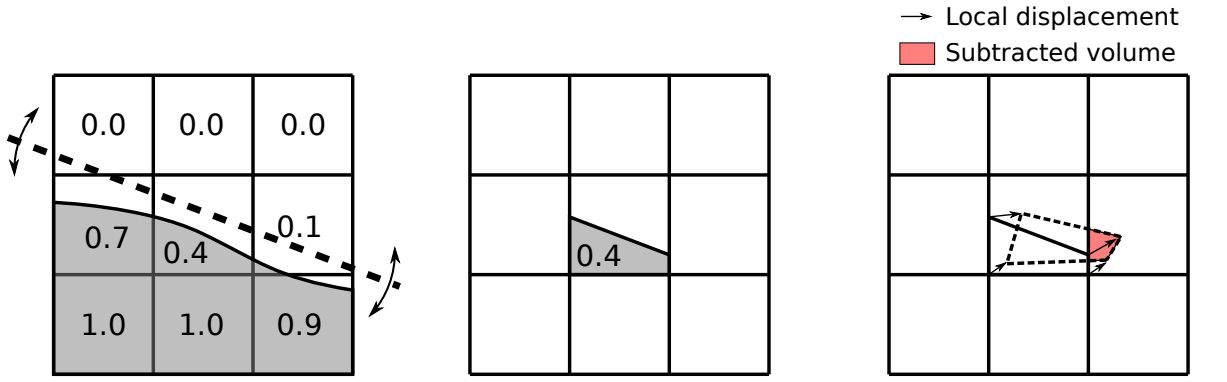


Fig. 4.21: Snapshots of non-axisymmetric merging bubbles at different times. Mesh=$32 \times 64 \times 32$ **cells,** $Eo = 50$, $M = 1$, $\rho_1/\rho_2 = 20$ **and** $\mu_1/\mu_2 = 26$ **computed using Roe-type Riemann solver.**

Chapter 5

Novel Volume Fraction Reconstruction and Flux Computation

The piecewise linear interface calculation (PLIC) is a very widely used method for reconstruction of an accurate fluid interface. In the PLIC method a sharp interface is reconstructed for each finite volume cell, by using the local volume fraction data. In each cell the interface is represented by a straight line dividing the cell into two parts – a filled region and an unfilled region. The interface reconstruction consists of two steps. In the first step the interface normal is computed, and in the second step the location of the interface within the cell is computed, such that the volume fraction of the cell is satisfied. The two steps of PLIC method are illustrated in Fig. 5.1(a) and Fig. 5.1(b). Once the interface is reconstructed, the volume fraction of the cell is updated, i.e. advanced in time, by deforming the filled volume geometry depending on the local velocity field. The volume moving out is subtracted from the cell and added to the neighbors to obtain the updated volume fraction of the cell. The updation procedure is illustrated in Fig. 5.1(c). The details of PLIC method can be found in [23, 129].



(a) interface orientation (b) interface location (c) volume fraction updation

Fig. 5.1: Steps of the PLIC method.

Even though the PLIC method is a widely used, it suffers from a few shortcomings. The first shortcoming is that the computation of the location of the interface (step two

of interface reconstruction), is limited to rectangular and cuboid shaped cells. However, this limitation has been recently overcome and the reconstruction procedure has been generalized to convex cell shapes [20, 21]. Another major limitation of the PLIC method is that the flux computation is closely tied to the shape of the cell, and therefore the method is mostly restricted to simple Cartesian mesh. A few attempts to overcome this limitation have appeared in the literature [24, 25], for triangular and tetrahedral cells. However, these methods tend to be very complicated, as they involve geometry slicing and tracking of individual pieces of the sliced geometry, for flux computation.

Another method for capturing sharp interfaces, which is also more promising, is the interface compression method [43, 49]. In this method simple advection algorithms are utilized for solving the volume fraction advection equation. The spread of interface due to numerical dissipation is kept in check by artificially modifying the flux [35, 43] or by adding a compressible velocity field [49] which acts locally near the interface, and is designed to maintain a sharp interface. These methods are generic and therefore can be easily extended to unstructured meshes. Hence, the methods have been widely used in commercial and open-source CFD codes. The new method developed in this thesis is also generic and easy to extend to different mesh types. Therefore, the newly developed method is compared with interface compression method, on unstructured meshes.

In this research work, a novel method is developed which simplifies the process of interface reconstruction and flux computation for any complex shaped cell. This work was originally published in our paper [131]. The new method of interface reconstruction is inspired from the marching cubes method [132], which is a commonly used method in visualization softwares. To solve the second part of the problem, i.e. computation of volume fraction flux, a simple upwind Riemann solver is used. The process of flux calculation using a Riemann solver is much simpler compared to geometric manipulation, and also extends very well to arbitrary shaped cells. The algorithm for interface reconstruction and flux computation is described in the following sections.

5.1 Reconstruction of interface

The interface reconstruction is inspired from the marching cubes method [132] which is a widely used method for computation of contour surfaces or iso-function surfaces of a discrete scalar field. Even though the method is called “marching cubes”, the method easily generalizes to cells of any shape. Henceforth, a two-dimensional interface reconstruction is discussed for simplicity, however the method easily extends to three-dimensions. The method of interface line reconstruction is as follows. The first step is to compute the volume fraction values at nodes, which may be done by interpolating the cell data using the distance weighted least-square method. Let the nodal value be denoted by C_n. To obtain the interface line corresponding to any chosen volume fraction level, C_k, the nodes are marked as zero or one. A node is marked as 0

if $C_n < C_k$ and marked as 1 otherwise. This leads to 2^M possible configurations of the linear interface inside the cell, where M is the number of nodes of the cell. All the possible configurations for a quadrilateral cell are shown in Fig. 5.2. In the figure, the filled circle represents a node marked as 1 and unfilled circle represents a node marked as 0.

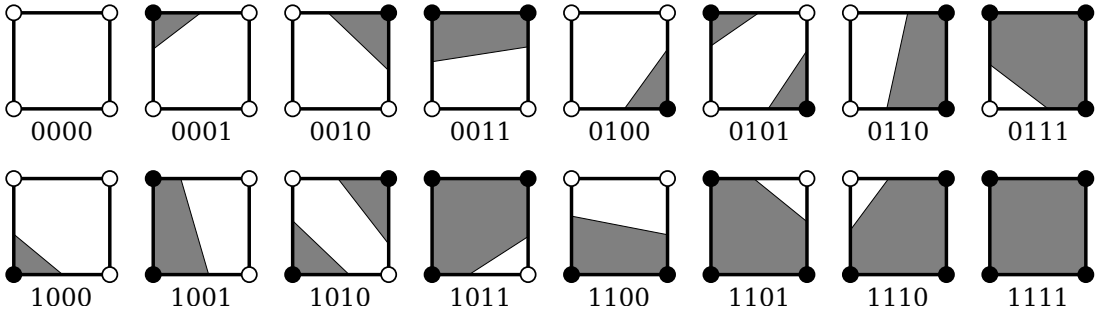


Fig. 5.2: Possible configurations of an interface for a quadrilateral cell.

The corner points of the filled polygon on the faces can be computed using linear interpolation, as the values at the nodes, C_n, and the value at the interface, C_k, are known. Once the interface configuration for the cell is identified from the possible list of configurations (Fig. 5.2), the filled area, A_k, in the cell can be exactly computed using simple formula for area calculation for the filled polygon. Once the filled area A_k is computed, the volume fraction of the cell can be computed as $C_{ik} = A_k / A_i$, where A_i is the total area of the cell.

Any chosen value of C_k will result in a unique configuration of the interface and therefore a unique volume fraction, $0 \leq C_{ik} \leq 1$ for the cell. The function mapping a volume fraction level, C_k, to the cell volume fraction, C_{ik}, can be written as,

$$f\left(C_k\right) = C_{ik}\,.$$

The reconstructed interface can be obtained by finding the root of the equation,

$$f\left(C_k\right) - C_i = 0\,, \tag{5.1}$$

where, C_i is the cell averaged volume fraction value of the cell, which is available in the finite volume formulation. The root of this equation can be obtained very quickly by using the secant method, starting with initial values of $\left[C_n^{\min}, C_n^{\max}\right]$, where, $C_n^{\min}$ and $C_n^{\max}$ are the minimum and maximum volume fraction values at nodes of the cell respectively. In other words, if the computed C_{ik} is not same as C_i, then iteration is needed till they match.

The left state values required for calculation of the flux using a Riemann solver can be obtained from the reconstructed interface. The left state values, C_L, are computed by using the proportion of the face covered by the fluid. An example of calculation of C_L for the faces of a quadrilateral cell is shown in Fig. 5.3 for the configuration "0110" (see Fig. 5.2). The right state, C_R, at the face is simply the left state, C_L, obtained by the

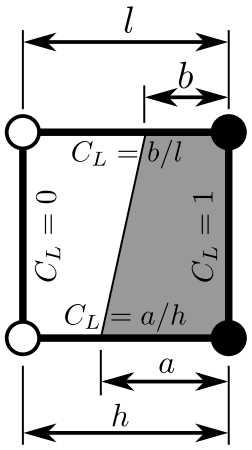


Fig. 5.3: An example showing calculation of left state, C_L, of Riemann problem for the faces of a quadrilateral cell.

other neighbor of the face.

The interface reconstruction procedure described above is generic and can be applied to any arbitrary shaped cell. This procedure can be simplified further for triangular cells by utilizing the symmetries of triangles and transforming the triangle to natural coordinate system (see Appendix D for details). To reconstruct the interface for a triangular cell consider the schematic shown in Fig. 5.4. In the figure, $C_n^{\min}$, $C_n^{\max}$ and C_n^{mid} are the minimum, maximum and intermediate value of volume fraction at cell nodes, where node positions are arbitrary. The normalized value of volume fraction, C', is defined as,

$$C' = \frac{C - C_n^{\min}}{C_n^{\max} - C_n^{\min}} \implies C'_{\min} = 0\,; \quad C'_{\max} = 1\,; \quad C'_{\text{mid}} = \frac{C_n^{\text{mid}} - C_n^{\min}}{C_n^{\max} - C_n^{\min}}\,.$$

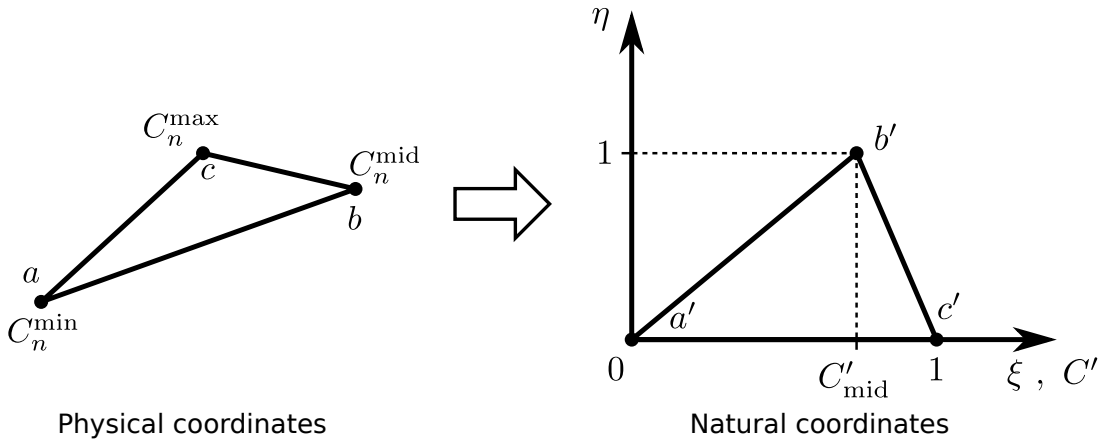
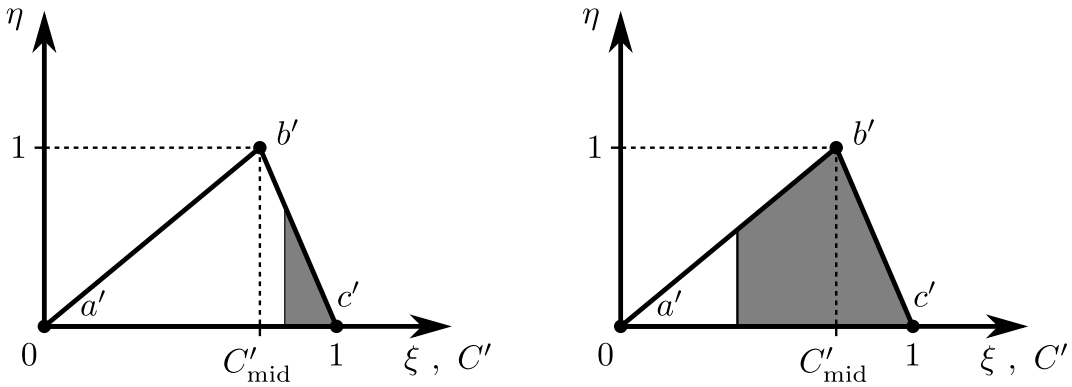


Fig. 5.4: Schematic of transformation of a triangular cell.

After transforming the triangle to natural coordinate system and normalizing the volume fraction, the interface can only be positioned vertically, as C' forms the horizontal axis (see Fig. 5.4). Moreover, there exists only two possible cases of the interface location, either to the right side or left side of C'_{mid} as shown in Fig. 5.5(a) and Fig. 5.5(b) respectively. Therefore, it is possible to obtain a closed-form formula for reconstruction



(a) $C_i <= \left(1 - C'_{\text{mid}}\right)$ (b) $C_i > \left(1 - C'_{\text{mid}}\right)$

Fig. 5.5: Two possibilities of interface location in a triangular cell.

of the interface such that it satisfies the cell averaged volume fraction, C_i, without any need for iterations. The closed-form formulas can be obtained to be,

if, $\quad 0 < C_i \leq \left(1 - C'_{\text{mid}}\right)$,

$$C_L^{ab} = 0\,; \quad C_L^{bc} = \frac{\sqrt{C_i\ \left(1 - C'_{\text{mid}}\right)}}{1 - C'_{\text{mid}}}\,; \quad C_L^{ac} = \sqrt{C_i\ \left(1 - C'_{\text{mid}}\right)}\,,$$

if, $\quad \left(1 - C'_{\text{mid}}\right) < C_i < 1,$

$$C_L^{ab} = 1 - \frac{\sqrt{C'_{\text{mid}}\ \left(1 - C_i\right)}}{C'_{\text{mid}}}\,; \quad C_L^{bc} = 1\,; \quad C_L^{ac} = 1 - \sqrt{C'_{\text{mid}}\ \left(1 - C_i\right)}\,,$$

otherwise,

$$C_L^{ab} = C_L^{bc} = C_L^{ac} = C_i\,.$$

This procedure greatly simplifies the reconstruction of volume fraction at the cell faces for triangular cells. The details of interface reconstruction for triangular cells is provided in Appendix D.

5.2 Computation of volume fraction flux

The governing equation for advection of the interface in an incompressible fluid flow is given by,

$$\frac{\partial C}{\partial t} + \frac{\partial\left(u\,C\right)}{\partial x} + \frac{\partial\left(v\,C\right)}{\partial y} = 0\,,$$

where, C is the volume fraction of the primary fluid, t is time and (x, y) are Cartesian space coordinates. The velocity vector is given by $\vec{V} = (u, v)$. The above equation can be discretized using the finite volume formulation for a cell to obtain,

$$\Omega_i \frac{\partial C_i}{\partial t} + \int_\Gamma \left(u\,C, v\,C\right) \cdot \hat{n} d\Gamma = 0\,,$$

where, $\hat{n}$ is the unit normal pointing outside of the cell volume, Γ is the surface enclosing the cell. For a cell enclosed by M linear faces the above equation can be simplified to,

$$\Omega_i \frac{\partial C_i}{\partial t} + \sum_{f=1}^{M} (u\,C, v\,C)_f \cdot \hat{n}_f\,\Gamma_f = 0\,,$$

where, $\hat{n}_f$ is the unit normal of face f and Γ_f is the area of face f. The vector quantity $(u\,C, v\,C)_f$ is the volume fraction flux through face f.

As the advection equation is a hyperbolic partial differential equation, the volume fraction flux can be computed using a Riemann solver. A very high order accurate, left and the right states, C_L and C_R, at the face can be reconstructed using the procedure described in the previous section. The values of C_L and C_R are such that the unit normal at the face points from left state to the right state. The convective flux for the advection problem can be obtained simply by upwinding, which results in the following flux at the face,

$$(u\,C, v\,C)_f \cdot \hat{n}_f = \begin{cases} \lambda_f\,C_L & \text{if } \lambda_f > 0, \\ \lambda_f\,C_R & \text{otherwise.} \end{cases}$$

where, $\lambda_f = \vec{V}_f \cdot \hat{n}_f$ is the eigenvalue of the advection equation at the face, with interpolated velocity at the face denoted by $\vec{V}_f$.

5.3 Results and discussion

In all the figures displaying the interface, three contours are plotted, at levels 0.2, 0.5 and 0.8, to display the spread of the interface. The numerical error in the solution is computed for a mesh with N cells as,

$$\text{error} = \frac{\sum_{i=1}^{N} \left| C_i^{\text{exact}} - C_i^{\text{numer}} \right|}{N}\,,$$

where, C_i^{exact} and C_i^{numer} are the exact and numerical cell averages of cell i, respectively. The error in results is computed for the new method and compared with interface compression algorithm. In all the test cases a Courant number, $CFL = 0.1$, is used to maintain stability of the solver.

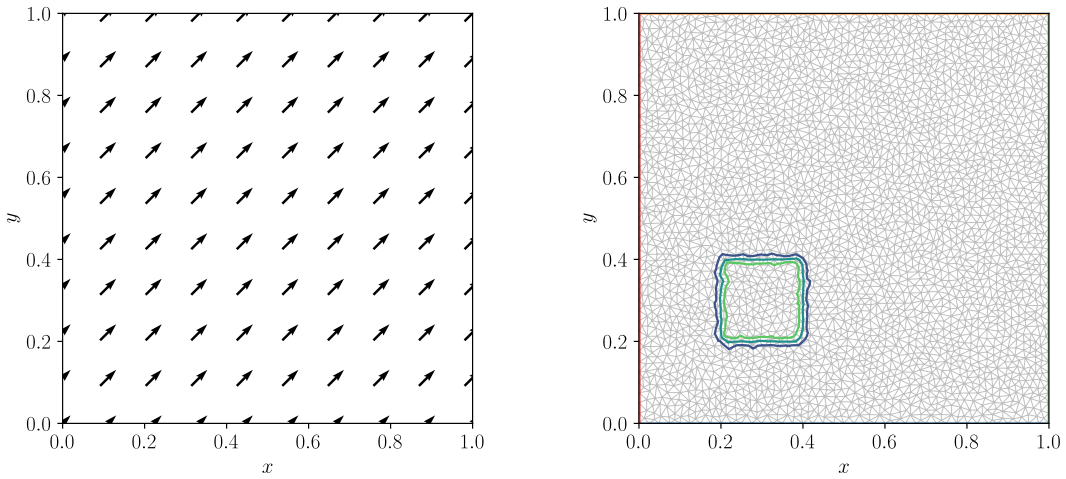
Advection of a square

The uniform velocity field with $\vec{V} = (1, 1)$ and the contours of initial volume fraction for the problem are shown in Fig. 5.6(a) and Fig. 5.6(b) respectively. The initial volume

fraction is defined as,

$$C = \begin{cases} 1 & 0.2 \leq x, y \leq 0.4, \\ 0 & \text{otherwise.} \end{cases}$$

In such a velocity field, the exact solution can be easily computed, by translating the initial volume fraction (square in this case) and calculating the cell averages at the translated location. The exact and numerical solution at a time of 0.4 units is plotted in Fig. 5.7 with refined unstructured meshes consisting of 370 cells, 1444 cells and 5870 cells. The left side column of the figure shows the exact solution, the middle column shows the results obtained with the new interface reconstruction method and the right side column shows the results obtained with interface compression. The errors for different meshes are plotted on a log-log axes in Fig. 5.8 and tabulated in Tab. 5.1. It can be observed from the contour plots and the error analysis that the new reconstruction method captures the interface more sharply and has smaller errors compared to the commonly used interface compression method.



(a) Velocity field (b) Initial condition (5870 cells)

Fig. 5.6: The velocity field and initial conditions for scalar advection of square.

Tab. 5.1: Numerical errors in the square advection problem.

Cell size (No. Cells)	Error	
	Interface reconstruction	Interface compression
0.1 (370 cells)	7.869×10^{-3}	1.8925×10^{-2}
0.05 (1444 cells)	6.252×10^{-3}	1.2189×10^{-2}
0.025 (5870 cells)	2.722×10^{-3}	5.675×10^{-3}

Solid rotation of slotted disk (Zalesak disk problem)

The velocity field, $\vec{V} = (0.5 - y, x - 0.5)$, and the initial volume fraction for the problem are shown in Fig. 5.9(a) and Fig. 5.9(b) respectively. The velocity field is such that it

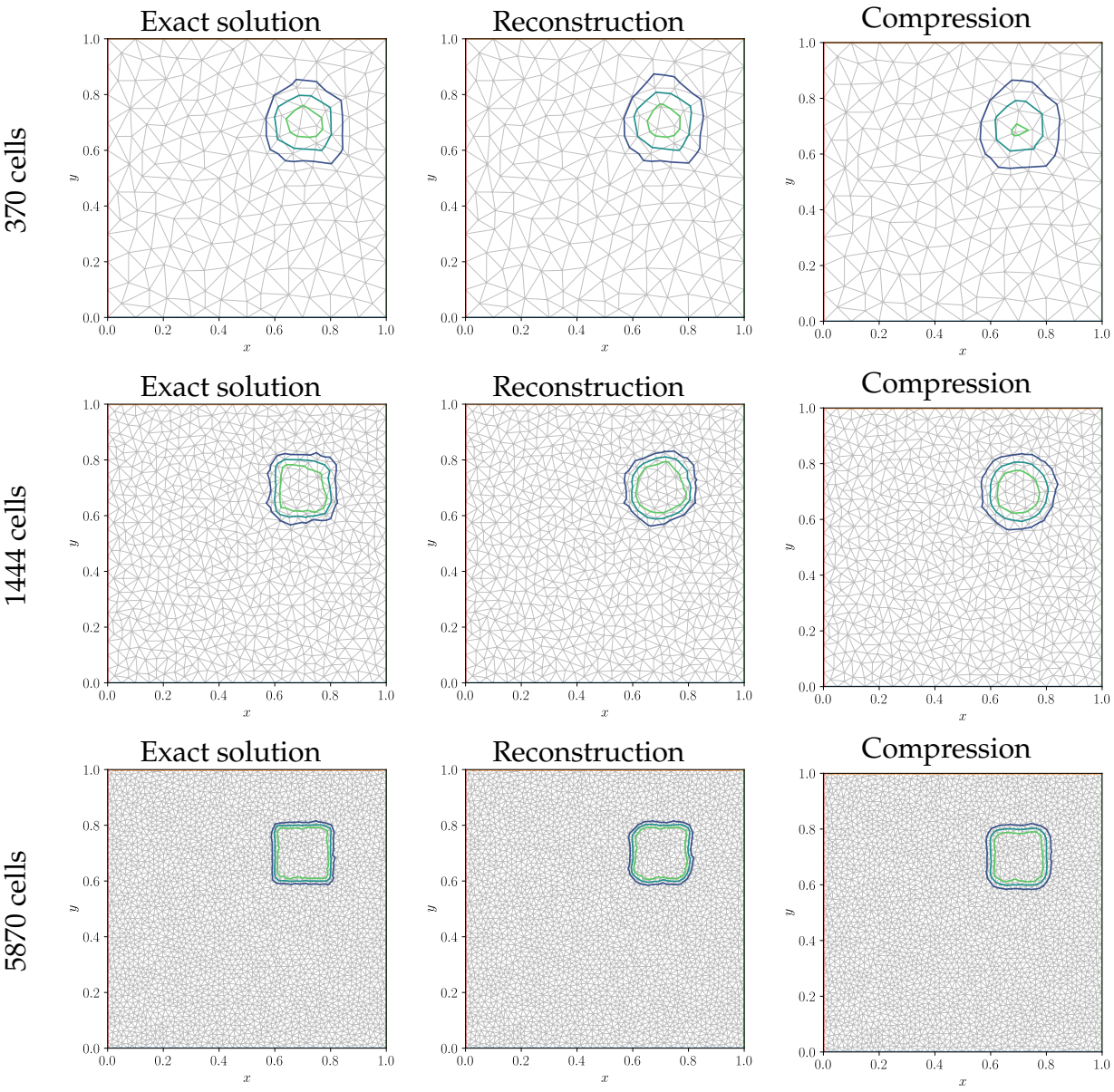


Fig. 5.7: Comparison of solutions obtained at $t = 0.4$, by interface reconstruction and interface compression.

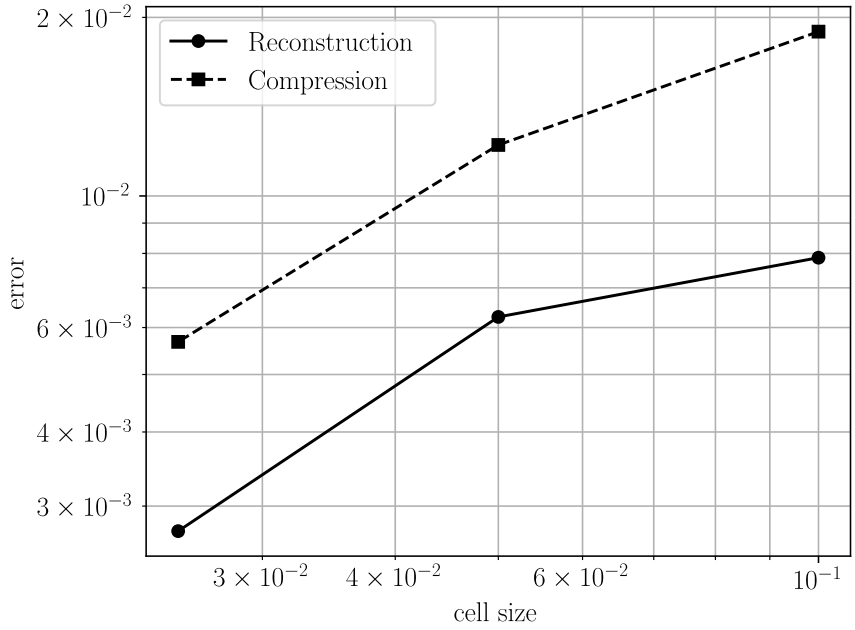
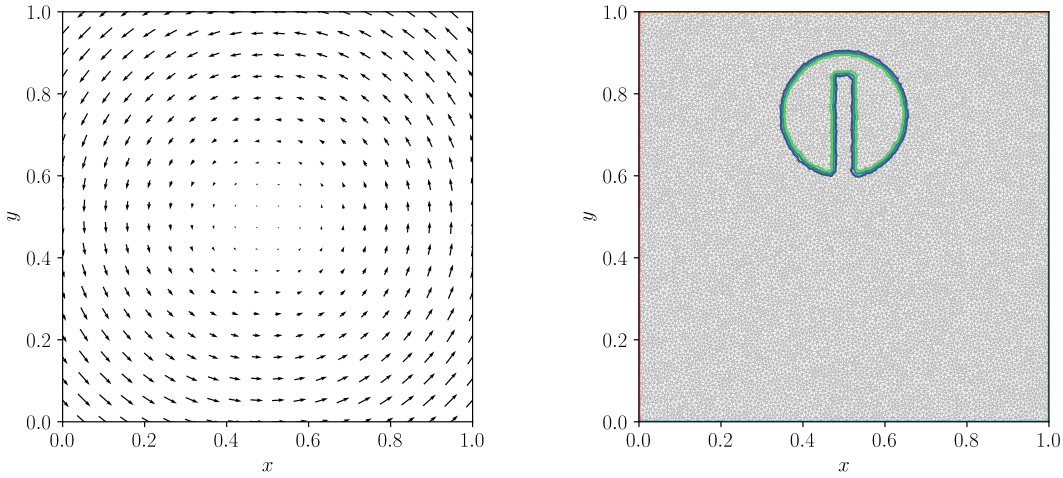


Fig. 5.8: Comparison of numerical errors in the square advection problem using the interface reconstruction and interface compression methods with different levels of mesh refinement.

causes solid rotation without any shear. The initial volume fraction is defined by a slotted circle of radius 0.15 centered at $(0.5, 0.75)$, and a slot-width of 0.05 and slot-height of 0.25. After one complete rotation, in $2\,\pi$ time units, the initial volume fraction returns to its original position. The exact solution after one complete rotation is therefore same as the initial volume fraction. The exact and numerical solution at a time of $2\,\pi$ units is plotted in Fig. 5.10 with refined unstructured meshes consisting of 1444 cells, 5870 cells and 23420 cells. The left side column of the figure shows the exact solution, the middle column shows the results obtained with the new interface reconstruction method and the right side column shows the results obtained with interface compression. The errors for different meshes are plotted on a log-log axes in Fig. 5.11. The errors are also tabulated in Tab. 5.2. It can be observed from the contour plots and the error analysis that the new reconstruction method captures the interface more sharply and has smaller errors compared to the interface compression method.

Tab. 5.2: Numerical errors in the solid rotation of slotted disk problem.

Cell size (No. Cells)	Error	
	Interface reconstruction	Interface compression
0.05 (1444 cells)	1.752×10^{-2}	2.5571×10^{-2}
0.025 (5870 cells)	6.613×10^{-3}	1.1255×10^{-2}
0.0125 (23420 cells)	3.299×10^{-3}	5.361×10^{-3}



(a) Velocity field (b) Initial condition (23420 cells)

Fig. 5.9: The velocity field and initial conditions for solid rotation of slotted disk.

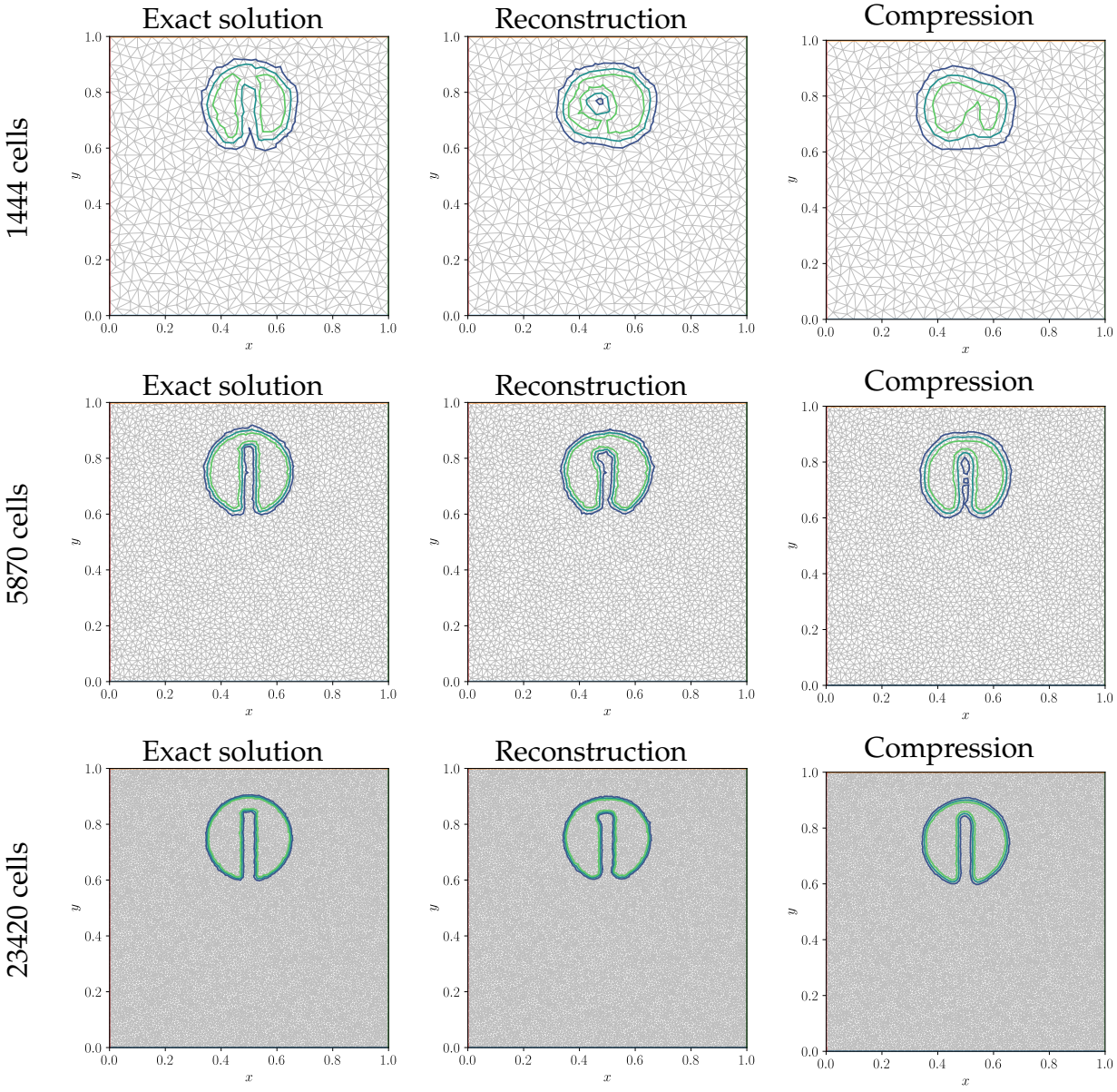


Fig. 5.10: Comparison of solutions obtained at $t = 2\,\pi$, by interface reconstruction and interface compression.

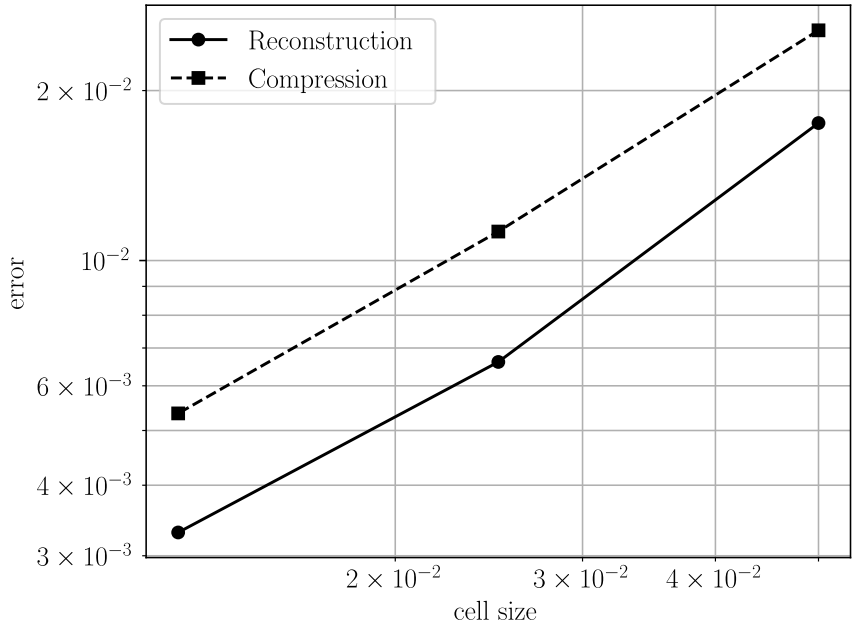


Fig. 5.11: Comparison of numerical errors in the rotation of slotted disk problem using the interface reconstruction and interface compression methods with different levels of mesh refinement.

Circle in shear velocity field (Vortex problem)

In this problem a circular area, with volume fraction $C = 1$, is placed in a non-divergent shear velocity field. The simulation is carried out for a time of $t = 2$ units, with the velocity field reversed at the half-time, $t = 1$ units. The non-divergent shear velocity field is given by,

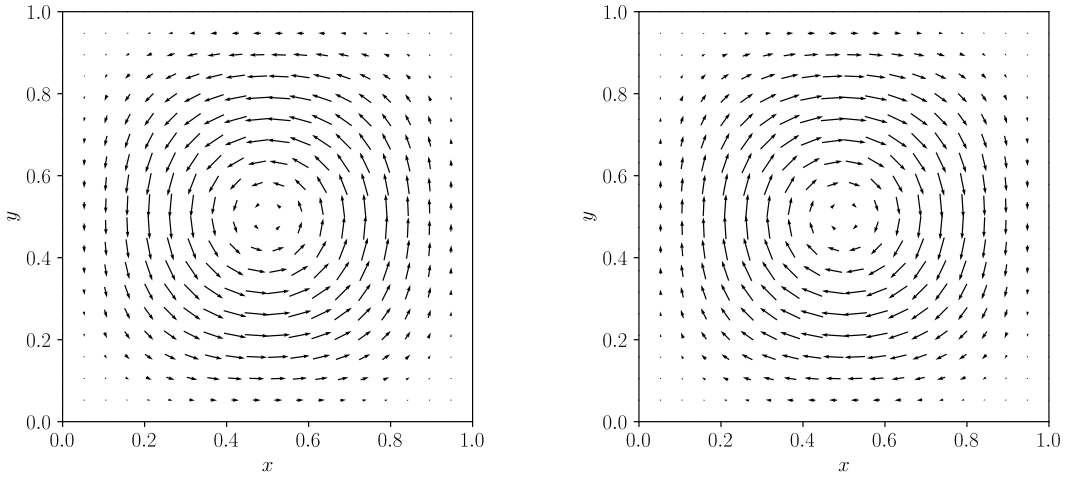
$$\vec{V} = \begin{cases} \left(\sin^2(\pi x)\,\sin(2\pi y)\,, -\sin^2(\pi y)\,\sin(2\pi x)\right) & 0 \leq t \leq 1\,, \\ \left(-\sin^2(\pi x)\,\sin(2\pi y)\,, \sin^2(\pi y)\,\sin(2\pi x)\right) & 1 < t \leq 2\,. \end{cases}$$

and is also shown in Fig. 5.12(a) and Fig. 5.12(b). Such a reversing velocity field is expected to restore the initial volume fraction at time $t = 2$ units. The initial volume fraction is defined by a circle of radius 0.15 centered at $(0.5, 0.75)$, such that,

$$C = \begin{cases} 1 & (x-0.5)^2 + (y-0.75)^2 \leq 0.15^2, \\ 0 & \text{otherwise.} \end{cases}$$

At $t = 2$ units, the initial volume fraction returns to its original position, therefore the exact solution is same as the initial volume fraction. The exact and numerical solution at the intermediate and final time are plotted in Fig. 5.13, Fig. 5.14 and Fig. 5.15 with unstructured meshes consisting of 370 cells, 1444 cells and 5870 cells, respectively. The left side column of the figure shows the exact solution at time $t = 2$ units, the upper-right row shows the results obtained with the new interface reconstruction method and the lower row shows the results obtained with interface compression. The errors for different meshes are plotted on a log-log axes in Fig. 5.16. The errors are also tabulated

in Tab. 5.3. It can be observed from the contour plots and the error analysis that the new reconstruction method captures the interface more sharply and has smaller errors compared to the interface compression method.



(a) Velocity field ($0 \leq t \leq 1$) (b) Velocity field ($1 < t \leq 2$)

Fig. 5.12: The shearing velocity field for vortex problem.

Tab. 5.3: Numerical errors in the vortex problem.

Cell size (No. Cells)	Error	
	Interface reconstruction	Interface compression
0.1 (370 cells)	4.2151×10^{-2}	5.6375×10^{-2}
0.05 (1444 cells)	7.128×10^{-3}	2.5398×10^{-2}
0.025 (5870 cells)	2.83×10^{-3}	9.545×10^{-3}

Low amplitude sloshing

In this problem a tank is half filled with a heavy fluid and the remaining half is a light fluid. The tank has dimensions of 0.1×0.1 m with the mean interface height of 0.05 m. This problem was also studied in section 4.1.2. The interface between the fluids is slightly perturbed initially as shown in Fig. 5.17, defined as,

$$y' = 0.05 + 0.005 \cos\left(2\,\pi\,x/0.2\right)\,, \tag{5.2}$$

which results in an initial volume fraction of,

$$C = \begin{cases} 1 & y \leq y'\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{5.3}$$

Due to gravity, $g = (0, -9.81)$, the interface starts oscillating back and forth at a period $P \approx 0.374$ s. In this simulation the fluxes are computed using the HLL Riemann

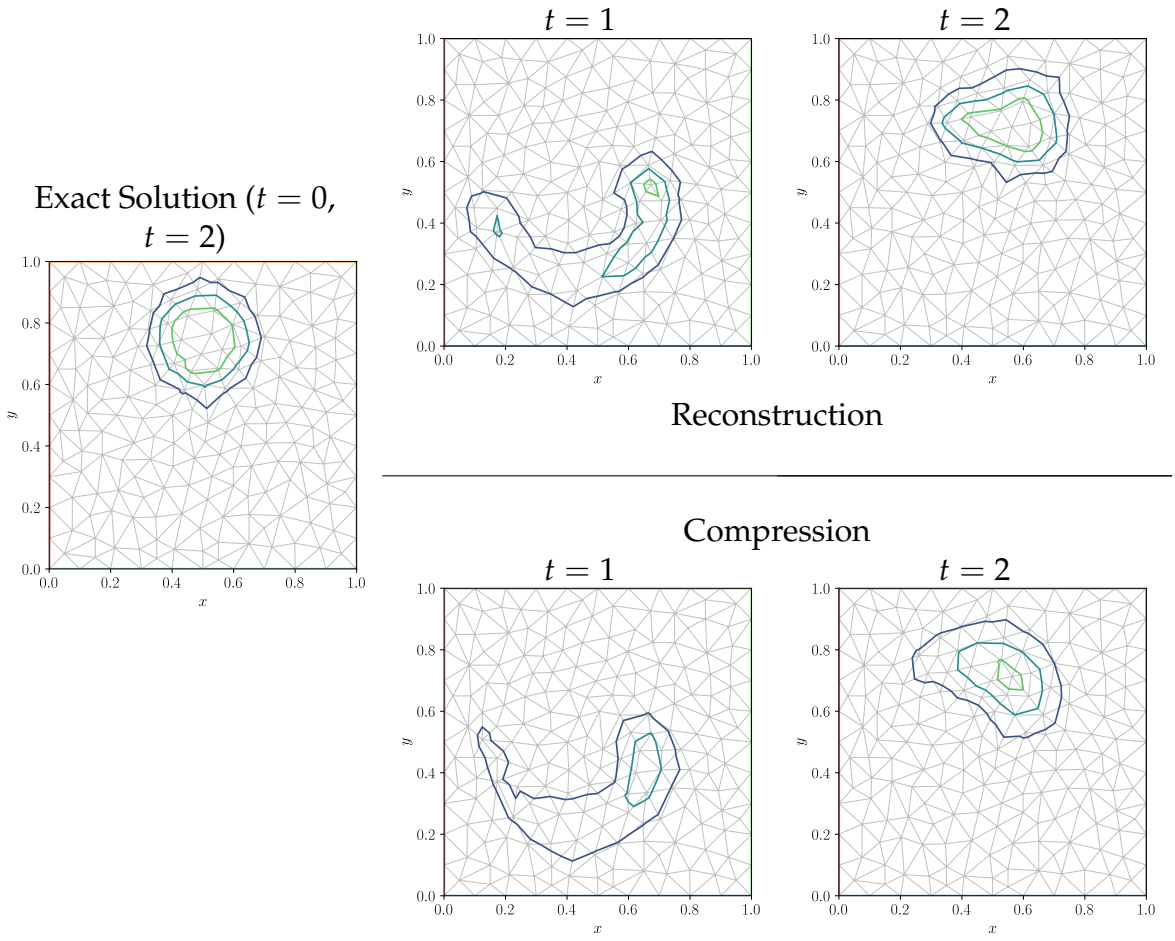


Fig. 5.13: Comparison of solutions obtained at $t = 1$ and $t = 2$, by interface reconstruction and interface compression with mesh consisting of 370 cells.

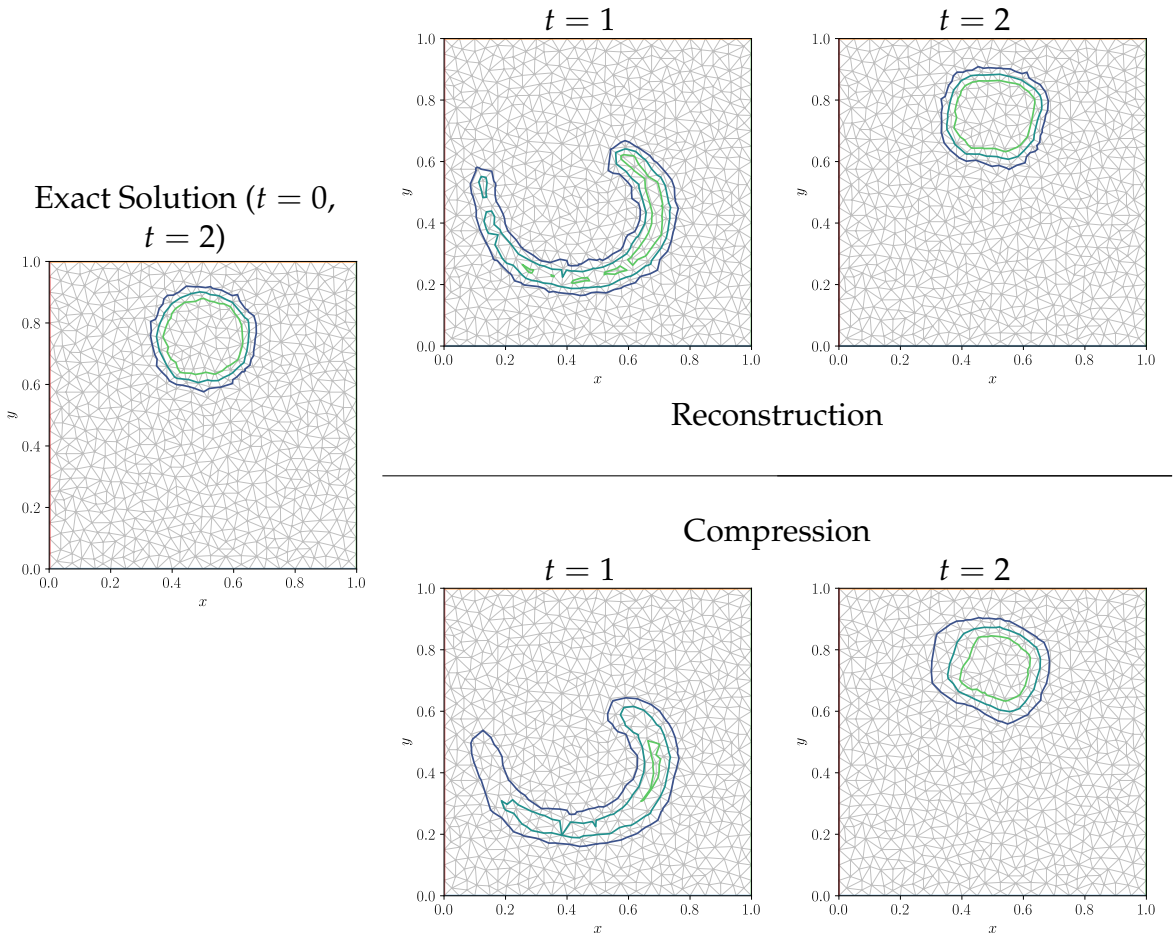


Fig. 5.14: Comparison of solutions obtained at $t = 1$ and $t = 2$, by interface reconstruction and interface compression with mesh consisting of 1444 cells.

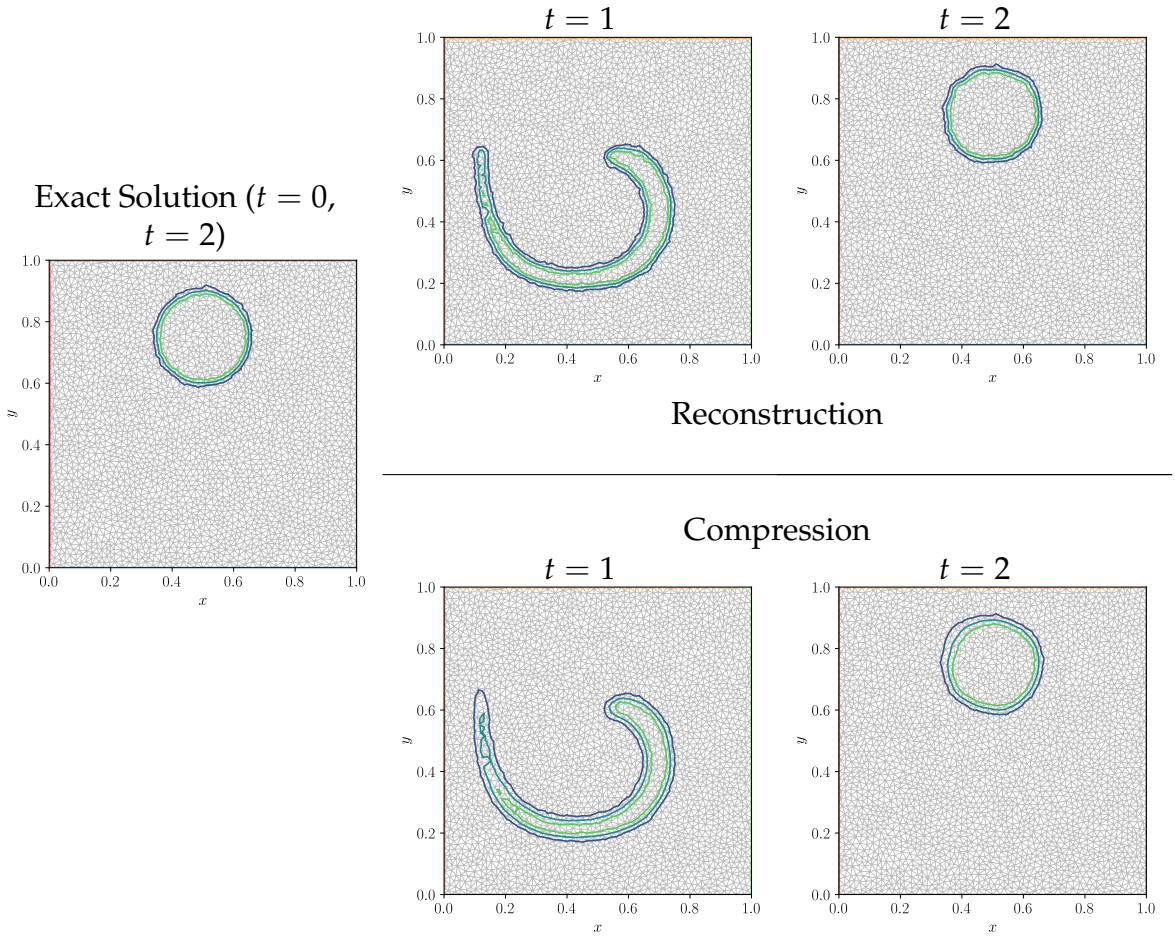


Fig. 5.15: Comparison of solutions obtained at $t = 1$ and $t = 2$, by interface reconstruction and interface compression with mesh consisting of 5870 cells.

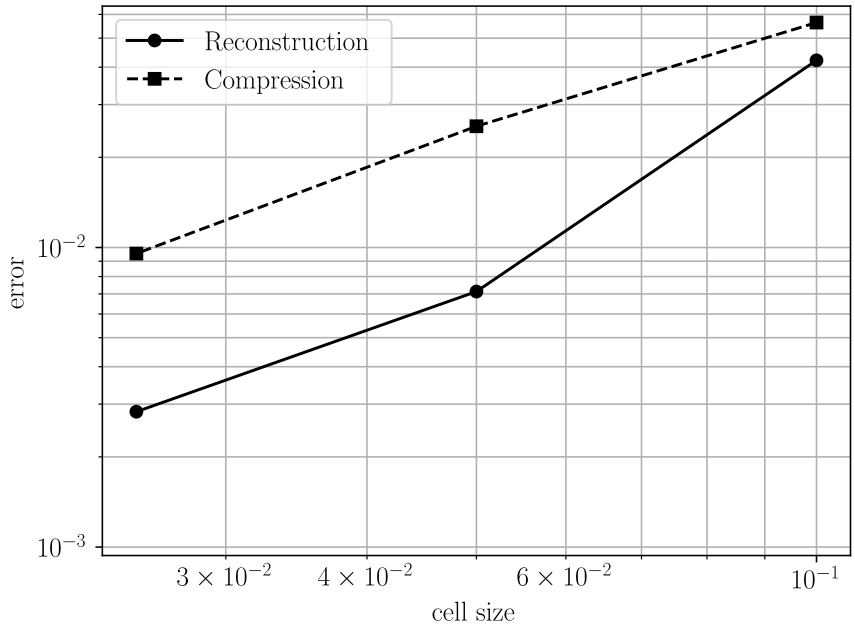


Fig. 5.16: Comparison of numerical errors in the vortex problem using the interface reconstruction and interface compression methods with different levels of mesh refinement.

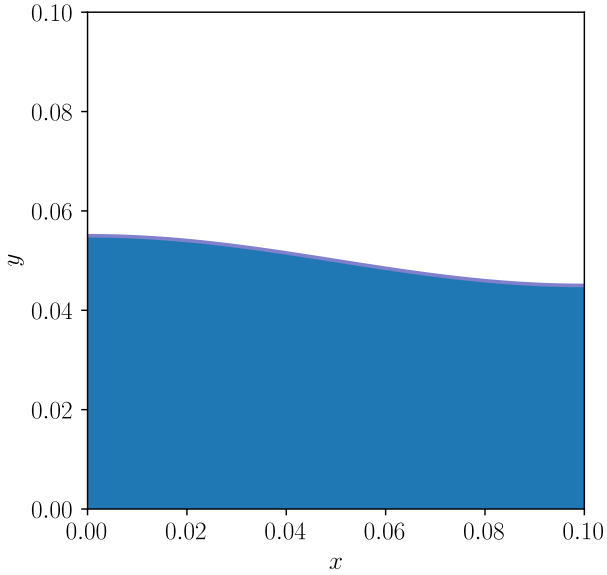


Fig. 5.17: Initial interface location for the low amplitude sloshing problem.

solver. The results obtained by using the two methodologies, one with interface reconstruction and other with interface compression, are compared in Fig. 5.18 using coarse meshes. In Fig. 5.19 the comparison is made for a finer mesh. The HLL Riemann solver is expected to induce numerical viscosity which results in smeared interfaces in case of the compression algorithm. However, it can be observed from the results that the new reconstruction method still maintains a sharp interface.

Dam break problem

In this problem a water column having a width of 0.45 m and a height of 0.9 m is placed at the left bottom corner of a domain of dimensions $2\,\mathrm{m} \times 1\,\mathrm{m}$. The remaining space is filled with air having a density of $1.2\,\mathrm{kg/m^3}$. The water has a density of $998\,\mathrm{kg/m^3}$. Due to gravity ($g = (0, -9.81)\,\mathrm{m/s^2}$) the column of water collapses and the water front moves towards the right wall of the domain. This transient simulation is carried out using the new interface reconstruction method. The flux of mass and momentum are computed using the HLL Riemann solver, which is know to be very diffusive and if used for interface flux computation will result in extremely wide interfaces on coarse meshes, as seen in the previous chapter. The result of the simulation at different times is displayed in Fig .5.20. It can be seen in the Fig. 5.20 the mesh is made up of very coarse triangular cells. Even then the new reconstruction algorithm is able to capture the interface within one cell size.

5.4 Concluding remarks

The results obtained from the new interface reconstruction algorithm are promising. The error analysis is performed for three scalar problems involving simple advection, rotation and shearing velocity fields. In all the scalar problems it has been observed

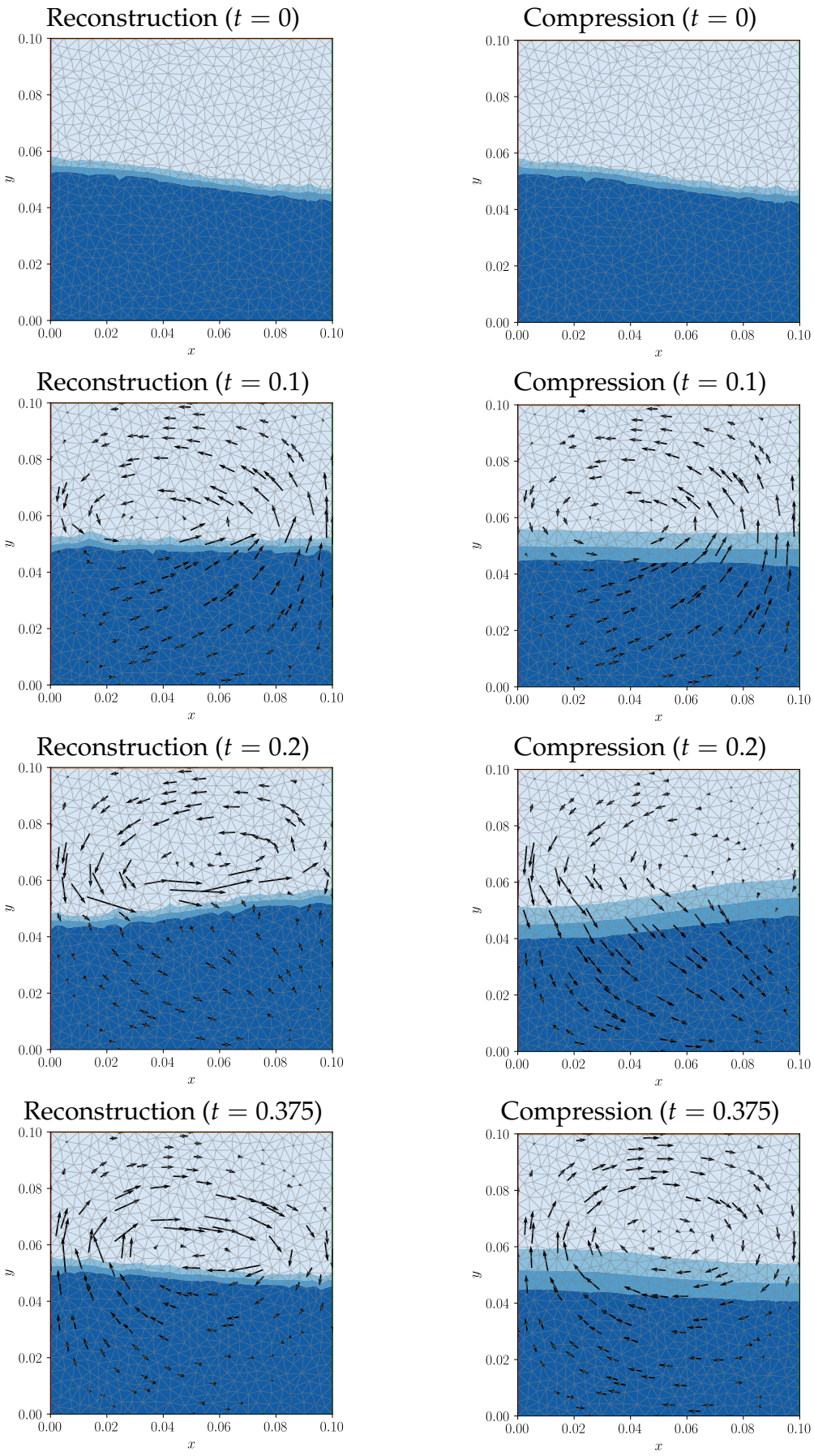


Fig. 5.18: Comparison of results obtained using interface reconstruction and interface compression methods in a sloshing problem on a coarser mesh.

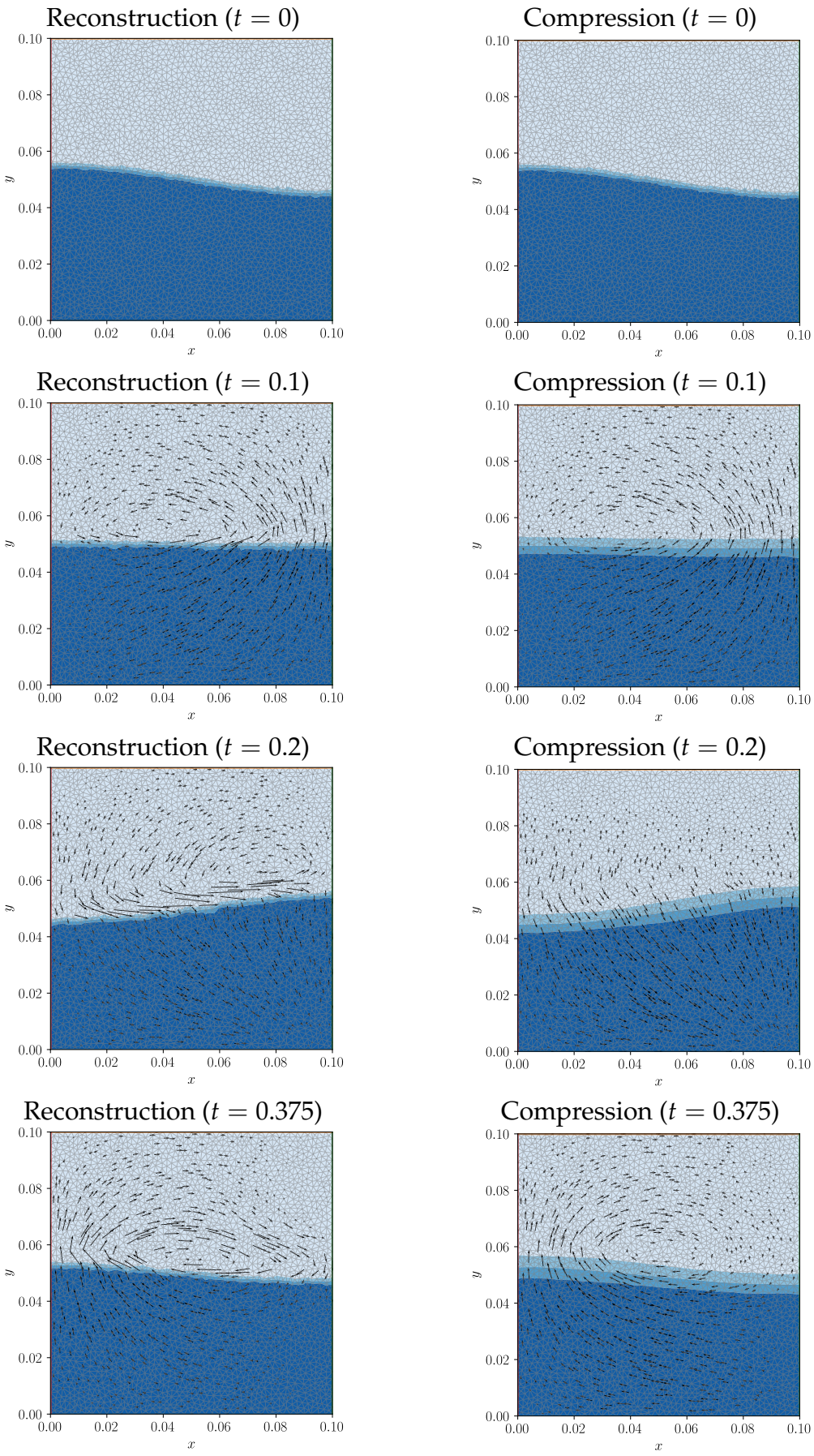


Fig. 5.19: Comparison of results obtained using interface reconstruction and interface compression methods in a sloshing problem on a finer mesh.

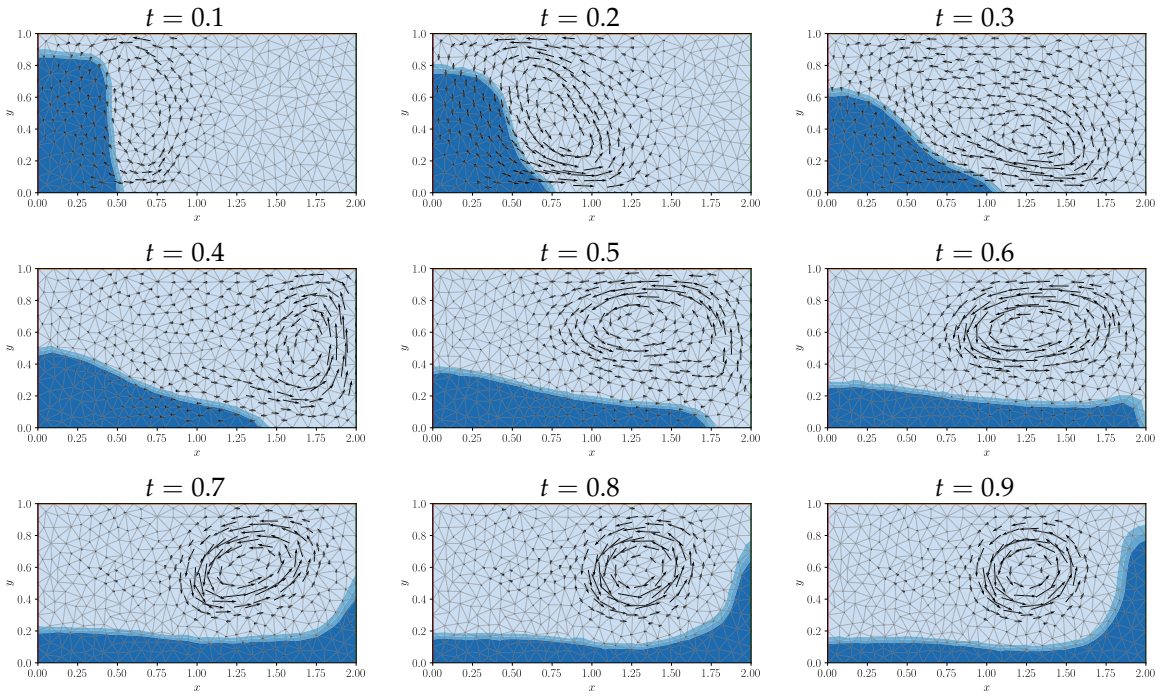


Fig. 5.20: Snapshots of the simulation of the dam break problem at different time levels.

that the error in the new reconstruction algorithm is consistently smaller compared to interface compression algorithm on the same mesh.

To understand the applicability of the method to two-phase problems the sloshing problem is studied on a coarse mesh and fine mesh. The interface reconstruction method performs better on this problem as it is capable of maintaining a sharp fluid interface. The two-phase dam break problem is also studied, which is known to be a difficult problem as it involves high density ratio of about 1000 across the interface, with high velocity fields. Moreover, the problem is studied on a coarse mesh and using the HLL Riemann solver, which are known to cause excessive smearing of the interfaces, as seen from the previous results in section 4.1.2. Even under these conditions the new interface reconstruction algorithm successfully captures a very sharp interface.

Even though this method displays promising results, the technique of reconstruction also poses a severe restriction on time step size. Due to the a high order reconstruction of volume fraction, a very small Courant number needs to be used to maintain stability. A high order explicit time integration or implicit time integration may be needed, but has not been attempted during this work. At this stage of work, the new method of reconstruction and flux calculation is accurate, but is computationally costly, compared to interface compression algorithm.

Chapter 6

Summary, Future Work and Contributions

The density-based solvers are widely used to simulate compressible flows. These solvers can also be used for incompressible flows through artificial compressibility formulation. The artificial compressibility formulation was initially developed for single-phase incompressible flows, and accurate Riemann solvers have since been developed. However, for incompressible two-phase flows, robust Riemann solvers are still lacking. In this work, an accurate and robust Riemann solver called HLLC-VOF is developed. Also, to reduce the dissipation error, an interface reconstruction technique is developed which can work in tandem with Riemann solvers.

6.1 Summary

The objective of this research was to develop accurate and practical methods for numerical simulation of incompressible two-phase flows. This objective has been fulfilled through two novel contributions. The incompressible two-phase flow is modeled using the artificial compressibility formulation closely coupled with the VOF equation.

The first major contribution resulting from this research is the development of HLLC-VOF Riemann solver for computing the convective flux. The convective flux Jacobian of a two-phase flow system has three *distinct* eigenvalues. The HLLC-VOF Riemann solver uses a three-wave system, which is therefore an exact solution to the linearized Riemann problem. The intermediate wave behaves like the contact wave in compressible flows. This is also verified using generalized Riemann invariant (GRI) analysis. The developed solver is found to produce superior results compared to the non-contact preserving Riemann solver. Additionally, the Roe-type Riemann solver for incompressible two-phase flows, from literature, is also improved by incorporating interface compression. The improved Roe-type Riemann solver is found to be equally good as the newly developed HLLC-VOF Riemann solver.

The second major contribution, resulting from this research work, is a method for

interface reconstruction. The new technique is based on the marching cubes method and easily extends to arbitrary polygonal or polyhedral cells. The additional benefit of this technique is that the high-order left and right Riemann states from the reconstructed fluid interface can be easily computed. Thus, the convection flux can be computed using a Riemann solver which is not limited by cell shape and works equally well for structured and unstructured meshes. The method is further simplified for triangular cells using the symmetries of triangles, which makes the method very efficient for triangular unstructured mesh.

6.2 Future research

The research area of incompressible two-phase flows is still very fertile and there are many open questions to be answered.

- The effect of artificial compressibility parameter, β, on results and computational cost, is not well-understood for two-phase flows. In case of single-phase flow, many investigations have been performed by various researchers to obtain an optimal β. Such investigations are also required for two-phase flows to improve the solver's robustness and efficiency.

- In this work, an explicit solver is used to advance the solution in pseudo-time. This imposes a restriction on the pseudo-time step size, thus reducing the rate of convergence. An implicit solver along with acceleration techniques such as multi-grid may be helpful in improving the convergence of the solver.

- The artificial compressibility formulation provides a way for pressure-velocity coupling similar to the pressure based methods. However, the computational cost of these methods in solving a given problem needs to be investigated and compared for two-phase flows.

- In this research, the interface reconstruction algorithm is studied only for two dimensional cells. This method needs to be extended to three-dimensions.

- Heat and mass transfer in two-phase flows have important engineering applications, such as power generation. Such examples are not solved, as this research focused on developing methods for accurately capturing the interface. The methods developed in this work can be applied to such complex flows with heat and mass transfer in the future.

- The open-source code, developed during this research, can be extended to work efficiently on clusters using advanced computing techniques. Also, algorithms such as multi-grid and implicit time integration can be incorporated to obtain better computational efficiency.

6.3 Contributions

This section provides a list of the research contributions along with a very brief overview. The research publications resulting from the work are also enlisted.

6.3.1 Main contributions

- A contact preserving Riemann solver, named as HLLC-VOF – It is a new solver for incompressible two-phase flows, capable of computing accurate convective flux and better interface resolution.
- An accurate technique for interface reconstruction and flux computation – It is a technique for computing high-order volume fraction reconstruction, which works in tandem with a Riemann solver. The new method is also easy to extend to arbitrarily shaped cells.

The above contributions are described in detail in this thesis, and therefore are not elaborated here.

6.3.2 Other contributions

- An open-source code for solving wide variety of partial differential equations using object-oriented-design principles – the code is designed for extensibility without compromising on computational efficiency. The solver heavily relies on polymorphic principles of software design, thus making use of interface-class instead of concrete-class for most of its software components. The same code therefore can be used to solve, scalar advection equation, single-phase incompressible equations, two-phase incompressible equations, compressible Euler equations, Navier-Stokes equations etc. The code has also been tested with phase-field model for grain-growth. To change the governing equations only a single class needs to be defined, implementing the `GoverningEquations` interface-class, while the remaining heavy-lifting is taken care of by the solver. The solver is available at: https://spbhat.in/solver/
  A few features of the developed solver are briefly described below:
  - It is easy to define new governing equations describing new physical phenomena.
  - It is easy to define new or use pre-defined Riemann solvers.
  - The solver is capable of solving 1D, 2D or 3D problems efficiently.
  - The solver is capable of simulating steady-state and transient problems.
  - It is easy to define new or use pre-defined time integrators (Euler, second-order and third-order stability preserving Runge-Kutta are already implemented).

  - It is easy to define new accurate reconstruction schemes or choose the solution accuracy – such as piecewise constant, linear TVD, ENO/WENO reconstructions.
  - It is easy to define new boundary conditions – many commonly used boundary conditions are already predefined.
  - The supported mesh file format is similar to SU^2, therefore, software such as Gmsh or Pointwise can be used for mesh generation.
  - Parallel computing using all the available processors on the machine is inbuilt, and the code is run in parallel by default. The code can be made to run as a serial process by modifying a single file.
  - All the Visualization Toolkit (VTK) geometry types are supported – 1D, 2D and 3D. Thus, cell shapes such as, line, triangle, quadrilateral, polygons, tetrahedron, hexahedron, pyramid, prism etc. are supported.
  - The VTK library is inbuilt and thus does not need any additional installation.
  - The modules for handling geometry, gradient calculators etc. are implemented using standardized efficient algorithms.
  - The dependence on external libraries is minimal. The solver uses only one highly efficient external math library – open source Apache commons math library. This is automatically wired through Maven tools, but also can be downloaded separately, if necessary.
  - The solver uses automated unit testing framework and has a code coverage of about 98%. Also, online continuous integration tools – Travis, Shippable – are used for testing and deployment.
  - The easiest way to use the solver is to install IntelliJ IDEA, and open the project. Then, navigate to `test/main/` and run any of the code files, such as `LidDrivenCavity2DTest` or `TransientFlowOverCylinderTest`. The code will automatically scale to use all the CPUs of the computer.

- An open-source library for writing Visualization Toolkit (VTK) files – capable of creating output in legacy or modern XML, compressed base64 format to save disk space. The files are readable by commonly used visualization software, such as ParaView or Tecplot.

- An open-source library for creating structured 2D mesh using transfinite interpolation – with a capability of creating complex meshes using block-mesh technique.

- An open-source library for creating structured 3D mesh using transfinite interpolation.

6.3.3 Journals

- **Bhat Sourabh**, Mandal J. C., "Contact preserving Riemann solver for incompressible two-phase flows", *Journal of Computational Physics*, vol. 379, pp. 173–191, Feb. 2019.

6.3.4 Conferences

- **Bhat S P**, Parameswaran S, Mandal J C, "On the order of accuracy analysis of SDWLS method", *ICTACEM 2017*, December 2017, IIT Kharagpur, India.
- **Bhat S P**, Mandal J C, "Artificial Compressibility Based Method for Two-phase Surface Tension Dominated Flows", *Annual CFD Symposium 2017*, August 2017, NAL Bangalore, India.
- **Bhat S P**, Mandal J C, "A Novel HLLC-Type Riemann Solver for Two-Phase Incompressible Flows", *Proceedings of 6th International & 43rd National Conference on Fluid Mechanics and Fluid Power*, December 2016, Allahabad, India.
- **Bhat S P**, Mandal J C, "A novel reinitialization technique to conserve mass and enhance accuracy in VOF method", *Proceedings of 4th International Conference on Computational Methods for Thermal Problems*, Issue 217349, July 6-8, 2016, Georgia Tech, Atlanta, USA.
- Mandal J C, Parsai A, Parameswaran S, **Bhat S P**, "A novel volume of fluid method for interface tracking", *5th International and 41st National Conference on Fluid Mechanics and Fluid Power*, December 12-14, 2014, Kanpur, India.

Acknowledgements

I would like to express my deepest gratitude and respect to my Ph.D. supervisor **Prof. J. C. Mandal** for the encouragement and critical feedback which was necessary to take up challenging problems. His knowledgeable comments at the right time helped me to tackle the difficult research problems. It would be impossible for me to complete the research successfully without his insightful guidance. I am also grateful to my research committee members **Prof. Prabhu Ramachandran**, **Prof. S. Gopalakrishnan** and **Prof. Kannan Iyer**, for asking the key questions which kept the research moving forward in the right direction. I would like to sincerely thank my lab mates **Sanal Parameswaran**, **Shainath Kalamkar**, **Sangeeth Simon**, **Sundeep Rao** and **Mohammad Belal**, for the fruitful discussions. I would like to especially thank **Sanal Parameswaran** for the long and inspiring technical discussions which gave a new shape and dimension to the research.

I would like to extend my gratitude towards the Yogastha club at IIT Bombay and its members, for providing a platform for extra curricular activities during my stay on the campus. This helped me maintain a healthy body and calm mind to continue through the stressful situations. Especially, I would like to thank **Mrs. Kaumudi Gadre**, **Dr. Amrendra Narayan Bharat**, **Chandra Pratap Singh**, **Sowmya Gupta**, **Priyanka Jena**, **Poornachandra Shwetha**, **Supriya Khandeparkar** and **Kedar Khandeparkar**.

I would also like to thank the institute and the Aerospace department at IIT Bombay for providing the computational facilities to conduct the numerical simulations and providing the financial assistance during the course of the research.

Finally, I would like to take this opportunity to thank the near and dear to my heart, my friends and family. I would like to thank **Athira Pushpakaran** for being with me through the challenging times during the PhD. I am sure that I would have not been able to bring this work to this degree of completion without her support. I would also like to express my deepest gratitude and respect to my Mother, **Seema Pramod Bhat** for always encouraging me, being supportive and having a positive outlook towards the future.

Last but not the least, I would like to thank the almighty power that rules the world, which brought it all together and made it all possible.

– Sourabh Pramod Bhat

References

[1] A. Reza Kohansal and H. Ghassemi, "A numerical modeling of hydrodynamic characteristics of various planing hull forms," *Ocean Engineering*, vol. 37, pp. 498–510, apr 2010.

[2] K. Saha, A. Kumar Agarwal, K. Ghosh, and S. Som, eds., *Two-Phase Flow for Automotive and Power Generation Sectors*. Energy, Environment, and Sustainability, Singapore: Springer Singapore, 2019.

[3] R. Singh and V. Bajpai, *Coolant and Lubrication in Machining*, pp. 1–34. London: Springer London, 2013.

[4] D. Mazumdar and R. I. L. Guthrie, "The Physical and Mathematical Modelling of Gas Stirred Ladle Systems," *ISIJ International*, vol. 35, no. 1, pp. 1–20, 1995.

[5] M. Narcy and C. Colin, "TWO-PHASE PIPE FLOW IN MICROGRAVITY WITH AND WITHOUT PHASE CHANGE: RECENT PROGRESS AND FUTURE PROSPECTS," *Interfacial Phenomena and Heat Transfer*, vol. 3, no. 1, pp. 1–17, 2015.

[6] A. A. Hyman, C. A. Weber, and F. Jülicher, "Liquid-Liquid Phase Separation in Biology," *Annual Review of Cell and Developmental Biology*, vol. 30, pp. 39–58, oct 2014.

[7] D. Fusco and P. Charbonneau, "Crystallization of asymmetric patchy models for globular proteins in solution," *Physical Review E*, vol. 88, p. 012721, jul 2013.

[8] D. Enright, S. Marschner, and R. Fedkiw, "Animation and rendering of complex water surfaces," *ACM Transactions on Graphics*, vol. 21, jul 2002.

[9] R. Bridson, *Fluid Simulation for Computer Graphics, Second Edition*. A K Peters/CRC Press, sep 2015.

[10] F. H. Harlow and J. E. Welch, "Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface," *Physics of Fluids*, vol. 8, no. 12, pp. 2182–2189, 1965.

[11] C. Hirt and B. Nichols, "Volume of fluid (VOF) method for the dynamics of free boundaries," *Journal of Computational Physics*, vol. 39, pp. 201–225, Jan. 1981.

[12] B. J. Daly, "Numerical Study of Two Fluid Rayleigh-Taylor Instability," *Physics of Fluids*, vol. 10, no. 2, pp. 297–307, 1967.

[13] W. F. Noh and P. Woodward, "SLIC (Simple Line Interface Calculation)," in *Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics*, (Twente University, Enschede), 1976.

[14] D. L. Youngs, *Time-dependent multi-material flow with large fluid distortion.* Academic Press, 1982.

[15] S. O. Unverdi and G. Tryggvason, "A Front-Tracking Method for Viscous, Incompressible, Multi-fluid Flows," *Journal of Computational Physics*, vol. 100, pp. 25–37, 1992.

[16] 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, pp. 146–159, 1994.

[17] E. Olsson and G. Kreiss, "A conservative level set method for two phase flow," *Journal of Computational Physics*, vol. 210, no. 1, pp. 225–246, 2005.

[18] D. Jacqmin, "Calculation of Two-Phase Navier-Stokes Flows Using Phase-Field Modeling," *Journal of Computational Physics*, vol. 155, pp. 96–127, oct 1999.

[19] C.-n. Ji and Y. Shi, "Application of the VOF method based on unstructured quadrilateral mesh," *Journal of Marine Science and Application*, vol. 7, pp. 24–32, mar 2008.

[20] S. Diot and M. M. François, "An interface reconstruction method based on an analytical formula for 3D arbitrary convex cells," *Journal of Computational Physics*, vol. 305, pp. 63–74, jan 2016.

[21] M. Skarysz, A. Garmory, and M. Dianat, "An iterative interface reconstruction method for PLIC in general convex grids as part of a Coupled Level Set Volume of Fluid solver," *Journal of Computational Physics*, vol. 368, pp. 254–276, sep 2018.

[22] 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, pp. 708–759, May 2001.

[23] G. Tryggvason, R. Scardovelli, and S. Zaleski, *Direct Numerical Simulations of Gas-Liquid Multiphase Flows*. Cambridge: Cambridge University Press, 2011.

[24] M. Huang, L. Wu, and B. Chen, "A Piecewise Linear Interface-Capturing Volume-of-Fluid Method Based on Unstructured Grids," *Numerical Heat Transfer Part B: Fundamentals*, vol. 61, no. 51176152, pp. 412–437, 2012.

[25] K. Ito, T. Kunugi, H. Ohshima, and T. Kawamura, "A volume-conservative PLIC algorithm on three-dimensional fully unstructured meshes," *Computers & Fluids*, vol. 88, pp. 250–261, Dec. 2013.

[26] J. M. Hyman, "Numerical Methods for Tracking Interfaces," *Physica D: Nonlinear Phenomena*, vol. 12, no. 1-3, pp. 396–407, 1984.

[27] I.-l. Chern, J. Glimm, O. McBryan, B. Plohr, and S. Yaniv, "Front Tracking for Gas Dynamics," *Journal of Computational Physics*, vol. 62, pp. 83–110, 1986.

[28] M. R. H. Nobari and G. Tryggvason, "Numerical Simulations of Three-Dimensional Drop Collisions," *AIAA Journal*, vol. 34, no. 4, pp. 750–755, 1996.

[29] J. U. Brackbill, D. B. Kothe, and C. Zemach, "A Continuum Method for Modeling Surface Tension," *Journal of Computational Physics*, vol. 100, pp. 335–354, 1992.

[30] F. H. Harlow, J. P. Shannon, and J. E. Welch, "Liquid Waves by Computer," *American Association for the Advancement of Science*, vol. 149, no. 3688, pp. 1092–1093, 1965.

[31] F. H. Harlow and J. P. Shannon, "The Splash of a Liquid Drop," *Journal of Applied Physics*, vol. 38, no. 10, pp. 3855–3866, 1967.

[32] A. A. Amsden and F. H. Harlow, "A Simplified MAC Technique for Incompressible Fluid Flow Calculations," *Journal of Computational Physics*, vol. 6, pp. 322–325, 1970.

[33] A. A. Amsden and F. H. Harlow, "The SMAC Method: A Numerical Technique for Calculating Incompressible Fluid Flows," tech. rep., 1970.

[34] A. Tomiyama, I. Zun, A. Sou, and T. Sakaguchi, "Numerical analysis of bubble motion with the VOF method," *Nuclear Engineering and Design*, vol. 141, pp. 69–82, June 1993.

[35] M. Rudman, "Volume Tracking Methods for Interfacial Flows Calculations," *International Journal for Numerical Methods in Fluids*, vol. 24, pp. 671–691, 1997.

[36] W. J. Rider and D. B. Kothe, "Reconstructing Volume Tracking," *Journal of Computational Physics*, vol. 141, pp. 112–152, 1998.

[37] 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, pp. 251–274, 2003.

[38] J. E. Pilliod and E. G. Puckett, "Second-order accurate volume-of-fluid algorithms for tracking material interfaces," *Journal of Computational Physics*, vol. 199, pp. 465–502, Sept. 2004.

[39] E. Aulisa, S. Manservisi, R. Scardovelli, and S. Zaleski, "Interface reconstruction with least-squares fit and split advection in three-dimensional Cartesian geometry," *Journal of Computational Physics*, vol. 225, pp. 2301–2319, Aug. 2007.

[40] F. Xiao, Y. Honma, and T. Kono, "A simple algebraic interface capturing scheme using hyperbolic tangent function," *International Journal for Numerical Methods in Fluids*, vol. 48, pp. 1023–1040, jul 2005.

[41] B. Xie, S. Ii, and F. Xiao, "An efficient and accurate algebraic interface capturing method for unstructured grids in 2 and 3 dimensions: The THINC method with quadratic surface representation," *International Journal for Numerical Methods in Fluids*, vol. 76, pp. 1025–1042, Dec. 2014.

[42] K. Yokoi, "Efficient implementation of THINC scheme: A simple and practical smoothed VOF algorithm," *Journal of Computational Physics*, vol. 226, pp. 1985–2002, oct 2007.

[43] O. Ubbink and R. I. Issa, "A Method for Capturing Sharp Fluid Interfaces on Arbitrary Meshes," *Journal of Computational Physics*, vol. 153, pp. 26–50, 1999.

[44] A. J. Chorin, "A numerical method for solving incompressible viscous flow problems," *Journal of Computational Physics*, vol. 2, pp. 12–26, aug 1967.

[45] D. Pan and C. H. Chang, "The capturing of free surfaces in incompressible multi-fluid flows," *International Journal for Numerical Methods in Fluids*, vol. 33, no. 2, pp. 203–222, 2000.

[46] Y. Zhao, H. Hui Tan, and B. Zhang, "A High-Resolution Characteristics-Based Implicit Dual Time-Stepping VOF Method for Free Surface Flow Simulation on Unstructured Grids," *Journal of Computational Physics*, vol. 183, pp. 233–273, Nov. 2002.

[47] S. N. Yakovenko and K. C. Chang, "Volume fraction flux approximation in a two-fluid flow," *Thermophysics and Aeromechanics*, vol. 15, pp. 169–186, Nov. 2008.

[48] Y. G. Chen, W. G. Price, and P. Temarel, "An anti-diffusive VOF method for computational modelling of sloshing in an LNG tank," in *11th International Symposium on Practical Design of Ships and Other Floating Structures*, (Rio de Janeiro, BR), p. 9, 2010.

[49] H. Rusche, *Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phase Fractions*. PhD thesis, Imperial College of Science, Technology & Medicine, 2002.

[50] H. Jasak, "OpenFOAM: Open source CFD in research and industry," *International Journal of Naval Architecture and Ocean Engineering*, vol. 1, pp. 89–94, dec 2009.

[51] H. Lee and S. H. Rhee, "A dynamic interface compression method for VOF simulations of high-speed planing watercraft," *Journal of Mechanical Science and Technology*, 2015.

[52] N. Ashgriz and J. Poo, "FLAIR: Flux line-segment model for advection and interface reconstruction," *Journal of Computational Physics*, vol. 93, pp. 449–468, apr 1991.

[53] J. P. Boris and D. L. Book, "Flux-corrected transport. I. SHASTA, a fluid transport algorithm that works," *Journal of Computational Physics*, vol. 11, pp. 38–69, jan 1973.

[54] 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.

[55] W. Mulder, S. Osher, and J. A. Sethian, "Computing interface Motion in Compressible Gas Dynamics," *Journal of Computational Physics*, vol. 100, no. 2, pp. 209–228, 1992.

[56] D. Adalsteinsson and J. A. Sethian, "A Fast Level Set Method for Propagating Interfaces," *Journal of Computational Physics*, vol. 118, pp. 269–277, 1995.

[57] Y. C. Chang, T. Y. Hou, B. Merriman, and S. Osher, "A Level Set Formulation of Eulerian Interface Capturing Methods for Incompressible Fluid Flows," *Journal of Computational Physics*, vol. 124, pp. 449–464, 1996.

[58] M. Sussman, E. Fatemi, P. Smereka, and S. Osher, "An improved level set method for incompressible two-phase flows," *Computers & Fluids*, vol. 27, pp. 663–680, June 1998.

[59] M. Sussman and E. G. Puckett, "A Coupled Level Set and Volume-of-Fluid Method for Computing 3D and Axisymmetric Incompressible Two-Phase Flows," *Journal of Computational Physics*, vol. 162, pp. 301–337, Aug. 2000.

[60] S. Osher and R. P. Fedkiw, "Level Set Methods: An Overview and Some Recent Results," *Journal of Computational Physics*, vol. 169, pp. 463–502, May 2001.

[61] J. A. Sethian, "Evolution, Implementation, and Application of Level Set and Fast Marching Methods for Advancing Fronts," *Journal of Computational Physics*, vol. 169, pp. 503–555, May 2001.

[62] D. Sun and W. Tao, "A coupled volume-of-fluid and level set (VOSET) method for computing incompressible two-phase flows," *International Journal of Heat and Mass Transfer*, vol. 53, pp. 645–655, Jan. 2010.

[63] E. Olsson, G. Kreiss, and S. Zahedi, "A conservative level set method for two phase flow II," *Journal of Computational Physics*, vol. 225, no. 1, pp. 785–807, 2007.

[64] A. W. Adamson and A. P. Gast, *Physical Chemistry of Surfaces, Sixth Edition*. John Wiley & Sons, Inc., 1997.

[65] B. Lafaurie, C. Nardone, R. Scardovelli, S. Zaleski, and G. Zanetti, "Modelling Merging and Fragmentation in Multiphase Flows with SURFER," *Journal of Computational Physics*, vol. 113, pp. 134–147, jul 1994.

[66] D. Gerlach, G. Tomar, G. Biswas, and F. Durst, "Comparison of volume-of-fluid methods for surface tension-dominant two-phase flows," *International Journal of Heat and Mass Transfer*, vol. 49, pp. 740–754, Feb. 2006.

[67] S. Patankar and D. Spalding, "A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows," *International Journal of Heat and Mass Transfer*, vol. 15, pp. 1787–1806, oct 1972.

[68] J. P. Van Doormaal and G. D. Raithby, "Enhancements of the Simple Method for Predicting Incompressible Fluid Flows," *Numerical Heat Transfer*, vol. 7, pp. 147–163, apr 1984.

[69] R. Issa, "Solution of the implicitly discretised fluid flow equations by operator-splitting," *Journal of Computational Physics*, vol. 62, pp. 40–65, jan 1986.

[70] D. Drikakis, O. P. Iliev, and D. P. Vassileva, "A Nonlinear Multigrid Method for the Three-Dimensional Incompressible Navier-Stokes Equations," *Journal of Computational Physics*, vol. 146, no. 1, pp. 301–321, 1998.

[71] M. T. Manzari, "An explicit finite element algorithm for convection heat transfer problems," *International Journal of Numerical Methods for Heat & Fluid Flow*, vol. 9, pp. 860–877, dec 1999.

[72] J. C. Mandal and A. S. Iyer, "An upwind method for incompressible flow computations using pseudo-compressibility approach," in *19th AIAA Computational Fluid Dynamics Conference*, 2009.

[73] A. G. Malan, R. W. Lewis, and P. Nithiarasu, "An improved unsteady, unstructured, artificial compressibility, finite volume scheme for viscous incompressible flows: Part I. Theory and implementation," *International Journal for Numerical Methods in Engineering*, vol. 54, pp. 695–714, jun 2002.

[74] P. Nithiarasu, "An efficient artificial compressibility (AC) scheme based on the characteristic based split (CBS) method for incompressible flows," *International Journal for Numerical Methods in Engineering*, vol. 56, pp. 1815–1845, apr 2003.

[75] E. Turkel, "Preconditioned methods for solving the incompressible and low speed compressible equations," *Journal of Computational Physics*, vol. 72, pp. 277–298, oct 1987.

[76] A. L. Gaitonde, "A dual-time method for two-dimensional unsteady incompressible flow calculations," *International Journal for Numerical Methods in Engineering*, vol. 41, no. 6, pp. 1153–1166, 1998.

[77] R. R. Nourgaliev, T. N. Dinh, and T. G. Theofanous, "A pseudocompressibility method for the numerical simulation of incompressible multifluid flows," *International Journal of Multiphase Flow*, vol. 30, no. 7, pp. 901–937, 2004.

[78] W. G. Price and Y. G. Chen, "A simulation of free surface waves for incompressible two-phase flows using a curvilinear level set formulation," *International Journal for Numerical Methods in Fluids*, vol. 51, no. 3, pp. 305–330, 2006.

[79] N. Evstigneev, "Integration Of 3D Incompressible Free Surface Navier-Stokes Equations On Unstructured Tetrahedral Grid Using Distributed Computation On TCP-IP Networks," *Advances in Fluid Mechanics*, vol. 59, pp. 65–77, May 2008.

[80] Y.-Y. Niu and J. R. Edwards, "A simple incompressible flux splitting for sharp free surface capturing," *International Journal for Numerical Methods in Fluids*, vol. 69, pp. 1661–1678, jul 2011.

[81] A. Yang, S. Chen, L. Yang, and X. Yang, "An upwind finite volume method for incompressible inviscid free surface flows," *Computers & Fluids*, vol. 101, pp. 170–182, sep 2014.

[82] S. Bhat and J. Mandal, "Contact preserving Riemann solver for incompressible two-phase flows," *Journal of Computational Physics*, vol. 379, pp. 173–191, feb 2019.

[83] P. Tamamidis, G. Zhang, and D. N. Assanis, "Comparison of Pressure-Based and Artificial Compressibility Methods for Solving 3D Steady Incompressible Viscous Flows," *Journal of Computational Physics*, vol. 124, no. 1, pp. 1–13, 1996.

[84] A. Jameson, "Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings," in *10th Computational Fluid Dynamics Conference*, Fluid Dynamics and Co-located Conferences, American Institute of Aeronautics and Astronautics, jun 1991.

[85] P. Nithiarasu, J. S. Mathur, N. P. Weatherill, and K. Morgan, "Three-dimensional incompressible flow calculations using the characteristic based split(CBS) scheme," *International Journal for Numerical Methods in Fluids*, vol. 44, pp. 1207–1229, apr 2004.

[86] Z. Zhang, A. Gil, O. Hassan, and K. Morgan, "The simulation of 3D unsteady incompressible flows with moving boundaries on unstructured meshes," *Computers & Fluids*, vol. 37, pp. 620–631, jun 2008.

[87] S.K.Godunov, "A Difference Method for Numerical Calculation of Discontinuous Solutions of the Equations of Hydrodynamics," *Matematicheskii Sbornik*, vol. 47(89), no. 3, pp. 271–306, 1959.

[88] E. F. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[89] V. Rusanov, "The calculation of the interaction of non-stationary shock waves and obstacles," *USSR Computational Mathematics and Mathematical Physics*, vol. 1, pp. 304–320, jan 1962.

[90] A. Harten, P. D. Lax, and B. van Leer, "On Upstream Differencing and Godunov-Type Schemes for Hyperbolic Conservation Laws," *SIAM Review*, vol. 25, pp. 35–61, jan 1983.

[91] P. Roe, "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, vol. 43, pp. 357–372, oct 1981.

[92] E. F. Toro, M. Spruce, and W. Speares, "Restoration of the contact surface in the HLL-Riemann solver," *Shock Waves*, vol. 4, pp. 25–34, 1994.

[93] E. F. Toro, "Viscous Flux Limiters," in *Proceedings of the Ninth GAMM-Conference on Numerical Methods in Fluid Mechanics*, pp. 592–600, 1991.

[94] B. van Leer, "Towards the ultimate conservative difference scheme I. The quest of monotonicity," in *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics* (R. Cabannes, Henri and Temam, ed.), pp. 163–168, Springer Berlin Heidelberg, 1973.

[95] D. Book, J. Boris, and K. Hain, "Flux-corrected transport II: Generalizations of the method," *Journal of Computational Physics*, vol. 18, pp. 248–283, jul 1975.

[96] J. Boris and D. Book, "Flux-corrected transport. III. Minimal-error FCT algorithms," *Journal of Computational Physics*, vol. 20, pp. 397–431, apr 1976.

[97] A. Harten, "High resolution schemes for hyperbolic conservation laws," *Journal of Computational Physics*, vol. 49, pp. 357–393, mar 1983.

[98] A. Harten, "On a Class of High Resolution Total-Variation-Stable Finite-Difference Schemes," *SIAM Journal on Numerical Analysis*, vol. 21, no. 1, pp. 1–23, 1984.

[99] P. K. Sweby, "High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws," *SIAM Journal on Numerical Analysis*, vol. 21, no. 5, pp. 995–1011, 1984.

[100] P. Sweby, ""TVD" Schemes for Inhomogeneous Conservation Laws," in *Nonlinear Hyperbolic Equations - Theory, Computation Methods, and Applications* (J. Ballmann and R. Jeltsch, eds.), vol. 24 of *Notes on Numerical Fluid Mechanics*, pp. 599–607, Vieweg Teubner Verlag, 1989.

[101] P. Roe, "Some contributions to the modelling of discontinuous flows," in *Large-Scale Computations in Fluid Mechanics* (B. Engquist, S. Osher, and R. Somerville, eds.), pp. 163–193, 1985.

[102] G. van Albada, B. van Leer, and W. Roberts Jr., "A comparative study of computational methods in cosmic gas dynamics," *Astronomy and Astrophysics*, vol. 108, pp. 76–84, apr 1982.

[103] P. Roe, "Characteristic-Based Schemes for the Euler Equations," *Annual Review of Fluid Mechanics*, vol. 18, pp. 337–365, 1986.

[104] B. Van Leer, "Towards the ultimate conservative difference scheme III. Upstream-centered finite-difference schemes for ideal compressible flow," *Journal of Computational Physics*, vol. 23, pp. 263–275, mar 1977.

[105] J. Mandal and J. Subramanian, "On the link between weighted least-squares and limiters used in higher-order reconstructions for finite volume computations of hyperbolic equations," *Applied Numerical Mathematics*, vol. 58, pp. 705–725, May 2008.

[106] J. C. Mandal and S. P. Rao, "High resolution finite volume computations on unstructured grids using solution dependent weighted least squares gradients," *Computers and Fluids*, vol. 44, pp. 23–31, May 2011.

[107] C.-w. Shu, "Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws," tech. rep., NASA Langley Research Center, 1997.

[108] C. Hu and C.-W. Shu, "Weighted Essentially Non-Oscillatory Schemes on Triangular Meshes," *Journal of Computational Physics*, vol. 150, pp. 97–127, 1999.

[109] D. S. Balsara and C.-W. Shu, "Monotonicity Preserving Weighted Essentially Non-oscillatory Schemes with Increasingly High Order of Accuracy," *Journal of Computational Physics*, vol. 160, no. 2, pp. 405–452, 2000.

[110] M. Lahooti and A. Pishevar, "A CWENO ghost fluid method for compressible multimaterial flow," *International Journal for Numerical Methods in Fluids*, vol. 73, pp. 904–926, 2013.

[111] Y. Liu and Y. T. Zhang, "A Robust Reconstruction for Unstructured WENO Schemes," *Journal of Scientific Computing*, vol. 54, no. 2-3, pp. 603–621, 2013.

[112] V. Coralic and T. Colonius, "Finite-volume WENO scheme for viscous compressible multicomponent flows.," *Journal of computational physics*, vol. 274, pp. 95–121, Oct. 2014.

[113] Z. J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. T. Huynh, N. Kroll, G. May, P. O. Persson, B. van Leer, and M. Visbal, "High-order CFD methods: Current status and perspective," *International Journal for Numerical Methods in Fluids*, vol. 72, pp. 811–845, July 2013.

[114] M. Dumbser and C.-D. Munz, "ADER discontinuous Galerkin schemes for aeroacoustics," *Comptes Rendus Mécanique*, vol. 333, pp. 683–687, sep 2005.

[115] G. Montecinos, C. Castro, M. Dumbser, and E. Toro, "Comparison of solvers for the generalized Riemann problem for hyperbolic systems with source terms," *Journal of Computational Physics*, vol. 231, pp. 6472–6494, aug 2012.

[116] V. Venkatakrishnan, "Convergence to Steady State Solutions of the Euler Equations on Unstructured Grids with Limiters," *Journal of Computational Physics*, vol. 118, pp. 120–130, apr 1995.

[117] S. Gottlieb, D. I. Ketcheson, and C.-W. Shu, "High Order Strong Stability Preserving Time Discretizations," *Journal of Scientific Computing*, vol. 38, pp. 251–289, mar 2009.

[118] D. J. Mavriplis and A. Jameson, "Multigrid solution of the Navier-Stokes equations on triangular meshes," *AIAA Journal*, vol. 28, pp. 1415–1425, aug 1990.

[119] R. C. Swanson, E. Turkel, and J. A. White, "An effective multigrid method for high-speed flows," tech. rep., ICASE Report No. 91-56, 1991.

[120] J. Blazek, "Chapter 6 - Temporal Discretization," in *Computational Fluid Dynamics: Principles and Applications* (J. Blazek, ed.), ch. 6, pp. 167–211, Oxford: Butterworth-Heinemann, third ed., 2015.

[121] A. Arnone, M.-S. Liou, and L. A. Povinelli, "Multigrid time-accurate integration of Navier-Stokes equations," in *11th Computational Fluid Dynamics Conference*, Fluid Dynamics and Co-located Conferences, American Institute of Aeronautics and Astronautics, jul 1993.

[122] M. M. Andersen, B. Barker, S. Brisard, A. D. Chou, M. Diggory, and R. B. Donkin, "Commons Math: The Apache Commons Mathematics Library," *Apache Commons*, vol. 3.6.1, 2016.

[123] Y. Mehmani, "Wrinkle-Free Interface Compression for Two-Fluid Flows," nov 2018.

[124] I. Tadjbakhsh and J. B. Keller, "Standing surface waves of finite amplitude," *Journal of Fluid Mechanics*, vol. 8, no. 03, pp. 442–451, 1960.

[125] V. R. Gopala and B. G. van Wachem, "Volume of fluid methods for immiscible-fluid and free-surface flows," *Chemical Engineering Journal*, vol. 141, pp. 204–221, July 2008.

[126] J. R. Shewchuk, "Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator," in *Applied Computational Geometry Towards Geometric Engineering*, pp. 203–222, 1996.

[127] J. C. Martin and W. J. Moyce, "Part IV. An Experimental Study of the Collapse of Liquid Columns on a Rigid Horizontal Plane," *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, vol. 244, no. 882, pp. 312–324, 1952.

[128] S. Koshizuka and Y. Oka, "Moving-particle semi-implicit method for fragmentation of incompressible fluid," *Nuclear Science and Engineering*, vol. 123, no. 3, pp. 421–434, 1996.

[129] E. G. Puckett, A. S. Almgren, J. B. Bell, D. L. Marcus, and W. J. Rider, "A High-Order Projection Method for Tracking Fluid Interfaces in Variable Density Incompressible Flows," *Journal of Computational Physics*, vol. 130, pp. 269–282, 1997.

[130] S. Parameswaran and J. Mandal, "A novel Roe solver for incompressible two-phase flow problems," *Journal of Computational Physics*, vol. 390, pp. 405–424, aug 2019.

[131] S. P. Bhat and J. C. Mandal, "A novel reinitialization technique to conserve mass and enhance accuracy in VOF method," in *International Conference on Computational Methods for Thermal Problems* (N. P. Joshi Y. Massarotti N., ed.), no. 217349, 2016.

[132] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *Computer Graphics*, vol. 21, pp. 163–169, aug 1987.

[133] Maxima, "Maxima, a computer algebra system. version 5.34.1," 2014.

Appendix A

Derivation of HLLC-VOF Riemann Solver (3D)

One of the important contributions in this thesis is the development of an accurate Riemann solver, named as HLLC-VOF, for computation of convective flux in an incompressible two-phase flow system. The necessary mathematical expressions of the newly developed solver were presented, and used for solving various two-phase flow problems, in chapter 4. Here a detailed derivation of the HLLC-VOF Riemann solver is presented.

The rotational invariant property of the governing equations effectively converts the Riemann problem to a one-dimensional problem, as discussed in section 3.4.2. Hence, it is convenient to derive the Riemann solver for the x-split governing equations, which simplifies the notations. The temporal and the convective part of the x-split three-dimensional governing equations can be written in a compact form as,

$$\frac{\partial \mathbf{U}}{\partial \tau} + \frac{\partial \mathbf{F}}{\partial x} = 0 \tag{A.1}$$

where,

$$\mathbf{U} = \begin{bmatrix} p/\beta \\ \rho\, u \\ \rho\, v \\ \rho\, w \\ C \end{bmatrix}; \quad \mathbf{F} = \begin{bmatrix} u \\ \rho\, u^2 + p \\ \rho\, u\, v \\ \rho\, u\, w \\ C\, u \end{bmatrix}. \tag{A.2}$$

The Jacobian $\partial \mathbf{F}/\partial \mathbf{U}$ is obtained to be,

$$\partial \mathbf{F}/\partial \mathbf{U} = \begin{bmatrix} 0 & 1/\rho & 0 & 0 & (\rho_2 - \rho_1)\, u/\rho \\ \beta & 2\,u & 0 & 0 & (\rho_2 - \rho_1)\, u^2 \\ 0 & v & u & 0 & (\rho_2 - \rho_1)\, u\, v \\ 0 & w & 0 & u & (\rho_2 - \rho_1)\, u\, w \\ 0 & C/\rho & 0 & 0 & \rho_2\, u/\rho \end{bmatrix}. \tag{A.3}$$

The eigenvalues of the Jacobian are,

$$\lambda = [u_C - a, u, u, u, u_C + a] \tag{A.4}$$

where,

$$u_C = \frac{(\rho + \rho_2)\ u}{2\,\rho} = \frac{u}{2}\left(1 + \frac{\rho_2}{\rho}\right); \quad a = \sqrt{u_C^2 + \beta/\rho}\,. \tag{A.5}$$

The corresponding right eigenvectors written as columns of a matrix are,

$$R_{ev} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ \frac{\rho\,\lambda_1\,(u-\lambda_5)}{\lambda_1-u} & 0 & 0 & 1 & \frac{\rho\,\lambda_5\,(u-\lambda_1)}{\lambda_5-u} \\ \frac{\rho\,v\,\lambda_1}{\lambda_1-u} & 1 & 0 & 0 & \frac{\rho\,v\,\lambda_5}{\lambda_5-u} \\ \frac{\rho\,w\,\lambda_1}{\lambda_1-u} & 0 & 1 & 0 & \frac{\rho\,w\,\lambda_5}{\lambda_5-u} \\ \frac{C\,\lambda_1}{\lambda_1-u} & 0 & 0 & \frac{1}{(\rho_1-\rho_2)\,u} & \frac{C\,\lambda_5}{\lambda_5-u} \end{bmatrix} \tag{A.6}$$

The wave structure is displayed in Fig. 4.1. The three Rankine-Hugoniot jump conditions can be written as,

$$\mathbf{F}_{*L} - \mathbf{F}_L = S_L\,(\mathbf{U}_{*L} - \mathbf{U}_L)\ , \tag{A.7}$$

$$\mathbf{F}_{*R} - \mathbf{F}_{*L} = S_*\,(\mathbf{U}_{*R} - \mathbf{U}_{*L})\ , \tag{A.8}$$

and

$$\mathbf{F}_R - \mathbf{F}_{*R} = S_R\,(\mathbf{U}_R - \mathbf{U}_{*R})\ . \tag{A.9}$$

Assuming that somehow the left and right wave speeds, S_L and S_R respectively, are estimated, the unknowns in the above equations are,

$$\mathbf{U}_{*L},\ \mathbf{U}_{*R},\ \mathbf{F}_{*L},\ \mathbf{F}_{*R} \text{ and } S_*\ . \tag{A.10}$$

We can eliminate two of the unknowns $\mathbf{F}_{*L}$ and $\mathbf{F}_{*R}$ from the above equations, by adding the equations, to obtain a single equation,

$$\mathbf{F}_R - \mathbf{F}_L = S_L\,(\mathbf{U}_{*L} - \mathbf{U}_L) + S_*\,(\mathbf{U}_{*R} - \mathbf{U}_{*L}) + S_R\,(\mathbf{U}_R - \mathbf{U}_{*R})\ . \tag{A.11}$$

Now, we have one equation but three unknowns,

$$\mathbf{U}_{*L},\ \mathbf{U}_{*R},\ \text{and } S_*\ . \tag{A.12}$$

Therefore, more conditions are necessary to get an unique solution. The generalized Riemann invariant (GRI) analysis shows that some of the quantities do not change across the intermediate waves (the waves corresponding to eigenvalues u). This provides the following additional conditions:

$$p_{*L} = p_{*R} = p_*,\ u_{*L} = u_{*R} = u_*,\ v_{*L} \neq v_{*R},\ C_{*L} \neq C_{*R}\ . \tag{A.13}$$

In addition to this, as there is no jump in u across the intermediate waves, the wave speed of all the three intermediate waves will be u_*, therefore,

$$S_* = u_* \tag{A.14}$$

Expanding the vectors of equation (A.11) produces the following system of equations,

$$\begin{aligned}\begin{bmatrix} u \\ \rho u^2 + p \\ \rho u v \\ \rho u w \\ C u \end{bmatrix}_R - \begin{bmatrix} u \\ \rho u^2 + p \\ \rho u v \\ \rho u w \\ C u \end{bmatrix}_L = S_L \left(\begin{bmatrix} p/\beta \\ \rho u \\ \rho v \\ \rho w \\ C \end{bmatrix}_{*L} - \begin{bmatrix} p/\beta \\ \rho u \\ \rho v \\ \rho w \\ C \end{bmatrix}_L \right) \\ + S_* \left(\begin{bmatrix} p/\beta \\ \rho u \\ \rho v \\ \rho w \\ C \end{bmatrix}_{*R} - \begin{bmatrix} p/\beta \\ \rho u \\ \rho v \\ \rho w \\ C \end{bmatrix}_{*L} \right) + S_R \left(\begin{bmatrix} p/\beta \\ \rho u \\ \rho v \\ \rho w \\ C \end{bmatrix}_R - \begin{bmatrix} p/\beta \\ \rho u \\ \rho v \\ \rho w \\ C \end{bmatrix}_{*R} \right) . \end{aligned} \tag{A.15}$$

The first equation is,

$$u_R - u_L + S_L \left(p/\beta\right)_L - S_R \left(p/\beta\right)_R = \left(p/\beta\right)_* \left(S_L - S_R\right) \tag{A.16}$$

which gives,

$$\boxed{\left(p/\beta\right)_* = \frac{u_L - u_R + S_R \left(p/\beta\right)_R - S_L \left(p/\beta\right)_L}{S_R - S_L}} . \tag{A.17}$$

The third equation is,

$$\begin{aligned}\left(\rho u v\right)_R - \left(\rho u v\right)_L = S_L \left(\rho v\right)_{*L} - S_L \left(\rho v\right)_L \\ + S_* \left(\rho v\right)_{*R} - S_* \left(\rho v\right)_{*L} + S_R \left(\rho v\right)_R - S_R \left(\rho v\right)_{*R}\end{aligned} \tag{A.18}$$

and by combining terms,

$$\begin{aligned}\left(S_L - S_*\right) \left(\rho v\right)_{*L} - S_L \left(\rho v\right)_L + \left(\rho u v\right)_L \\ + \left(S_* - S_R\right) \left(\rho v\right)_{*R} + S_R \left(\rho v\right)_R - \left(\rho u v\right)_R = 0 .\end{aligned} \tag{A.19}$$

Now, looking for similar solutions for $\left(\rho v\right)_{*L}$ and $\left(\rho v\right)_{*R}$, we may write,

$$\left(S_L - S_*\right) \left(\rho v\right)_{*L} - S_L \left(\rho v\right)_L + \left(\rho u v\right)_L = 0 \tag{A.20}$$

$$\implies \quad \boxed{\left(\rho v\right)_{*L} = \frac{S_L \left(\rho v\right)_L - \left(\rho u v\right)_L}{S_L - S_*}} . \tag{A.21}$$

$$\left(S_* - S_R\right) \left(\rho v\right)_{*R} + S_R \left(\rho v\right)_R - \left(\rho u v\right)_R = 0 \tag{A.22}$$

$$\implies \quad \boxed{(\rho\, v)_{*R} = \frac{S_R\,(\rho\, v)_R - (\rho\, u\, v)_R}{S_R - S_*}}\,. \tag{A.23}$$

Similarly, the fourth equation provides,

$$\boxed{(\rho\, w)_{*L} = \frac{S_L\,(\rho\, w)_L - (\rho\, u\, w)_L}{S_L - S_*}}\,. \tag{A.24}$$

and,

$$\boxed{(\rho\, w)_{*R} = \frac{S_R\,(\rho\, w)_R - (\rho\, u\, w)_R}{S_R - S_*}}\,. \tag{A.25}$$

The fifth equation is,

$$(C\, u)_R - (C\, u)_L = S_L C_{*L} - S_L C_L + S_* C_{*R} - S_* C_{*L} + S_R C_R - S_R C_{*R} \tag{A.26}$$

and by combining terms,

$$(S_L - S_*)\, C_{*L} - S_L C_L + (C\, u)_L + (S_* - S_R)\, C_{*R} + S_R C_R - (C\, u)_R = 0 \tag{A.27}$$

Now, looking for similar solutions for C_{*L} and C_{*R}, we may write,

$$(S_L - S_*)\, C_{*L} - S_L C_L + (C\, u)_L = 0 \tag{A.28}$$

$$\implies \quad \boxed{C_{*L} = \frac{S_L C_L - (C\, u)_L}{S_L - S_*}}\,. \tag{A.29}$$

$$(S_* - S_R)\, C_{*R} + S_R C_R - (C\, u)_R = 0 \tag{A.30}$$

$$\implies \quad \boxed{C_{*R} = \frac{S_R C_R - (C\, u)_R}{S_R - S_*}}\,. \tag{A.31}$$

The second equation is,

$$\begin{aligned}\left(\rho\, u^2 + p\right)_R - \left(\rho\, u^2 + p\right)_L &= S_L\,(\rho\, u)_{*L} - S_L\,(\rho\, u)_L \\ &\quad + S_*\,(\rho\, u)_{*R} - S_*\,(\rho\, u)_{*L} + S_R\,(\rho\, u)_R - S_R\,(\rho\, u)_{*R}\end{aligned} \tag{A.32}$$

and since,

$$u_{*L} = u_{*R} = u_* = S_* \tag{A.33}$$

the equation may be written, after combining terms, as,

$$\begin{aligned}&S_*\,(S_L - S_*)\,\rho_{*L} - S_*\,(S_R - S_*)\,\rho_{*R} + S_R\,(\rho\, u)_R \\ &\qquad - S_L\,(\rho\, u)_L - \left(\rho\, u^2 + p\right)_R + \left(\rho\, u^2 + p\right)_L = 0\,.\end{aligned} \tag{A.34}$$

This is a non-linear equation in S_* and cannot be solved to obtain an unique solution. Hence, more information is necessary. In case of a two-phase fluid flow the density

function, in the volume of fluid method, is defined as,

$$\rho\left(C\right) = \left(\rho_1 - \rho_2\right) C + \rho_2 \tag{A.35}$$

Therefore,

$$\rho_{*L} = \left(\rho_1 - \rho_2\right) C_{*L} + \rho_2 = \left(\rho_1 - \rho_2\right) \frac{S_L C_L - \left(C\,u\right)_L}{S_L - S_*} + \rho_2\,, \tag{A.36}$$

and,

$$\rho_{*R} = \left(\rho_1 - \rho_2\right) C_{*R} + \rho_2 = \left(\rho_1 - \rho_2\right) \frac{S_R C_R - \left(C\,u\right)_R}{S_R - S_*} + \rho_2\,. \tag{A.37}$$

Substituting the above definitions in equation (A.34) results in,

$$\begin{aligned} S_* \left(S_L - S_*\right) \left[\left(\rho_1 - \rho_2\right) \frac{S_L C_L - \left(C\,u\right)_L}{S_L - S_*} + \rho_2\right] \\ - S_* \left(S_R - S_*\right) \left[\left(\rho_1 - \rho_2\right) \frac{S_R C_R - \left(C\,u\right)_R}{S_R - S_*} + \rho_2\right] \\ + S_R \left(\rho\,u\right)_R - S_L \left(\rho\,u\right)_L - \left(\rho\,u^2 + p\right)_R + \left(\rho\,u^2 + p\right)_L = 0\,. \end{aligned} \tag{A.38}$$

After further simplification an unique solution for S_* can be obtained as,

$$\boxed{S_* = \frac{S_L \left(\rho\,u\right)_L - S_R \left(\rho\,u\right)_R + \left(\rho\,u^2 + p\right)_R - \left(\rho\,u^2 + p\right)_L}{S_L \rho_L - S_R \rho_R + \left(\rho_1 - \rho_2\right) \left[\left(C\,u\right)_R - \left(C\,u\right)_L\right]}}\,. \tag{A.39}$$

It is possible to derive the above equations manually, however a mathematical software like Maxima [133] is very useful in verifying and ascertaining the correctness of the obtained results. Finally, the only remaining unknowns, the values of S_L and S_R, are estimated as,

$$\boxed{S_L = \min\left(u_{CL} - a_L, u_{CR} - a_R\right)} \qquad \boxed{S_R = \max\left(u_{CL} + a_L, u_{CR} + a_R\right)}\,. \tag{A.40}$$

Appendix B

The Generalized Riemann Invariant Analysis (3D)

The generalized Riemann invariant (GRI) analysis is used to identify the quantities which change across the intermediate wave, corresponding to the repeating eigenvalues, u, as seen in equation (A.4). The three independent eigenvectors corresponding to the eigenvalues u are,

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \frac{1}{(\rho_1-\rho_2)\cdot u} \end{bmatrix}. \tag{B.1}$$

The analysis of the vector $[0,0,1,0,0]^T$ gives,

$$\frac{d\,(p/\beta)}{0} = \frac{d\,(\rho\,u)}{0} = \frac{d\,(\rho\,v)}{1} = \frac{d\,(\rho\,w)}{0} = \frac{dC}{0} \tag{B.2}$$

Therefore, across this wave the changes in p/β, $\rho\,u$, $\rho\,w$ and C are negligible compared to the changes in $\rho\,v$. Thus across this wave,

$$d\,(p/\beta) = 0 \quad \Longrightarrow \quad \boxed{dp = 0}\,. \tag{B.3}$$

$$\boxed{dC = 0}\,. \tag{B.4}$$

Also,

$$d\,(\rho\,u) = 0 \tag{B.5}$$

$$d\,((\rho_1 C + \rho_2\,(1-C))\;u) = 0 \tag{B.6}$$

$$d\,(((\rho_1-\rho_2)\,C + \rho_2)\;u) = 0 \tag{B.7}$$

$$((\rho_1-\rho_2)\,C + \rho_2)\;du + u\,d\,((\rho_1-\rho_2)\,C + \rho_2) = 0 \tag{B.8}$$

$$((\rho_1-\rho_2)\,C + \rho_2)\;du + u\,(\rho_1-\rho_2)\;dC = 0 \tag{B.9}$$

Since $dC = 0$ across this wave, and $((\rho_1 - \rho_2)\, C + \rho_2) \neq 0$, the above equation gives,

$$\boxed{du = 0}\,. \tag{B.10}$$

Also,

$$d\,(\rho\, w) = 0 \tag{B.11}$$

$$d\,(((\rho_1 - \rho_2)\; C + \rho_2)\; w) = 0 \tag{B.12}$$

$$((\rho_1 - \rho_2)\; C + \rho_2)\; dw + w\, d\,((\rho_1 - \rho_2)\; C + \rho_2) = 0 \tag{B.13}$$

$$((\rho_1 - \rho_2)\; C + \rho_2)\; dw + w\; (\rho_1 - \rho_2)\; dC = 0 \tag{B.14}$$

Since $dC = 0$ across this wave, and $((\rho_1 - \rho_2)\; C + \rho_2) \neq 0$, the above equation gives,

$$\boxed{dw = 0}\,. \tag{B.15}$$

Also,

$$d\,(\rho\, v) \neq 0 \tag{B.16}$$

$$((\rho_1 - \rho_2)\; C + \rho_2)\; dv + v\,(\rho_1 - \rho_2)\; dC \neq 0 \tag{B.17}$$

Since $dC = 0$ across this wave, the above equation gives,

$$\boxed{dv \neq 0}\,. \tag{B.18}$$

.....

The analysis of the second vector $[0, 0, 0, 1, 0]^T$ is similar to the analysis of first vector. The analysis of the second vector gives,

$$\frac{d\,(p/\beta)}{0} = \frac{d\,(\rho\, u)}{0} = \frac{d\,(\rho\, v)}{0} = \frac{d\,(\rho\, w)}{1} = \frac{dC}{0} \tag{B.19}$$

which implies,

$$d\,(p/\beta) = 0 \quad \Longrightarrow \quad \boxed{dp = 0}\,. \tag{B.20}$$

and

$$\boxed{dC = 0}\,. \tag{B.21}$$

Also,

$$d\,(\rho\, u) = 0 \tag{B.22}$$

which gives,

$$\boxed{du = 0}\,. \tag{B.23}$$

Also,

$$d\,(\rho\, v) = 0 \tag{B.24}$$

which gives,

$$\boxed{dv = 0}\,. \tag{B.25}$$

Also,

$$d\left(\rho\, w\right) \neq 0 \tag{B.26}$$

which gives

$$\boxed{dw \neq 0}\,. \tag{B.27}$$

.....

Across the third wave having an eigenvalue of u, the eigenvector is $[0, 1, 0, 0, 1/\left[\left(\rho_1 - \rho_2\right)\, u\right]]^T$, which implies,

$$d\left(p/\beta\right) = 0 \quad \Longrightarrow \quad \boxed{dp = 0}\,. \tag{B.28}$$

Also,

$$\frac{d\left(\rho\, u\right)}{1} = \left(\rho_1 - \rho_2\right)\, u\, dC \tag{B.29}$$

$$\left(\left(\rho_1 - \rho_2\right)\, C + \rho_2\right)\, du + u\left(\rho_1 - \rho_2\right)\, dC = \left(\rho_1 - \rho_2\right)\, u\, dC \tag{B.30}$$

$$\left(\left(\rho_1 - \rho_2\right) C + \rho_2\right)\, du = 0 \tag{B.31}$$

since $\left(\left(\rho_1 - \rho_2\right)\, C + \rho_2\right) \neq 0$,

$$\boxed{du = 0}\,. \tag{B.32}$$

Since $1/\left[\left(\rho_1 - \rho_2\right)\, u\right] \neq 0$; comparing the last and the first term of eigenvector, the changes in C are significant compared to changes in p. Therefore,

$$\boxed{dC \neq 0}\,. \tag{B.33}$$

As it has been established that $dv \neq 0$ and $dw \neq 0$ at the location where the intermediate waves are placed in space-time, from the analysis of the other two eigenvectors, there is no need to analyze the remaining two equalities. Thus, we can conclude from the above analysis that in the region where the intermediate waves are placed,

$$\boxed{dp = 0,\; du = 0,\; dv \neq 0,\; dw \neq 0,\; dC \neq 0}\,. \tag{B.34}$$

Therefore, for the Riemann solver considering the intermediate wave, the following conditions apply,

$$\boxed{p_{*L} = p_{*R},\; u_{*L} = u_{*R},\; v_{*L} \neq v_{*R},\; w_{*L} \neq w_{*R},\; C_{*L} \neq C_{*R}}\,. \tag{B.35}$$

These conditions are imposed by the HLLC-VOF Riemann solver to obtain a closed form solution for the unknown intermediate states.

Appendix C

HLL Riemann Solver

The HLL Riemann solver is an approximate solver for calculation of the convective flux in a system of hyperbolic partial differential equations (PDEs). This solver was introduced by Harten, Lax and van Leer [90] in 1983. The HLL solver uses a two wave model for evolution of the initial Riemann states. The unknown intermediate state can be obtained by using Rankine-Hugoniot jump conditions across the two waves. Unlike the other advanced Riemann solvers, the HLL solver only requires estimates of the minimum and maximum wave speeds of the hyperbolic PDE, and does not depend on any other physics of the problem. The derivation of the solver and the resulting mathematical expressions are presented in the upcoming section.

Derivation of the HLL Riemann solver

The temporal and the convective part of the x-split multi-dimensional governing equations can be written in a compact form as,

$$\frac{\partial \mathbf{U}}{\partial \tau} + \frac{\partial \mathbf{F}}{\partial x} = 0 \,, \tag{C.1}$$

where, $\mathbf{U}$ is called the conservative variable vector and $\mathbf{F}$ is called the flux vector. The above equation can be re-written using the chain-rule as,

$$\frac{\partial \mathbf{U}}{\partial \tau} + \frac{\partial \mathbf{F}}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial x} = 0 \,, \tag{C.2}$$

where, $\frac{\partial \mathbf{F}}{\partial \mathbf{U}}$ is called the Jacobian matrix. The system of governing equations is said to be hyperbolic if all the eigenvalues of the Jacobian matrix are real with linearly independent eigenvectors. A Riemann solver is used to compute the convective flux, $\mathbf{F}_f$, at the cell face in the finite volume method, by using the reconstructed values at the face, $\mathbf{U}_L$ and $\mathbf{U}_R$ (see section 3.4 for further details).

The HLL Riemann solver uses a two wave model to capture the evolution of the initial Riemann states as shown in Fig. C.1. The wave speeds of the two waves is denoted as S_L and S_R, which are also indicated in the Fig. C.1. In such a model the flux

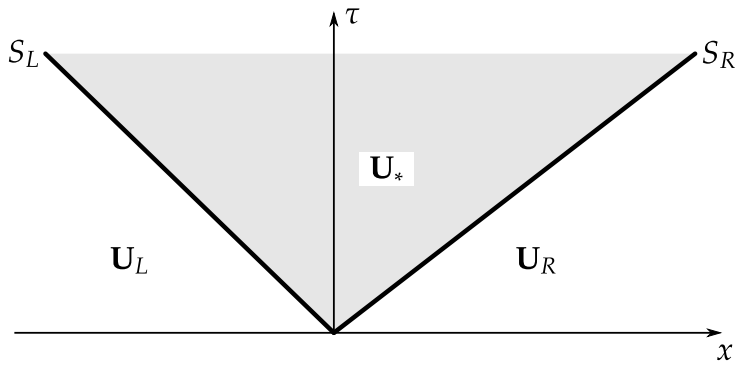


Fig. C.1: Schematic diagram depicting the wave structure of a Riemann solver with two distinct waves.

at the cell face can be written as,

$$\mathbf{F}_f^{\text{HLL}} = \begin{cases} \mathbf{F}_L & \text{if} \quad 0 \leq S_L \,, \\ \mathbf{F}_* & \text{if} \quad S_L \leq 0 \leq S_R \,, \\ \mathbf{F}_R & \text{if} \quad S_R \leq 0 \,, \end{cases} \tag{C.3}$$

where, the flux at the left and the right state can be readily evaluated as,

$$\mathbf{F}_L = \mathbf{F}\left(\mathbf{U}_L\right) \,, \quad \mathbf{F}_R = \mathbf{F}\left(\mathbf{U}_R\right) \,. \tag{C.4}$$

The value of wave speeds, S_L and S_R, can be estimated using the minimum and the maximum eigenvalues, $\lambda^{\min}$ and $\lambda^{\max}$, evaluated at the left and right states, as,

$$S_L = \min(\lambda_L^{\min}, \lambda_R^{\min}) \,, \quad S_R = \max(\lambda_L^{\max}, \lambda_R^{\max}) \,. \tag{C.5}$$

Therefore, the problem now reduces to the calculation of $\mathbf{F}_*$, so that the HLL flux, $\mathbf{F}_f^{\text{HLL}}$, can be fully evaluated using equation (C.3).

Using the Rankine-Hugoniot jump conditions across the two waves, results in the following two equations for $\mathbf{F}_*$,

$$\mathbf{F}_* = \mathbf{F}_L + S_L \, \left(\mathbf{U}_* - \mathbf{U}_L\right) \,, \tag{C.6}$$

$$\mathbf{F}_* = \mathbf{F}_R + S_R \, \left(\mathbf{U}_* - \mathbf{U}_R\right) \,. \tag{C.7}$$

These two equations can be solved to obtain the intermediate state, $\mathbf{U}_*$, to be,

$$\mathbf{U}_* = \frac{\mathbf{F}_L - \mathbf{F}_R + S_R \, \mathbf{U}_R - S_L \, \mathbf{U}_L}{S_R - S_L} \,. \tag{C.8}$$

The intermediate flux can be obtained by substituting the value of $\mathbf{U}_*$ in either equation (C.6) or equation (C.7) to obtain,

$$\mathbf{F}_* = \frac{S_R \, \mathbf{F}_L - S_L \, \mathbf{F}_R + S_L \, S_R \, \left(\mathbf{U}_R - \mathbf{U}_L\right)}{S_R - S_L} \,. \tag{C.9}$$

After substituting $\mathbf{F}_*$ in equation (C.3), the final expression for HLL Riemann solver can be obtained as,

$$\mathbf{F}_f^{\text{HLL}} = \begin{cases} \mathbf{F}_L & \text{if} \quad 0 \leq S_L \,, \\ \frac{S_R\,\mathbf{F}_L - S_L\,\mathbf{F}_R + S_L\,S_R\,(\mathbf{U}_R - \mathbf{U}_L)}{S_R - S_L} & \text{if} \quad S_L \leq 0 \leq S_R \,, \\ \mathbf{F}_R & \text{if} \quad S_R \leq 0 \,. \end{cases} \tag{C.10}$$

The HLL Riemann solver is derived for the x-split multi-dimensional governing equations. Nevertheless, the formulation can be applied to generic multi-dimensional problems by using the rotational invariance of the governing equations as discussed in section 3.4.2.

Appendix D

Interface Reconstruction for Triangular Cells

The newly developed interface reconstruction method is based on the marching cubes method, which extends easily to any polygonal or polyhedral shapes. The general algorithm for interface reconstruction and computation of volume fraction flux is explained in this thesis in section 5.1 and section 5.2 respectively, by using quadrilateral cell as an example. The proposed general algorithm requires iterations for interface reconstruction, and the same algorithm can be applied to triangular cells as well. Nevertheless, a simplified algorithm can be derived for triangular cells to obtain the required Riemann states using direct formulas. Here, such a link between the iterative method and direct formulas is established for triangular cells.

The first step towards interface reconstruction is to interpolate the volume fraction value at the nodes, which are denoted as C_n. The cell-averaged volume fraction, of the cell being reconstructed, is denoted as C_i. For a chosen value of volume fraction level C_k, the volume fraction of the cell can be computed as C_{ik} using the marching cubes method. An iterative approach can be used to compute the value of C_k such that $C_{ik} = C_i$, which results in the reconstructed interface. This process was explained in section 5.1. This iterative process can be eliminated in case of a triangle.

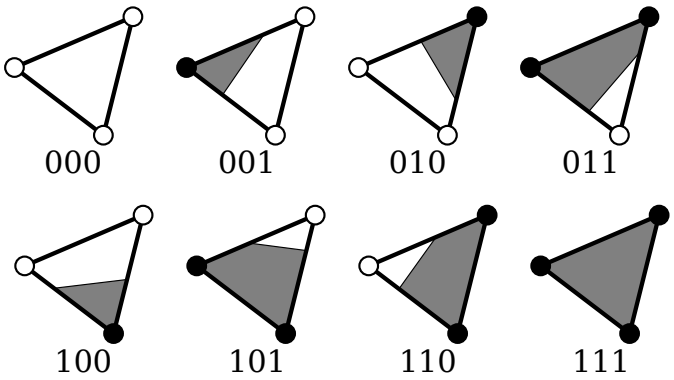


Fig. D.1: Possible configurations of an interface for a triangular cell

As a triangle has three nodes it can have eight possible interface configurations, i.e. $2^3 = 8$. The possible configurations of the interface for a triangular cell are dis-

played in Fig. D.1. The nodal values can be sorted as $C_n^{\min}$, C_n^{mid} and $C_n^{\max}$, where $C_n^{\min}$ is the minimum, C_n^{mid} is the intermediate and $C_n^{\max}$ is the maximum value of volume fraction, respectively. It can be observed that the cases 000, 001, 010 and 100 can be mapped to a computational coordinates (ξ, η) as shown in Fig. D.2. Similarly, the cases 011, 101, 110 and 111 can be mapped to computational coordinates (ξ, η) as shown in Fig. D.3. The transformation to the (ξ, η) coordinate system is such that the node corresponding to $C_n^{\min}$ maps to $(0, 0)$, node corresponding to $C_n^{\max}$ maps to $(1, 0)$ and the node corresponding to C_n^{mid} maps to $\left(C'_{\text{mid}}, 1\right)$ as shown in Fig. D.4. Such a mapping can be achieved by normalizing the volume fraction as,

$$C' = \frac{C - C_n^{\min}}{C_n^{\max} - C_n^{\min}} \implies C'_{\min} = 0\,; \quad C'_{\max} = 1\,; \quad C'_{\text{mid}} = \frac{C_n^{\text{mid}} - C_n^{\min}}{C_n^{\max} - C_n^{\min}}\,.$$

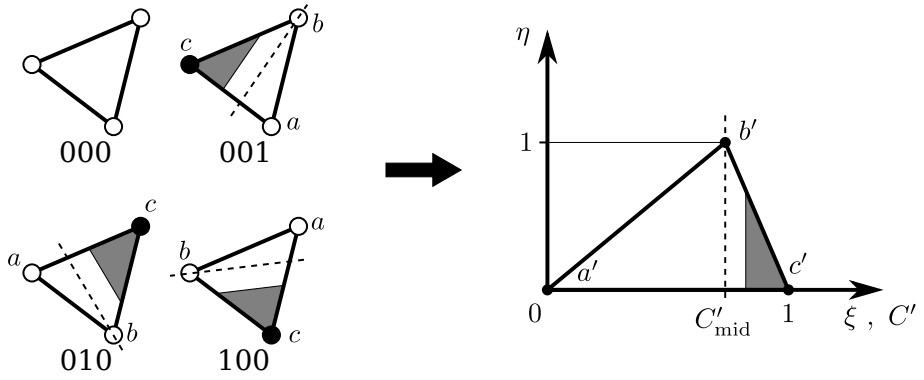


Fig. D.2: First case of interface reconstruction $0 \leq C_i \leq \left(1 - C'_{\textbf{mid}}\right)$

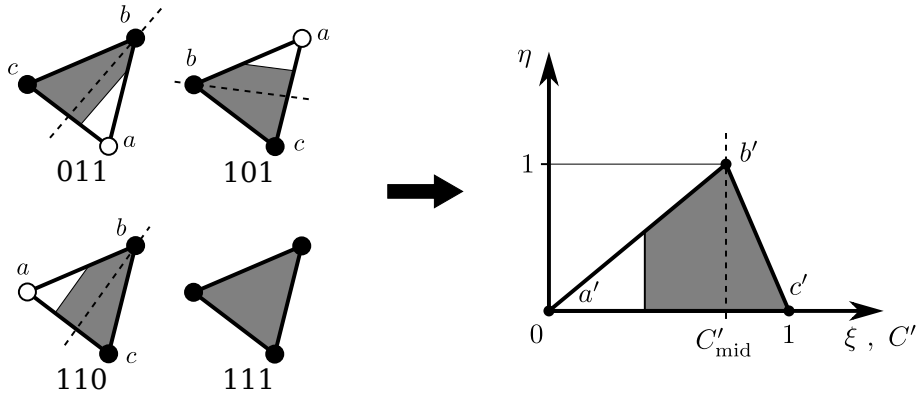


Fig. D.3: Second case of interface reconstruction $\left(1 - C'_{\textbf{mid}}\right) \leq C_i \leq 1$

In such a transformation, the interface remains vertical as shown in Fig. D.2 and Fig. D.3, as C' is on the horizontal axis. The reverse co-ordinate mapping from (ξ, η) to (x, y) can be easily obtained, however it is not necessary as the volume fraction, A_k / A_i, in the computational coordinates will remain same as in the physical coordinate system. This can be verified easily by using formula for area of triangle: area $=$ $0.5 \times$ base $\times$ height, with the base of the triangle on the dashed line in Fig. D.2 and Fig. D.3. Therefore, only two fundamental cases in the computational coordinate system are necessary to cover all the eight cases in physical coordinate system.

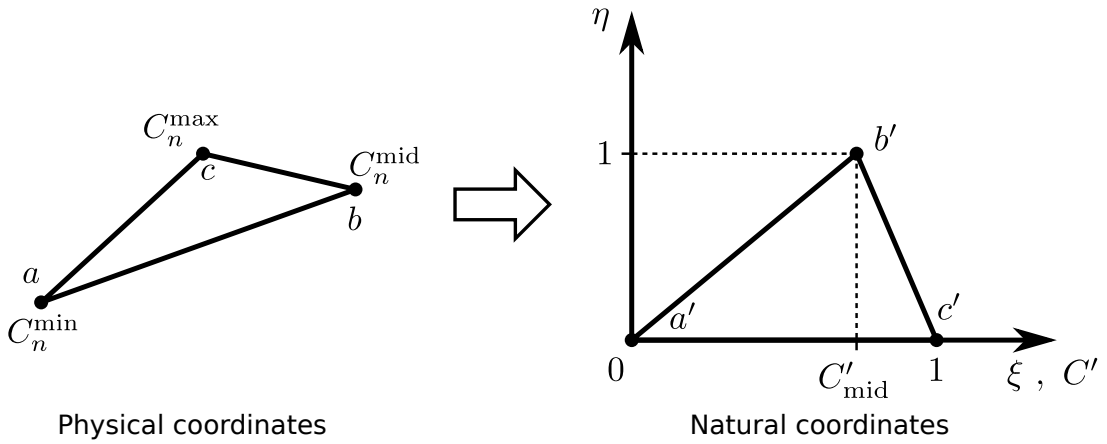


Fig. D.4: Schematic of transformation of a triangular cell

To obtain a closed form formula of interface reconstruction for first case (Fig. D.2), consider the schematic shown in Fig. D.5. This case will occur when $0 \leq C_i \leq \left(1 - C'_{\text{mid}}\right)$. The values of w and h satisfying the volume fraction, C_i, of the cell can be obtained to be,

$$w = \sqrt{C_i \ \left(1 - C'_{\text{mid}}\right)}\ ; \quad h = \frac{w}{1 - C'_{\text{mid}}}\ .$$

This results in the Riemann states as,

$$C_L^{ab} = 0;\ C_L^{bc} = h;\ C_L^{ac} = w\ .$$

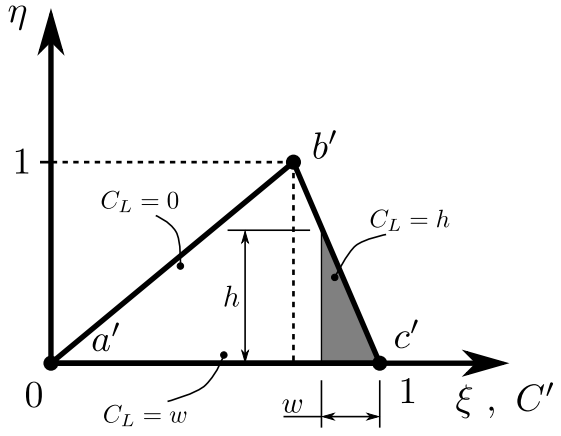


Fig. D.5: Schematic showing calculation of Riemann states for first case

To obtain a closed form formula of interface reconstruction for second case (Fig. D.3), consider the schematic shown in Fig. D.6. This case will occur when $\left(1 - C'_{\text{mid}}\right) \leq C_i \leq 1$. The values of w and h satisfying the volume fraction, C_i, of the cell can be obtained to be,

$$w = \sqrt{C'_{\text{mid}} \ \left(1 - C_i\right)}\ ; \quad h = \frac{w}{C'_{\text{mid}}}\ .$$

This results in the Riemann states as,

$$C_L^{ab} = 1 - h;\ C_L^{bc} = 1;\ C_L^{ac} = 1 - w\ .$$

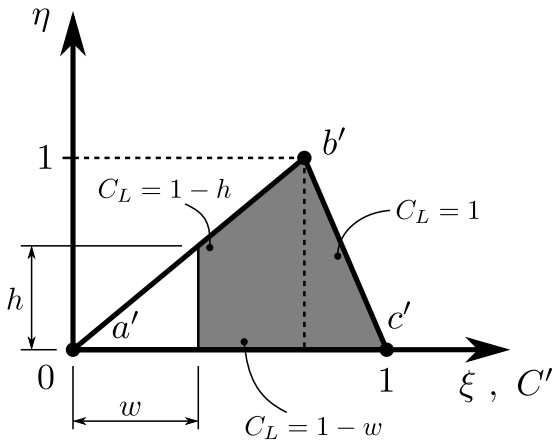


Fig. D.6: Schematic showing calculation of Riemann states for second case

These Riemann states can be used for accurate flux computation as explained in section 5.2.