Titre: Title:	Two Dimensional Compressible Flow Solver for Moving Geometries Using Immersed Boundary Method
Auteur: Author:	Md Sujaat Ali
Date:	2020
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Ali, M. S. (2020). Two Dimensional Compressible Flow Solver for Moving Geometries Using Immersed Boundary Method [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/5587/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/5587/
Directeurs de recherche: Advisors:	Jean-Yves Trépanier
Programme: Program:	Génie mécanique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Two Dimensional Compressible Flow Solver for Moving Geometries using Immersed Boundary Method

MD SUJAAT ALI
Département de génie mécanique

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie mécanique

Décembre 2020

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :
Two Dimensional Compressible Flow Solver for Moving Geometries using Immersed Boundary Method

présenté par **Md Sujaat ALI**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Sébastien LECLAIRE, président
Jean-Yves TRÉPANIER, membre et directeur de recherche
Guillaume PERNAUDAT, membre

DEDICATION

To my parents and grandparents...

ACKNOWLEDGEMENTS

I would like to thank my supervisor Professor Jean-Yves Trépanier for his guidance and advice throughout my master's program. Thank you for being available whenever I needed your help in the project.

I would like to recognize the valuable support and suggestions provided by Professor Ricardo Camarero, especially with respect to meshing.

I am grateful to General Electric Grid Solutions, France for supporting the project financially and also with various practical suggestions.

I am also thankful to Mr Renan de Holanda Sousa, Mr M. Ossman Awad and Mr. Yann Scheiffer for their moral and technical help during the entire duration of the project.

RÉSUMÉ

La méthode des frontières immergées (IB) a été mise en œuvre avec un certain succès pour différentes applications, y compris les géométries mobiles et stationnaires. Les travaux actuels portent sur l'extension de la méthode IB au traitement du déplacement des géométries pour les applications à écoulements compressible. En effet, pour les frontières mobiles, la mise en œuvre de l'approche IB comprend le changement du type de cellules, solide vers fluide, ce qui complexifie la méthodologie car il y a un manque dans l'historique de la solution numérique pour ces cellules. Afin de résoudre à cette problématique, deux approches sont disponibles dans la littérature. La première approche repose sur l'extrapolation des variables sur les cellules solides adjacentes (cellules fictives). La deuxième approche utilise la reconstruction de la solution toujours dans la région fluide. Un autre aspect de l'implémentation de la méthode IB est la représentation correcte d'une partie ou la totalité de la géométrie, en particulier lorsque l'échelle géométrique est inférieure à celle du maillage. Ce problème a été identifié et résolu pour les géométries stationnaires.

ABSTRACT

Immersed Boundary (IB) methods have been successfully implemented for different applications including moving as well as stationary geometries. Present work focuses on the implementation of IB method for moving geometries for compressible flow cases. IB implementation for moving boundaries includes the conversion of solid cells to fluid cells, which makes the problem a challenging one because of the abnormal values of the derivatives of pressure and velocity for the cells being converted. In order to solve this problem, mainly two different groups of methods are available. The first group of methods relies on extrapolating the variables on the adjacent solid cells(Ghost Cells) and the second group deals only with the interpolation in the fluid region. Another aspect of IB method implementation is the proper identification of the geometry or a part of the geometry, especially when the size of the geometry is smaller than the mesh size. This problem has been identified and solved for stationary geometries.

TABLE OF CONTENTS

DEDICATION iii

ACKNOWLEDGEMENTS iv

RÉSUMÉ v

ABSTRACT vi

TABLE OF CONTENTS vii

LIST OF TABLES x

LIST OF FIGURES xi

LIST OF SYMBOLS AND ACRONYMS xv

CHAPTER 1 INTRODUCTION 1
- 1.1 Computational Fluid Dynamics 1
- 1.2 Non-Body conforming Mesh 2
- 1.3 Motivation and Goals 3
- 1.4 Objectives 3
- 1.5 Thesis Structure 4

CHAPTER 2 LITERATURE REVIEW AND IB BASIS 5
- 2.1 IB method 5
- 2.2 IB method Basis 6
- 2.3 IB method Classification 7
- 2.4 Geometry representation 9
- 2.5 Tagging 9
- 2.6 Interface Cell Treatment 11
- 2.7 Interpolation Methodology 12

CHAPTER 3 IBM APPLICATION : DIFFUSION EQUATION 16
- 3.1 Problem definition 16
- 3.2 Mathematical model 17
  - 3.2.1 Mesh 17

3.2.2 Cell Numbering and Tagging 17
3.2.3 Discretisation 17
3.2.4 Matrix formation 19
3.3 Results 20
3.3.1 Concentric cylinders 20
3.4 Advantages of the present methodology 27
3.5 Conclusion 29

CHAPTER 4 EULER EQUATIONS 30
4.1 Introduction 30
4.2 Euler Equations : Overview 31
4.2.1 Classification of Partial Differential Equations 31
4.2.2 Eigenvectors 32
4.2.3 Euler Equations: Eigenvalue and Eigenvector 33
4.3 Riemann Problem: 1-D Euler 35
4.3.1 Conservation Laws and Godunov method 37
4.4 Roe Scheme 40
4.4.1 The Inter-cell Flux 41
4.4.2 Roe Scheme : Euler Equations 41
4.4.3 Entropy fix 42
4.5 Inviscid compressible Flow: Immersed Boundary Method 42
4.5.1 Meshing, Discretisation and Tagging 43
4.5.2 Stationary boundary 43
4.5.3 Moving boundaries 44
4.5.4 Moving boundaries: solid-to-fluid cell conversion 47
4.5.5 Field Extension Method 48
4.5.6 Two-level Interpolation 51
4.5.7 Conclusion 52

CHAPTER 5 2-D EULER EQUATIONS 53
5.1 Euler equations and Roe-Scheme implementation for 2-D 53
5.1.1 Tagging and Solver coupling 54
5.2 Interpolation scheme 56
5.3 2-D IBM with Roe scheme 57
5.4 2-D IBM with stationary objects 59
5.4.1 Boundary condition implementation 62
5.4.2 Results and discussions 63

5.5 Problems with shock-tube for corner cells 65
5.6 2-D IBM with moving boundaries 65
5.6.1 Moving Boundaries: Fluid-to-Solid conversion 66
5.6.2 Moving Boundaries: Solid-to-Fluid conversion 68
5.6.3 Advantages of methods implemented 74
5.7 Conclusion 75

CHAPTER 6 CONCLUSION 76
6.1 Summary of Works 76
6.2 Limitations 77
6.3 Future Research 77

REFERENCES 78

LIST OF TABLES

Table 5.1 List of Fluid Cells; without interface Cells 55
Table 5.2 List of Solid Cells; Inside Geometry 56
Table 5.3 List of Intercepted/Interface Cells 56

LIST OF FIGURES

Figure 1.1	Steps for performing CFD	1
Figure 1.2	Conventional CFD approach involves: a) Geometry creation b) Mesh generation conforming to the geometry c) Solving the discretised equations	2
Figure 1.3	Non-Body conforming CFD approach involves: a) Mesh generation b) creating object and inserting it in the mesh c) Solving the discretised equations to get the fluid flow variables	2
Figure 2.1	Non-conforming cartesian mesh for a solid-fluid domain in Immersed Boundary Method as described in [1]	6
Figure 2.2	Domain, Geometry and Cell classification	6
Figure 2.3	Ghost cell representation and interpolation stencil	8
Figure 2.4	Different types of cell modifications for the cut-cell approach [2]	9
Figure 2.5	Summary of different IB approaches	9
Figure 2.6	Geometry representation using a set of points	10
Figure 2.7	Cross-product depiction	10
Figure 2.8	Cross-product failure	11
Figure 2.9	Bi-linear interpolation	13
Figure 2.10	Bi-linear interpolation stencil	14
Figure 2.11	Demonstration of inefficiency of ghost cell method to capture the shaded portion of geometry	14
Figure 3.1	Geometry Definition	16
Figure 3.2	2-Dimensional mesh representation	17
Figure 3.3	Cell Numbering without and with objects	18
Figure 3.4	Numerical temperature distribution in x and y-directions : Concentric cylinders at a temperature differential	20
Figure 3.5	Comparison of Temperature[T] for numerical and analytical solutions for concentric cylinders in radial direction[R].	21
Figure 3.6	Convergence Analysis: Temperature	21
Figure 3.7	Calculating analytical temperature for concentric cylinder case. Here, $T_{Analytical}$ is the analytical temperature calculated at the cell centre and $T_{Boundary}$ is the temperature on the boundary	22

Figure 3.8 $\frac{dT}{dn}$ comparison: Analytical vs Numerical; sharp drop in the temperature gradient represent that the temperature at these points is not correct . . . 23
Figure 3.9 Flux comparison : analytical vs numerical with 10000 points on the geometry . . . 23
Figure 3.10 Incorrect normal calculation for the Concentric cylinders . . . 24
Figure 3.11 Normalised error in flux calculation: The circle has been represented with 500 equidistant points and the exact normal has been found analytically by the intersection of circle and the centre of the circle, passing through the intercface cell centre. The error has been normalised with respect to the analytical solution . . . 25
Figure 3.12 Convergence for dT/dn when the geometry is represented analytically with respect to L2 norm . . . 25
Figure 3.13 Normalised error: 500 X 500 Mesh with 1000 Points on the geometry for flux with normal calculation using linearisation. The normalisation has been done with respect to the analytical solution . . . 26
Figure 3.14 Normalised error: 200 X 200 Mesh with 1000 Points on the geometry for flux . . . 26
Figure 3.15 Convergence for dT/dn with 1000 points on the geometry and assuming that the geometry is represented by straight line in between two points. The error has been reported with respect to L2 norm. . . . 27
Figure 3.16 Convergence for 100 X 100 Mesh with variable number of points on the geometry for dT/dn. The error has been calculated using the L2 norm. 27
Figure 3.17 Schematic and numerical result for thin plate . . . 28
Figure 3.18 Schematic and numerical result for Airfoil . . . 28
Figure 4.1 Various regions based on Eigenvalues [3] . . . 36
Figure 4.2 A control volume representation . . . 37
Figure 4.3 Cell classification for one dimensional problem . . . 43
Figure 4.4 Shock tube schematic . . . 44
Figure 4.5 Discretized domain for 1-D IB implementation . . . 44
Figure 4.6 Pressure comparison: With and without IBM for the shock tube problem at $t = 0.025s$ . . . 45
Figure 4.7 Fluid-to-solid conversion . . . 45
Figure 4.8 Wall movement towards left schematic . . . 46
Figure 4.9 Discretized domain for wall movement towards left . . . 46
Figure 4.10 Wall movement towards left at two different time steps . . . 46

Figure 4.11 Pressure plot for wall movement towards left at two different time steps 47
Figure 4.12 Density profile for wall movement towards left at two different time steps 47
Figure 4.13 Solid-to-Fluid conversion 48
Figure 4.14 Wall movement towards right 49
Figure 4.15 Schematic for wall movement towards right 49
Figure 4.16 Solid-to-Fluid conversion numerical result: velocity profile for two different time steps 50
Figure 4.17 Solid-to-Fluid conversion numerical result: Pressure profile for two different time steps 50
Figure 4.18 Solid-to-Fluid conversion numerical result: Density profile for two different time steps 50
Figure 4.19 Solid-to-Fluid conversion numerical result: Density profile for two different time steps 52
Figure 5.1 Cell Numbering without and with object 55
Figure 5.2 Geometry representation using points 56
Figure 5.3 Bi-linear interpolation 57
Figure 5.4 Finite Volume discretisation of Cartesian domain. Typical computing cell $I_{i,j}$ with four intercell fluxes 58
Figure 5.5 Shock Tube case schematic 59
Figure 5.6 a) Pressure contour, b) pressure variation along x-direction at the middle of the shock tube, c) Density contour and d) density variation along x-direction at the middle of the shock tube 60
Figure 5.7 Inclined Shock Tube schematic with an inclination of 30^0 60
Figure 5.8 Problem with individual cell interpolation 61
Figure 5.9 Implicit interpolation demonstration 61
Figure 5.10 Bi-linear interpolation 63
Figure 5.11 Slip boundary condition implementation 63
Figure 5.12 Pressure convergence for inclined shock tube with implicit bi-linear interpolation for an angle of rotation of 140^0 64
Figure 5.13 Pressure, Velocity and Density contours for inclined shock tube for 500 X 500 background mesh with the angle of inclination of 140^0 64
Figure 5.14 Demonstration of corner cell problem for Bi-linear interpolation . . . 65
Figure 5.15 Wall movement towards left 66
Figure 5.16 Wall movement towards left at t = 0.08 seconds 67
Figure 5.17 Wall movement towards left at t = 2 seconds 67
Figure 5.18 Wall movement towards left: Tube immersed in domain 67

Figure 5.19 Wall movement towards right: cell status at a) t^k time step and b) t^{k+1} time step 69
Figure 5.20 Problem schematic: Wall movement towards right 69
Figure 5.21 The field-extension procedure: (a) identification of the pseudo-fluid points/ghost cells where the solution will be extended; (b) possible extrapolation stencils [4] 70
Figure 5.22 Wall movement towards right: Whole domain at t = 0.08 seconds . . 71
Figure 5.23 Wall movement towards right: Whole domain at t = 6 seconds 71
Figure 5.24 Wall movement towards right: Tube immersed in domain 71
Figure 5.25 Generalised Two-level interpolation for two dimension 72
Figure 5.26 Solid-to-Fluid: Whole domain at t = 0.08 seconds using two-level interpolation 73
Figure 5.27 Solid-to-Fluid: Whole domain at t = 0.6 seconds using two-level interpolation 73
Figure 5.28 Solid-to-Fluid: Tube immersed in domain using Two-level interpolation 73
Figure 5.29 Demonstration of inefficiency of ghost cell method to capture the shaded portion of geometry 74
Figure 5.30 Problem with explicit interpolation as used in [4] 75

LIST OF SYMBOLS AND ACRONYMS

IBM	Immersed Boundary Method
IB	Immersed Boundary
CFD	Computational Fluid Dynamics
GC	Ghost Cell
GP	Geometric Point
P	Pressure
U	velocity in x-direction
D	Density
V	velocity in Y-direction
FP	Fluid Point

CHAPTER 1 INTRODUCTION

1.1 Computational Fluid Dynamics

With the increase in the computational capability, the use of numerical simulation is increasing. Research and innovation aim at increasing the efficiency and accuracy of numerical methods for various applications. Computational fluid dynamics deals with this process applied to the field of fluid dynamics. Traditionally, CFD consists of multiple steps, illustrated in Fig. 1.1.

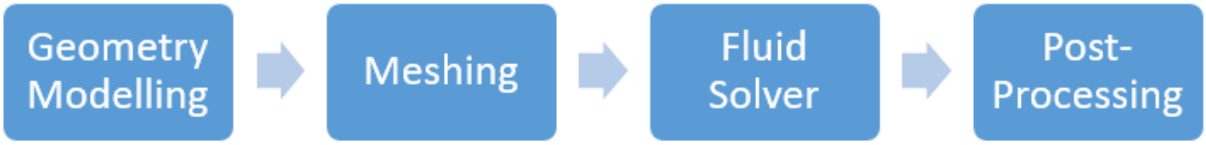
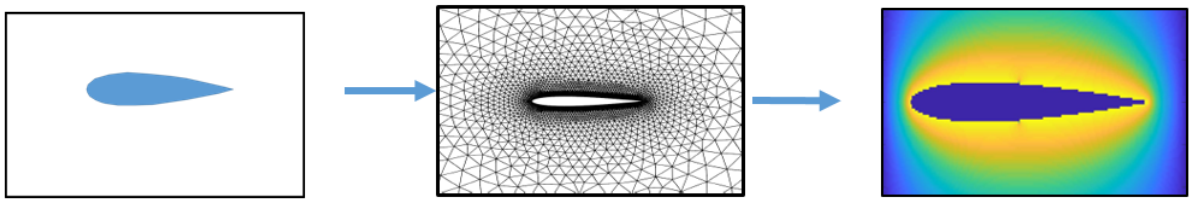


Figure 1.1 Steps for performing CFD

The first step includes geometry creation which can be as simple as a circle, for two-dimensional problems, or as complex as an aircraft model for three-dimensional problems. The second step involves creating a mesh which is a difficult and time-consuming task. Meshing is basically dividing the domain into smaller cells or elements where the discretised mathematical equations are applied to get the required flow field variables. The quality of the solution depends critically on the quality of the mesh generated. The basic requirement of meshing is that the grid generated should have proper resolution so as to capture the geometric and physical features. The discretised equations on the elements are assembled into a global algebraic system of equations, the solution for which provides the fluid flow variables. Finally, post-processing is used to extract the solution in the form of graphs, contours etc, for analysis purposes.

Mesh can be structured, with an implicit (i, j) numbering, or unstructured with an explicit connectivity table. The choice of mesh often depends on the shape of the geometry and also on the physics of the problem being solved. Moreover, the mesh generated traditionally conforms to the geometry and the boundaries involved. Based on these meshes, different solution methods, namely Finite Volume, Finite Difference or Finite Element, can be used. The conventional approach of the CFD methodology is illustrated in Fig. 1.2.



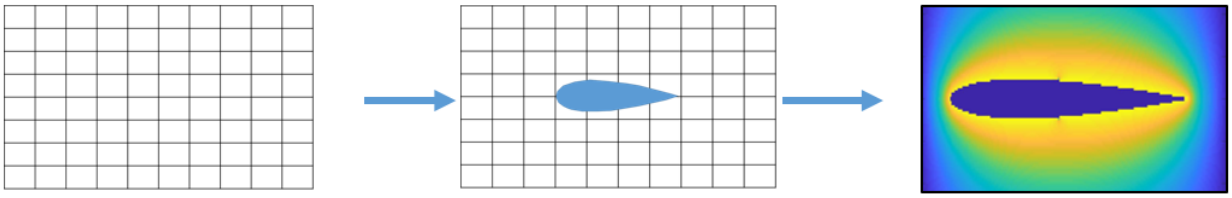
(a) (b) (c)

Figure 1.2 Conventional CFD approach involves: a) Geometry creation b) Mesh generation conforming to the geometry c) Solving the discretised equations

1.2 Non-Body conforming Mesh

In recent years, research has been undertaken to shorten or simplify the mesh generation process. The concept of performing numerical simulation by creating non-body conforming mesh was first introduced by [5] and since then, has been extended for different flow regimes and applications. One of the major advantages of these methods is the simplification of mesh generation step as it is based on a cartesian mesh. These methods are broadly termed as Immersed Boundary(IB) methods.

An important advantage of IB method over body-conforming mesh generation approach is for the moving boundaries. For body-conforming mesh, the mesh has to be updated for each time step whereas for IB method this is not required as body movement is independent of the grid. The CFD procedure in context of IB methodology is depicted in Fig. 1.3.



(a) (b) (c)

Figure 1.3 Non-Body conforming CFD approach involves: a) Mesh generation b) creating object and inserting it in the mesh c) Solving the discretised equations to get the fluid flow variables

1.3 Motivation and Goals

IB methods have been implemented for a variety of fluid flow problems, but mostly for incompressible flows. The main motivation for this work is to study the application to compressible flow cases. One of the major problems for the IB implementation is that the geometry or the part of the geometry is not captured correctly. This can be a predominant problem especially for the cases in which coarser mesh exists compared to one part of the geometry.

Geometry movement, translation or rotation or a combination of both, also pose a challenge especially when there is conversion from solid cell to a fluid cell. This requires a special treatment for the cells located near the boundary of the geometry.

The goal for this work is to investigate the various problems that arise in the IB implementation using simple problems. Specifically, to deal with the movement of the geometry for IB problems for compressible flows.

1.4 Objectives

In order to achieve the aforesaid goals, a number of objectives have been identified which include,

1. Ghost cell vs one-sided interpolation : Since the mesh used for IB methods does not conform to the geometry, proper identification of shape, size and location of the geometry is required which is implemented using state of the art methodologies. It might be possible that the geometry, or a part of the geometry, is not identified correctly, especially when the geometry is thinner than the mesh size. Developing an IB methodology suitable for thin geometries for compressible flow cases is one of the objectives of present work. The two available methods, namely Ghost Cell and one-sided interpolation, will be used and compared.

2. Explore approach for moving boundaries: Boundary motion is addressed and investigated such that, if required, a new methodology can be found out for its proper implementation for compressible flows problems.

3. Reconstruction Scheme selection : A reconstruction, also called interpolation, scheme is required to implement the boundary condition for non-body conforming mesh. Different reconstruction schemes are applied, highlighting how the choice and way of implementation affect the solution.

1.5 Thesis Structure

The thesis has been divided into six chapters. The first chapter consists of the introduction to the context and motivation to carry out the research, goals and specific objectives.

The second chapter deals with the literature review with respect to the implementation of the IB approach. It describes IB classification, selection of different IB methods for compressible flow problems.

The third chapter describes the IB implementation for the diffusion equation. This provides the general framework for the implementation of IB approach and to identify the critical steps encountered. The problem for properly capturing the geometry is assessed with respect to stationary geometries for Laplace's equation. The ideal condition is to be able to obtain the solution for a given geometry without any restriction on its thickness with respect to the mesh size.

The fourth chapter extends the approach to compressible flow problems. For the sake of simplicity and ease of implementation, at first the problems are solved for the 1-D Euler equations. A shock tube case with a stationary wall at one end, not aligned to the grid, is analysed with respect to the shock reflection from the wall. The next phase deals with the wall movements in both directions, which include cell conversion from solid-to-fluid as well as from fluid-to-solid. These cases in 1-D form the basis for 2-D problems.

The fifth chapter investigates the 2-D implementation which includes a 2-D immersed shock tube in a domain with stationary, as well as with moving walls. All possible cell conversions, solid-to-fluid as well as from fluid-to-solid are investigated to include all the cases analysed in 1-D.

Finally, the last chapter deals with the challenges and limitations with respect to present IB implementation along with the scope for future work.

CHAPTER 2 LITERATURE REVIEW AND IB BASIS

2.1 IB method

IB method was introduced by [5] which focused on cardiovascular flow simulations. The entire simulation was performed using a fixed cartesian grid independent of the geometry of the heart wall as well as its motion. The boundary effect was mimicked using appropriate source terms in the discretised equations. Since then, IB method has been modified and implemented for various applications.

IB methods have been predominantly very successful with respect to incompressible flows as presented in [6] , [7], [8] and [9]. The typical problem encountered in the IB method, is the mass conservation problem which has been explained in detail in [10] and recently, this problem has been tackled to a great extent by [11]. The lack of research for compressible flows using IBM is one of the motivation for the present work. For compressible flow, the mass conservation problem has been addressed in [12], [13] and [14]. While, other problems also exist with respect to capturing the boundary layer flows with IB methods, these can be tackled to some extent, but not completely, as described in [15], [16].

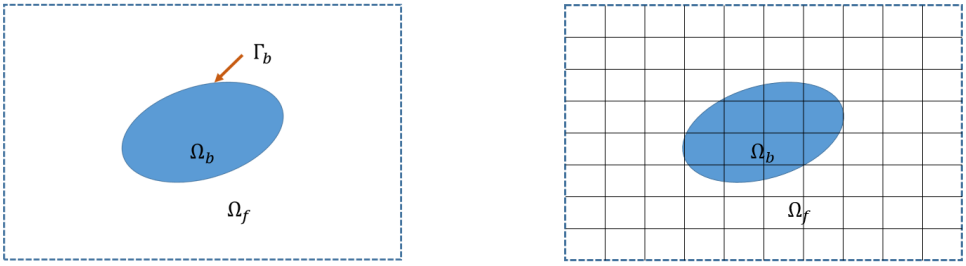
In [17], turbulent flows simulations are performed in straight pipes with large range of roughness topographies with second order accuracy. There are multiple implementation of IB approach for turbulent flow problems. Some of them includes work by [18] which focuses on complex turbulent flows by coupling IBM with Large-Eddy simulations, [19] performed LES-IBM with coarse mesh using wall functions.

The recent applications of IB methods for complex problems includes the simulation of pressure driven turbulent channel in the presence of 10,000 neutrally buoyant spherical particles in [20] with no-slip boundary condition on each of the particles. No other numerical method is reported to address the complexities and calculation efforts involved in resolving thousand spheres (particulates) and implementing no-slip boundary condition on each of those particulates [21].

IB implementation for moving geometries is also very challenging as evident from the work of [4], which describes in detail about the non-availability of the information while marching in time whenever geometry motion is encountered. The solution provided for this problem mostly exist taking into account the Ghost Cells which is evident from the work of [4], [22] and [23].

2.2 IB method Basis

In the traditional CFD approach, the mesh has to conform to the boundary if not, a special treatment of the boundary condition is required. As shown in Fig. 2.1, Ω_b is the volume of the geometry, a solid sub-domain, and Ω_f is the volume of the fluid, a fluid sub-domain, separated by a solid boundary, Γ_b. Fig. 2.1(b), shows the object with a non-conforming grid. The fact that the mesh is non-conforming to the geometry raises the problem for the boundary condition implementation. The improper boundary condition implementation in IB methods leads to inaccurate calculation of the fluid flow variables.

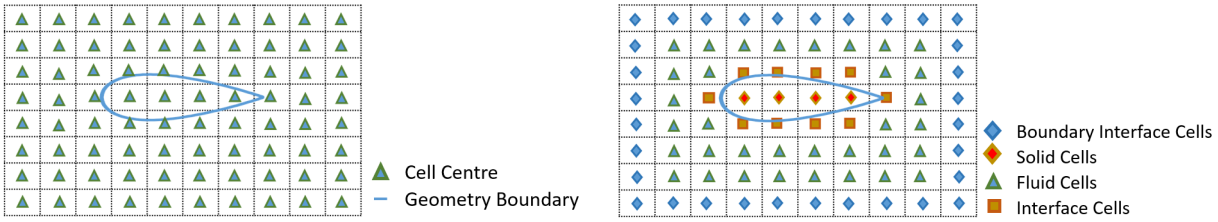


(a) Solid object in the fluid domain

(b) Solid object in fluid domain in a cartesian mesh

Figure 2.1 Non-conforming cartesian mesh for a solid-fluid domain in Immersed Boundary Method as described in [1]

Geometry and topology are represented in Fig. 2.2(a), with respect to IB methodology. The mesh used is cartesian and the object or geometry is introduced into this domain. The next step is to mark different kinds of cells which is depicted in Fig. 2.2(b). The process of cell identification or marking is described in the subsequent sections. It has to be understood that these processes are intrinsic to the current methodology.



(a) Domain and Geometry

(b) Cell centres classification

Figure 2.2 Domain, Geometry and Cell classification

The marked cells in Fig. 2.2(b) include,

1. **Fluid Cells** : Cells with complete computational stencils

2. **Solid Cells**: Cells whose centre lie inside the solid boundary

3. **Intercepted/interface cells**: Fluid cells with incomplete stencils, marked with orange squares and

4. **Boundary interface cells**: Fluid cells with incomplete stencils, for outer boundary of the domain.

Cells with complete stencils are treated as usual for a finite volume scheme. Those lying inside the geometry are not used and their contributions to the neighbouring cells are zero. The cell centers which are intercepted are to be treated uniquely with the implementation of an interpolation scheme for these cells.

2.3 IB method Classification

As per [1] IB methods can be broadly classified as Continuous Forcing Approach and Discrete Forcing Approach (Sharp interface method). The fundamental difference between these two different approaches lies in their discretizations for boundary condition implementation. For the Continuous forcing methods, a forcing function is added to the governing equations such that the physical conditions are obtained at the boundary and then the governing equations are discretised. For the Discrete forcing method, at first the equations are discretized on the entire domain and for the Immersed Boundary cells, a modified set of equations is applied based on the boundary conditions.

The Continuous forcing approach can be implemented for both the elastic as well as the rigid boundaries. Zhu et. al [24] used continuous forcing approach for simulating the interaction of two flapping filaments in a flowing soap film. Fauci et al [25] analysed the aquatic animal locomotion using the continuous forcing approach. For rigid body problems, it has been implemented for the backward facing step in [26].

Sharp Interface approach can be implemented in two different ways. The first one is the Indirect Boundary Condition Implementation and the second is called Direct Boundary Condition Implementation. The Direct Boundary condition implementation is the desirable candidate for high Reynolds number flows as it requires local accuracy because of the boundary layer formation [27]. The Indirect Boundary Condition implementation does not provide the required local accuracy as it uses non-user dependent forcing terms for boundary condition implementation. The sharp interface approach has proved to be the best candidate for simulating compressible flow cases as evident from the work presented in [28], [29] and [30].

Again, the sharp interface method can be implemented in multiple ways which includes Ghost Cell and Cut-Cell approaches. Ghost cells are the cells inside the solid region with at least one neighbour inside the fluid as illustrated in Fig. 2.3.

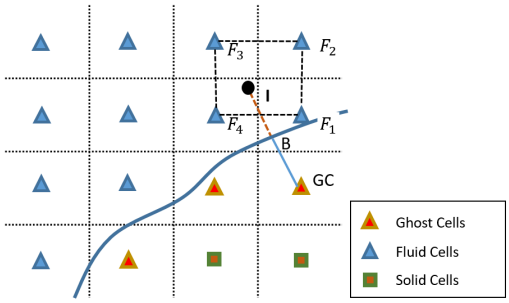


Figure 2.3 Ghost cell representation and interpolation stencil

In Fig. 2.3, point B is on the solid boundary such that BGC is normal to the surface of the geometry. Once these points are available any interpolation scheme can be applied to reconstruct the fluid variables. As per [31], using points F_1, F_2, F_3 and F_4, a bi-linear interpolation can be performed for point I and depending on the boundary condition, this can be used to find value at ghost cell GC. For a Dirichlet boundary condition, linear extrapolation can be performed between points I, B and GC and for a Neumann boundary condition of type $\frac{\partial \phi}{\partial n} = 0$, the values obtained for I can be used directly at GC.

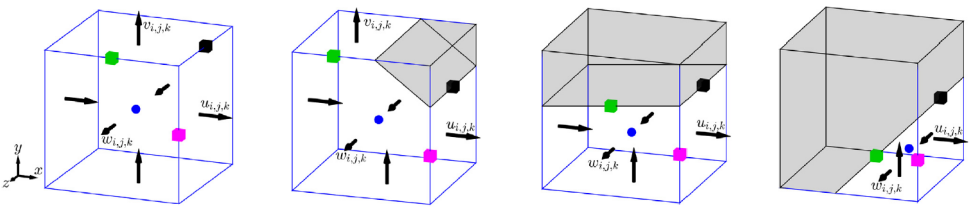
The Cut-cell method uses a control volume which is entirely in the fluid domain but is an irregular polygon near the boundary [2]. Such polygons result from the modified cells containing the boundary. These include the following for a boundary cell:

1. If the cell center lies inside solid geometry, the remaining parts of the cell is merged to the nearby cell

2. If the cell center lies outside the geometry, the solid portion of the cell is removed from the boundary cell

This whole procedure yields an approximate representation of the geometry with respect to the cartesian mesh with control volumes of trapezoidal shape near the boundary as described in detail in [2] using 3-D cells. This is presented in Fig. 2.4.

A summary of the different IB approaches has been presented in Fig. 2.5.



(a) A Cartesian Cell (b) Pentagonal Cell (c) Trapezoidal Cell (d) Triangular Cell

Figure 2.4 Different types of cell modifications for the cut-cell approach [2]

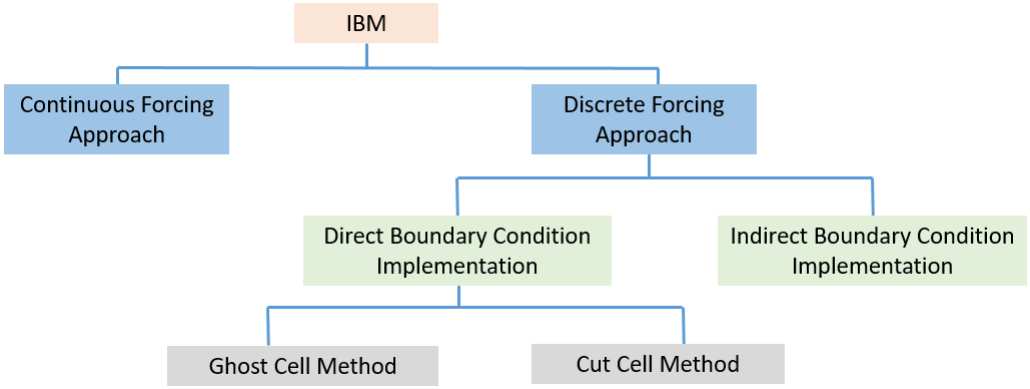


Figure 2.5 Summary of different IB approaches

2.4 Geometry representation

Solid objects inside the domain can be represented in the following ways:

1. Discrete representation: using a set of points or edges
2. Analytical representation : using polynomial function, spline, ellipse etc

One of the simplest ways to represent the geometry is using set of points, which has been implemented in this work. Although, wherever possible geometry should be represented analytically as any other method induce error with respect to proper identification of the geometry. However, in many cases analytical representation is not possible, therefore a robust method for geometry representation using points has been suggested that reduce the error for geometry identification to a great extent which is used for subsequent IB implementation.

2.5 Tagging

Once the geometry representation is done, the next step is to identify different kinds of cells based on their location with respect to the boundary. This process is called "Tagging" and is carried out using Ray-tracing [32–35] or cross-product. For the current work, the

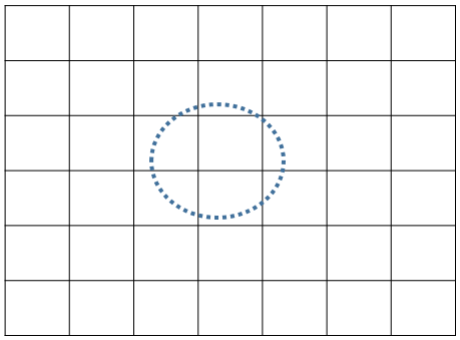


Figure 2.6 Geometry representation using a set of points

cross-product method is used because of its ease of implementation for simple geometries like circles, rectangles and NACA 0012 airfoil. This is illustrated in Fig. 2.7.

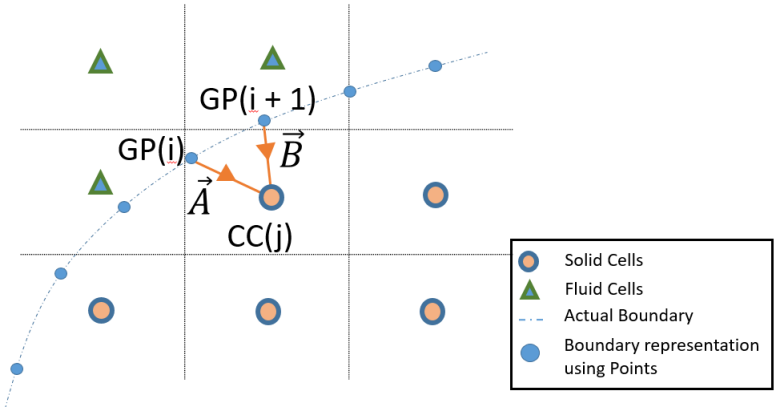


Figure 2.7 Cross-product depiction

The Tagging process is described in more detail in Algorithm 1. In short, while moving in the clockwise direction along the geometric points, if for a particular cell centre $|\vec{A}X\vec{B}| < 0$ for all the points on the geometry, then the cell centre lies inside the geometry, otherwise it lies outside. Once, inside and outside has been determined one needs to find the intercepted cells. For that, one needs to look into the cells which lie outside and for which the stencils are not complete.

Once all kinds of cells have been identified, the tagging part is complete. However, if the object is moving, tagging has to be repeated at each time step [6]. Once it is done for one time step, the cell identification part can be effectively implemented by doing a localized search rather than searching in the entire domain using the CFL criteria which restricts the movement of the geometry up to the length of one cell in one time step [18].

The current tagging approach is very efficient for simple geometry like cylinders or squares or

airfoil (NACA 0012), but can fail in some circumstances. One such case has been presented in Fig. 2.8 where, if cross-product is used to find whether the cell center is lying inside or outside, the method is likely to fail. $|\vec{A}X\vec{B}| > 0$ for this combination.

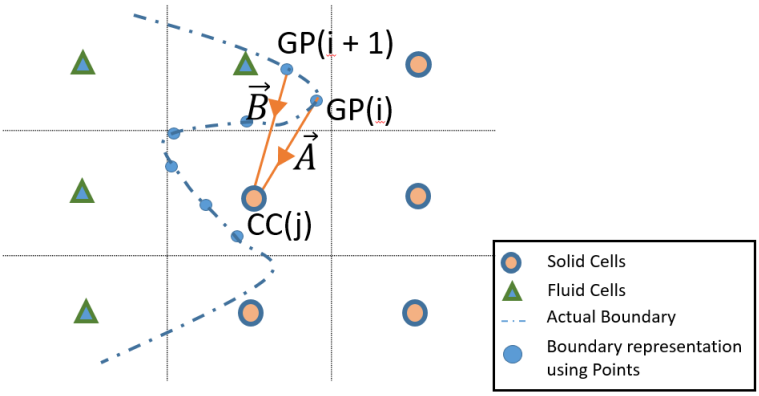


Figure 2.8 Cross-product failure

2.6 Interface Cell Treatment

The control volume scheme based on the integral form of the fluid equation is applied to the cells whose stencils are complete. The cell centres which are inside the geometry, are discarded as these cells do not contribute to the final numerical solution. The intercepted/interface cell centres require special treatment thus, the numerical solver has to be modified to incorporate the boundary condition information.

Different kinds of interpolation schemes have been implemented for solving the problem of interface cells which includes bi-linear interpolation [36–40], radial basis function [41–43], Least-square method and moving Least-square [44–46], Inverse Distance weighting [32,33,47] etc.

Algorithm 1 Explicit Tagging Algorithm

The Cartesian grid is represented by $N_{Cell} = n_x \times n_y$ cells, and the immersed boundary is described by $N_{Boundary}$.
for $i \in \{1, ..., N_{Cell}\}$ **do**
    for $j \in \{1, ..., N_{Boundary} - 1\}$ **do**
        Calculate the two vectors A_{ij} and B_{ij} that connect the boundary terminals $(j, j+1)$ to the center of cell (i), as shown in Fig. 2.7.
        Perform a cross product of the two vectors, such that $Cross_{ij} = A_{ij} \times B_{ij}$.
        if $Cross_{ij} < 0$ **then**
            Cell(i) is a solid cell and belong to $ALLSOLID\_cell$ list.
        else
            Cell(i) is fluid cell and belong to $ALLFLUID\_cell$ list.
        end if
    end for
end for
Create two lists, $ALLSOLID\_cell$ of $N_{ALLSolid}$ points, and $ALLFLUID\_cell$ list of $N_{ALLFluid}$, such that $N_{Cell} = N_{ALLSolid} + N_{ALLFluid}$.
for $i \in \{1, ..., N_{ALLSolid}\}$ **do**
    Get the number neighbour cells to the Cell(i) into $N_{Solid\_Neighbours}$
    if $N_{Solid\_Neighbours} = 4$ **then**
        Cell(i) is a solid cell and belong to $SOLID\_cell$ list.
    else
        Cell(i) is a solid interface cell and belong to $Gohst\_Cell$ list.
    end if
end for
Create two lists of solid cells, $SOLID\_cell$ and $Gohst\_Cell$ lists.
for $j \in \{1, ..., N_{ALLFluid}\}$ **do**
    Get the number neighbour cells to the Cell(j) into $N_{Fluid\_Neighbours}$
    if $N_{Fluid\_Neighbours} = 4$ **then**
        Cell(i) is a Fluid cell and belong to $Fluid\_cell$ list.
    else
        Cell(i) is a Fluid Interface Cell and belong to $Fluid\_Interface\_Cell$ list.
    end if
end for

2.7 Interpolation Methodology

For the present scheme, bi-linear interpolation has been selected because of its ease of implementation and simplicity. Moreover, obtaining a higher order interpolation was not the target, a bi-linear interpolation scheme would be a suitable candidate.

In general, the implementation of the interpolation can be done in different ways. [48] uses

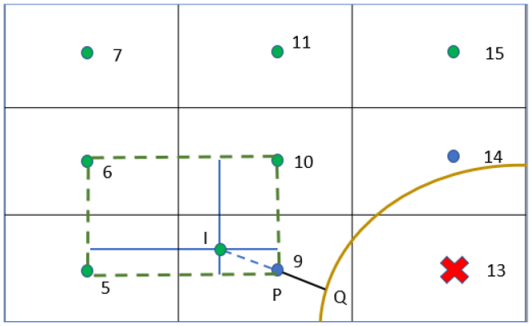


Figure 2.9 Bi-linear interpolation

the ghost cell for interpolation which uses the nearest point in the solid region to complete the 4-point bounding box, including the boundary points. A sample bi-linear interpolation using the ghost cell method has been described in Fig. 2.3.

Whereas, contrary to the Ghost Cell implementation, the second class of methods do not consider any contribution from the cells which lie inside the solid domain [4]. These rely on considering only the fluid domain cells. This can be implemented as shown in Fig. 2.9 where, point $P(9)$ is the intercepted cell and interpolation method has to be implemented for calculating the fluid flow values. For implementing a bi-Linear interpolation, a point Q has to be found on the geometry such that the line PQ is the normal to the geometry. This is really important for the flux calculation or for the Newman Boundary condition implementation.

The second step is to find the point I such that the cell-centre point P is equidistant from both point I and Q. From this, the reconstruction for the unknown values at P can be found as follows:

$$\phi_P = (\phi_I + \phi_Q)/2 \tag{2.1}$$

From bi-linear interpolation,

$$\phi_I = a_6\phi_6 + a_{10}\phi_{10} + a_9\phi_9 + a_5\phi_{15} \tag{2.2}$$

The final equation for ϕ_P:

$$(2 - a_P)\phi_P = \phi_Q + a_6\phi_6 + a_5\phi_5 + a_{10}\phi_{10} \tag{2.3}$$

The coefficients a_6, a_5, a_{10} can be found by solving Eq. 2.3. In order to solve equations similar to Eq. 2.2, one needs to solve the system of equations described in Eq. 2.4.

$$\begin{bmatrix} a_5 \\ a_9 \\ a_{10} \\ a_6 \end{bmatrix} = \left(\begin{bmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_2 & y_1 & x_2y_1 \\ 1 & x_2 & y_2 & x_2y_2 \\ 1 & x_1 & y_2 & x_1y_2 \end{bmatrix}^{-1} \right)^T \begin{bmatrix} 1 \\ x \\ y \\ xy \end{bmatrix} \tag{2.4}$$

where x_1, x_2, y_1, y_2, x and y are represented in Fig. 2.10 which can be compared to Fig. 2.9.

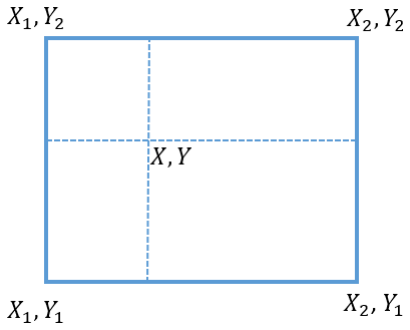


Figure 2.10 Bi-linear interpolation stencil

Although the results from the Ghost cell interpolation and the one-sided interpolation look very similar, the main difference arises when the size of the geometry or a part of the geometry is smaller than the mesh size. In those locations, the Ghost cell interpolation cannot be implemented, whereas the one-sided interpolation can be implemented easily. This problem with respect to the Ghost cell implementation is illustrated in Fig. 2.11. Here, the shaded region can not be captured using the Ghost cell method but the one-sided interpolation can easily capture such geometries. This will be more clear in the subsequent chapters where IB method is used for different applications, especially a heat transfer case for thin plate.

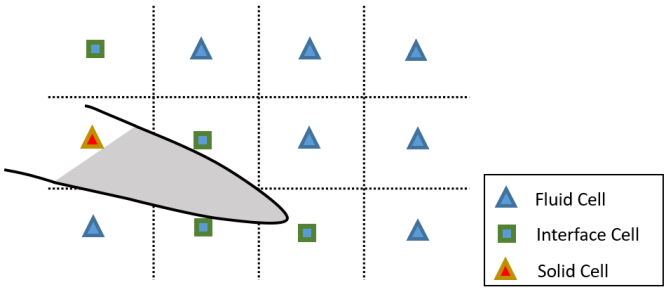


Figure 2.11 Demonstration of inefficiency of ghost cell method to capture the shaded portion of geometry

The IB methods have been implemented for various heat transfer and fluid flow problems. Since a basis for IB method has already been established, next chapters will deal with the IB applications for various problems. At first, a problem involving diffusion equations will be solved and then problems involving Euler equations will be solved.

CHAPTER 3 IBM APPLICATION : DIFFUSION EQUATION

The IB method can be implemented for a variety of fluid mechanics and heat transfer problems. One of the simplest equations to understand the IB implementation is the diffusion(Laplace's) equation. It can help to assess the advantages and the drawbacks of IB method and also in ensuring an efficient implementation for the more complex cases in the following chapters.

Previous studies of similar problem includes work by [49] with respect to the proper boundary condition implementation for convective and diffusive cases. Other work on heat transfer problems, notably conjugate heat transfer, has been studied in [50].

3.1 Problem definition

The schematic of the problem is described in Fig. 3.1, consisting of a domain bounded by two concentric circles with Dirichlet boundary condition, the inner circle has a temperature T_1 and the outer circle has temperature T_2. The heat conduction for this problem is governed by Eq. 3.1, which is the heat diffusion equation with a constant thermal conductivity, $(K = 1)$, and without any source term.

$$\frac{d^2T}{dx^2} + \frac{d^2T}{dy^2} = 0 \tag{3.1}$$

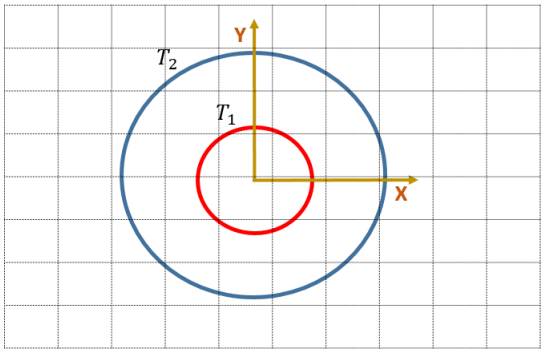


Figure 3.1 Geometry Definition

The analytical solution, as presented in [51], for this case is given by Eqs. 3.2 and 3.3:

$$\text{Heat Transfer[watts]} \quad = \quad q_r = 2\pi k(T_1 - T_2)/ln(r_2/r_1) \tag{3.2}$$

$$\text{Temperature[K]} \;=\; T(r) = \frac{(T_1 - T_2)}{ln(r_2/r_1)} ln(\frac{r}{r_2}) + T_2 \tag{3.3}$$

where, q_r is the heat transfer rate, $T(r)$ is the temperature at radius r, T_1 and T_2 are the temperature for the two circles located at r_1 and r_2 as presented in Fig. 3.1.

3.2 Mathematical model

3.2.1 Mesh

Cartesian mesh used is shown in Fig. 3.2. Point P is a cell centre and the corresponding East, West, North and South cell centres are represented as E, W, N and S respectively. The corresponding faces for the control volume, bounded by dotted lines, have been represented by e, w, n and s. The control volume is defined by the volume occupied by the discrete domain characterised by $\Delta x * \Delta y * \Delta z$ for a 3-D mesh. For two dimensional case Δz is assumed to be unity.

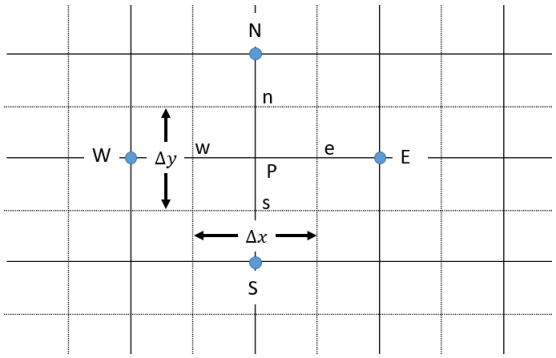


Figure 3.2 2-Dimensional mesh representation

3.2.2 Cell Numbering and Tagging

The cell numbering, shown in Fig. 3.3(a), is maintained throughout the IB implementation process. Once cells are numbered and the geometry is defined, it is required to classify the cells into interface, solid and fluid based on the tagging methodology presented in Sec. 2.5.

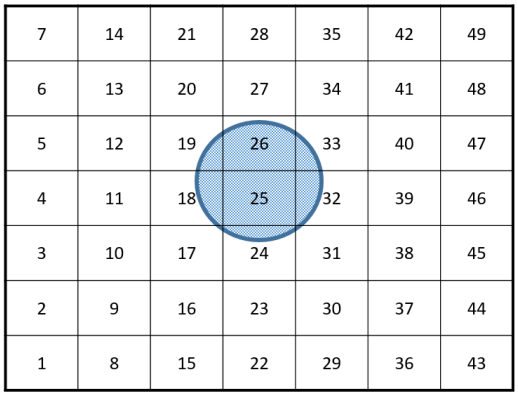
3.2.3 Discretisation

The Laplace equation,

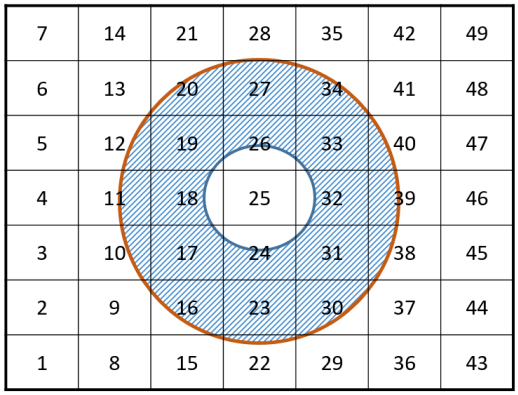
$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0 \tag{3.4}$$

7	14	21	28	35	42	49
6	13	20	27	34	41	48
5	12	19	26	33	40	47
4	11	18	25	32	39	46
3	10	17	24	31	38	45
2	9	16	23	30	37	44
1	8	15	22	29	36	43

(a) Cell Numbering



(b) Cell numbering with object



(c) Cell numbering : Concentric cylinder case

Figure 3.3 Cell Numbering without and with objects

is discretised on a cartesian mesh using a finite volume approach as presented in [52]. Once Eq. 3.4 is integrated over a control volume giving;

$$\int_{\Delta V} \frac{\partial^2 \phi}{\partial x^2} dx.dy + \int_{\Delta V} \frac{\partial^2 \phi}{\partial y^2} dx.dy = 0 \tag{3.5}$$

Assuming the areas for east, west, north and south faces to be A_e, A_w, A_n and A_s respectively such that $A_e = A_w = \Delta y$ and $A_n = A_s = \Delta x$, as illustrated in Fig. 3.2. Here Δv is the control volume which is already defined in the sub-section 3.2.1. The volume integral can be converted to the surface integral using Green's theorem which states that the sum of fluid out-flowing from a volume is equal to the total outflow summed about an enclosing area. As explained in [52] . After applying Green's theorem, converting volume integral to boundary integral, Eq. 3.5 finally becomes,

$$[A_e(\frac{\partial\phi}{\partial x})_e - A_w(\frac{\partial\phi}{\partial x})_w] + [A_n(\frac{\partial\phi}{\partial x})_n - A_s(\frac{\partial\phi}{\partial x})_s] = 0 \tag{3.6}$$

The numerically discretised fluxes through the faces for the control volume can be written as;

$$\text{East Face flux } = A_e\frac{\phi_E - \phi_P}{\delta_{PE}} \tag{3.7a}$$

$$\text{West Face flux } = A_w\frac{\phi_P - \phi_W}{\delta_{WP}} \tag{3.7b}$$

$$\text{South Face flux } = A_s\frac{\phi_P - \phi_S}{\delta_{PS}} \tag{3.7c}$$

$$\text{North Face flux } = A_n\frac{\phi_N - \phi_P}{\delta_{NP}} \tag{3.7d}$$

where, δ_{PE} is the distance between points P and E in Fig. 3.2. Inserting fluxes expressed in Eqs. 3.7 into Eq. 3.6,

$$(\frac{A_e}{\delta_{PE}} + \frac{A_w}{\delta_{WP}} + \frac{A_n}{\delta_{NP}} + \frac{A_s}{\delta_{PS}})\phi_P = (\frac{A_e}{\delta_{PE}})\phi_E + (\frac{A_w}{\delta_{PW}})\phi_W + (\frac{A_n}{\delta_{NP}})\phi_n + (\frac{A_s}{\delta_{PS}})\phi_s \tag{3.8}$$

which can be written in the form,

$$a_p\phi_P = a_e\phi_E + a_w\phi_W + a_n\phi_N + a_s\phi_S \tag{3.9}$$

where,

$$a_p = (\frac{A_e}{\delta_{PE}} + \frac{A_w}{\delta_{WP}} + \frac{A_n}{\delta_{NP}} + \frac{A_s}{\delta_{PS}});\ a_e = (\frac{A_e}{\delta_{PE}});\ a_w = (\frac{A_w}{\delta_{PW}});\ a_n = (\frac{A_n}{\delta_{NP}});\ \text{and } a_s = (\frac{A_s}{\delta_{PS}}) \tag{3.10}$$

3.2.4 Matrix formation

Eq 3.8 is applied to all cells in the fluid region, except for the interface cells. The steady state problem is solved by solving a system of equations, $Ax = b$, where A is a matrix of size $(NM * NM)$ and N and M are number of cells in X and Y directions, respectively. Matrix A consists of the coefficients a_p, a_s, a_n, a_e and a_w which have been defined in Eq. 3.10.

For all the fluid cells, coefficients obtained with respect to each cell have to be assembled in matrix A. For example, the coefficients for cell number 9, in Fig. 3.3(c), can be assembled in

the matrix as described below;

$$A = \begin{array}{c} \begin{matrix} 1 & 2 & ... & 8 & 9 & 10 & ... & 16 & ... \end{matrix} \\ \begin{bmatrix} a_{11} & a_{12} & ... & a_{18} & a_{19} & a_{10} & ... & a_{16} & ... \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & a_w & ... & a_s & a_p & a_n & ... & a_e & ... \end{bmatrix} \begin{matrix} 1 \\ \vdots \\ 9 \end{matrix} \end{array} \tag{3.11}$$

In Fig. 3.3(b), cell number 18 is an interface cell, therefore the coefficients are to be found using the bi-linear interpolation described by Eq. 2.3 in Sec. 2.7. Here again, the coefficients are inserted in the matrix corresponding to row number 18 with the columns corresponding to the cell number used for the interpolation.

3.3 Results

3.3.1 Concentric cylinders

The numerical solution is shown in Fig. 3.4 for a 200 x 200 cells in the mesh:

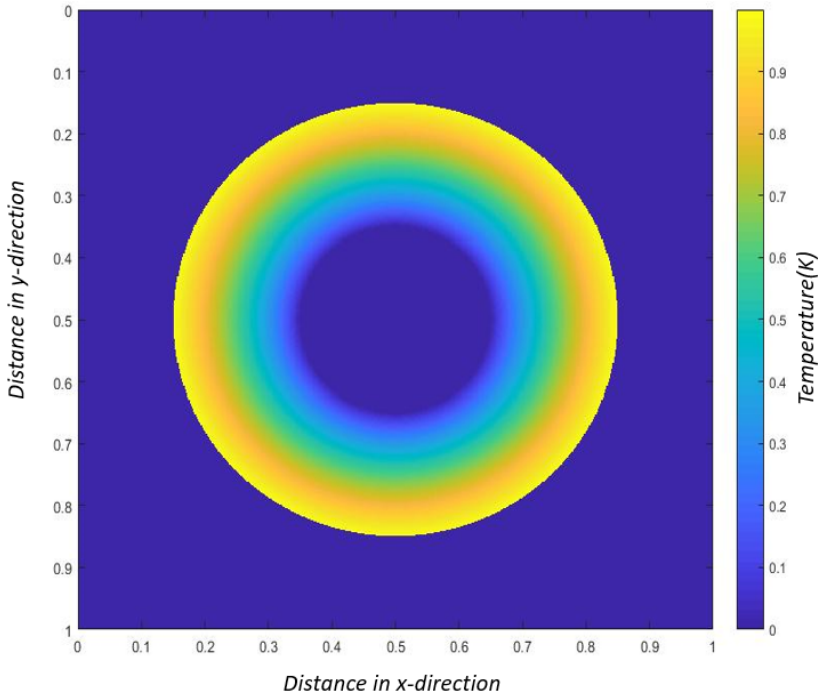


Figure 3.4 Numerical temperature distribution in x and y-directions : Concentric cylinders at a temperature differential

The variation of temperature in the radial direction is compared with the analytical solution in Fig. 3.5. The results match with the analytical solution and are accurate.

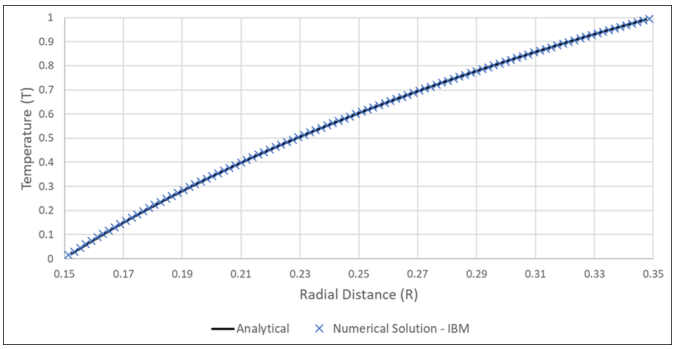


Figure 3.5 Comparison of Temperature[T] for numerical and analytical solutions for concentric cylinders in radial direction[R].

A convergence analysis is performed to verify the order of convergence that is obtained by the present numerical algorithm. This consists of finding the error(norm) (L2 or L1), against the number of points. In present case, convergence analysis has been done using the L2 norm and is given by Eq. 3.12:

$$Error = L2 = \sum_{i=1}^{M} \sqrt{(T_{i,analytical} - T_{i,numerical})^2}/Nx \tag{3.12}$$

where Nx is the total number points considered for the numerical calculation.

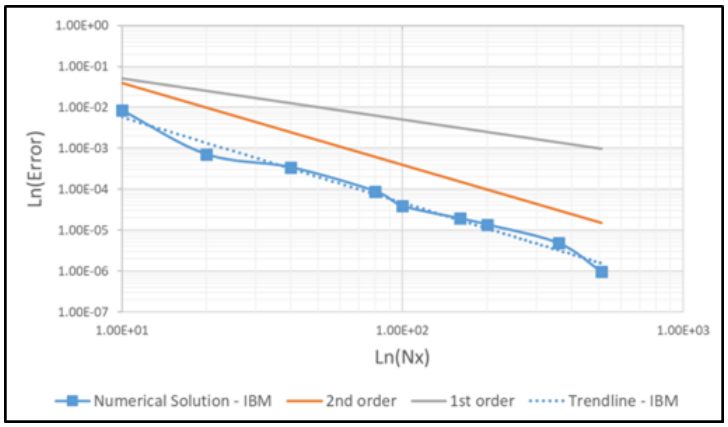


Figure 3.6 Convergence Analysis: Temperature

The convergence analysis for the numerical solution using L2 norm is shown in Fig. 3.6. The

order of convergence obtained is of second order, which is expected because second order discretisation of the governing equation has been used.

Another thing to be looked for the similar convergence analysis for the change in temperature in the normal direction, which is basically the flux calculation. Calculation of the analytical fluxes have been modified such that the numerical and analytical fluxes are compared on the same basis. For calculating the analytical fluxes, Eq. 3.13 is used.

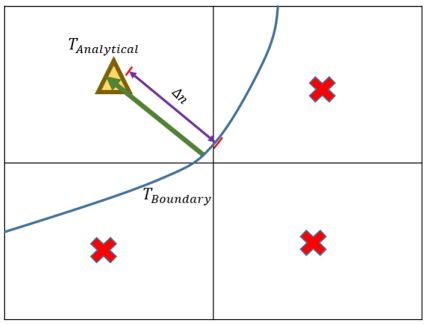


Figure 3.7 Calculating analytical temperature for concentric cylinder case. Here, $T_{Analytical}$ is the analytical temperature calculated at the cell centre and $T_{Boundary}$ is the temperature on the boundary

In Fig. 3.7,

$$\left(\frac{dT}{dn}\right)_{Analytical} = \frac{T_{Boundary} - T_{Analytical}}{\Delta n} \tag{3.13}$$

Similarly, for the numerical solution,

$$\left(\frac{dT}{dn}\right)_{Numerical} = \frac{T_{Boundary} - T_{Numerical}}{\Delta n} \tag{3.14}$$

where, $T_{Numerical}$ are interface cell centers where solution is obtained using the interpolation function.

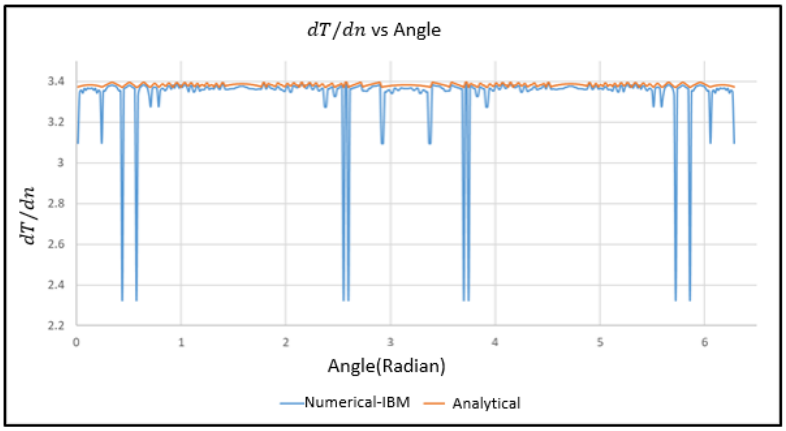


Figure 3.8 $\frac{dT}{dn}$ comparison: Analytical vs Numerical; sharp drop in the temperature gradient represent that the temperature at these points is not correct

Once these calculations are done, the flux comparison is presented in Fig. 3.8. It can be seen that there are sharp drops in the flux values at few points near the boundary. This fluctuation happens at specific locations and their occurrences are symmetric to the centre-line with respect to angle (3.14 radian) and the magnitudes are also same with respect to the corresponding symmetric point. The possible reasons for these sharp fluctuations can be attributed to one of the following:

1. Bi-linear interpolation is poorly implemented
2. Poor representation of the geometry
3. Problem with finding the normal direction for the bi-linear interpolation.

In order to find out the actual reason for this, a numerical solution with 10,000 equidistant points on the circle is considered. This eliminates the first two points from the list as represented in the Fig. 3.9.

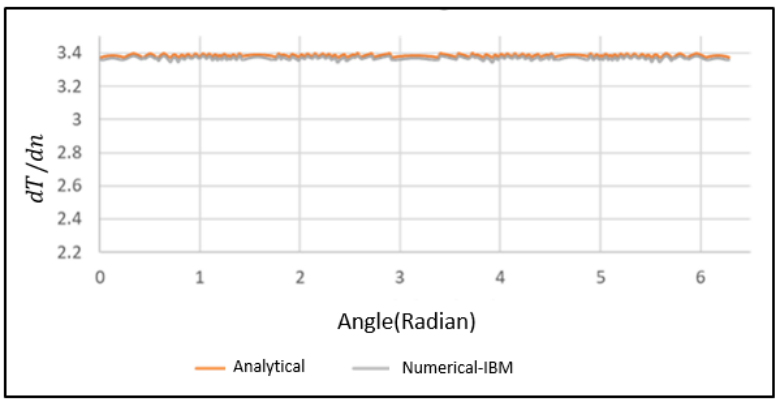


Figure 3.9 Flux comparison : analytical vs numerical with 10000 points on the geometry

This shows that:

1. The Bi-linear interpolation has been implemented correctly
2. The representation of the geometry using points is correct

The only thing that has to be corrected is the normal computation. In reality, the initial way of calculating normal is not correct as represented in Fig. 3.10. The main reason is that, it was believed that the line segment joining interface cell centre to the nearest point on the solid boundary will produce a normal. However, the accuracy for this is highly dependent on the number of points considered for representing the geometry and this works out only when sufficiently large number of points are used for the representation of the geometry.

Although the results obtained from geometry representation using 10000 points is good, it is not a very efficient way of geometry representation. So, the solution is to use fewer points for the geometry representation and at the same time, ensuring the flux calculated is correct.

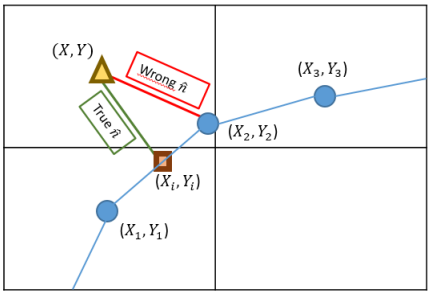


Figure 3.10 Incorrect normal calculation for the Concentric cylinders

One solution for this is to represent the geometry using fewer points and in order to find the correct normal treat the section between two points as line segment and then find out the normal from the cell centre to this line segment. This approach is a better geometric representation of the section between the points.

Another way of solving this problem is to represent the geometry analytically. The results from analytical representation of the geometry will be used to compare with the case in which the geometry is represented as a line between two geometric points. This will help in understanding and analysing the error that is encountered.

In order to represent the geometry analytically, at first the equation of the circle is considered which is given by Eq. 3.15 where, (a, a) is the center of the circle and r is the radius of the circle.

$$(x - a)^2 + (y - a)^2 = r^2 \tag{3.15}$$

with this and the centre of the circle, the line that passes from the interface cells and the centre of the circle is actually the normal to the circle at its boundary. In order to find the boundary point of the circle, we have the equation of the line and the equation of the circle which indeed will give two points. The one nearest to the intercepted point will be the point on the circle and from which the image point "I" is found and then the bi-linear interpolation can be done afterwards. The flux variation along the outer circle is presented in the Fig. 3.11. The corresponding convergence graph is presented in Fig. 3.12. It can be seen that the order of convergence obtained is close to the desired order of 2.

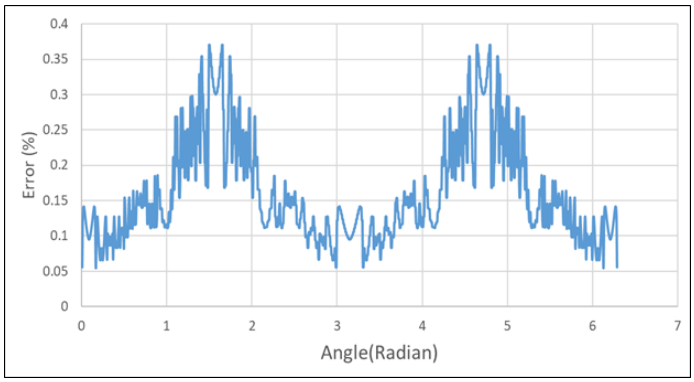


Figure 3.11 Normalised error in flux calculation: The circle has been represented with 500 equidistant points and the exact normal has been found analytically by the intersection of circle and the centre of the circle, passing through the intercface cell centre. The error has been normalised with respect to the analytical solution

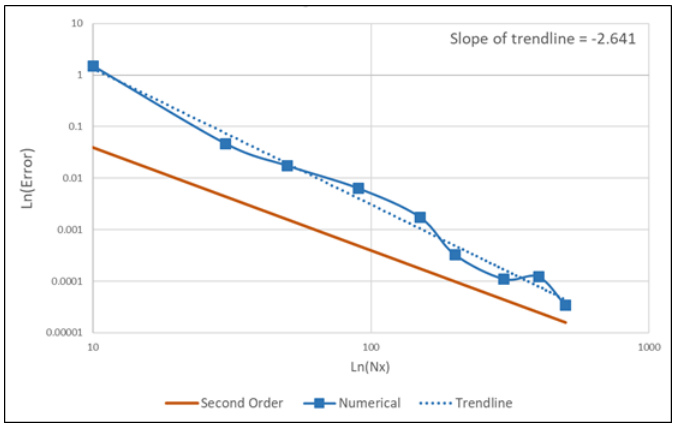


Figure 3.12 Convergence for dT/dn when the geometry is represented analytically with respect to L2 norm

Moving to the representation of the geometry with a finite number of points, the linearisation

for finding the normal is done as represented in Fig. 3.10. The case for 500 X 500 mesh size is repeated with linearisation and is presented in Fig. 3.13.

It can be seen that the percentage of the normalized error remains the same but there is an increase in the 'noise' in the values which have insignificant contribution to the result. This can be identified by comparing Fig. 3.13 and Fig. 3.11. Moreover, if the grid size is increased, the shape of the graph remains the same. The same has been depicted in the Fig. 3.14. However the error increases, which is expected, with the increase in the grid size. The observed order of convergence is two, which is as expected and is presented in Fig. 3.15.

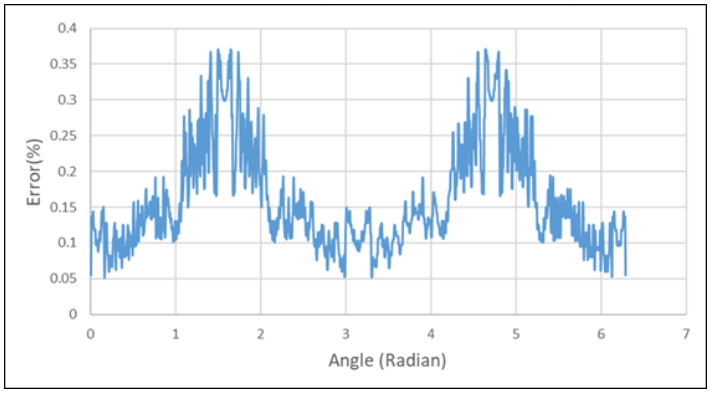


Figure 3.13 Normalised error: 500 X 500 Mesh with 1000 Points on the geometry for flux with normal calculation using linearisation. The normalisation has been done with respect to the analytical solution

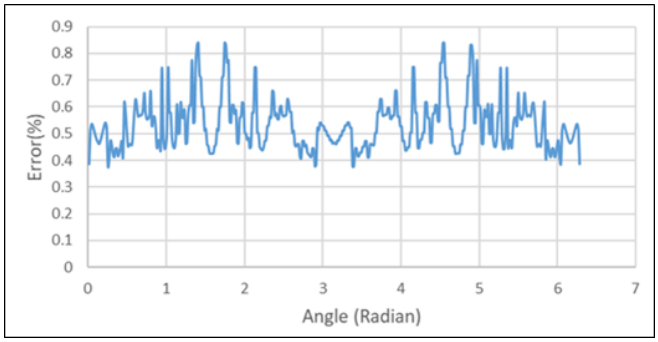


Figure 3.14 Normalised error: 200 X 200 Mesh with 1000 Points on the geometry for flux

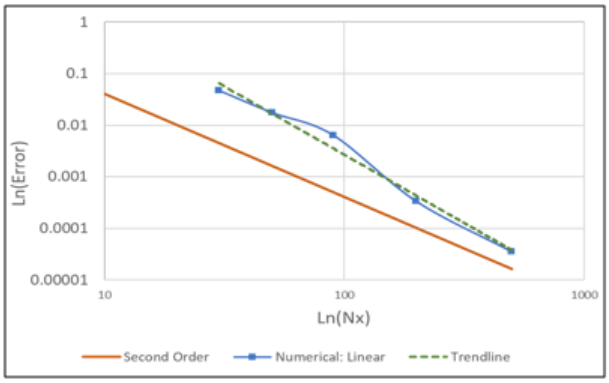


Figure 3.15 Convergence for dT/dn with 1000 points on the geometry and assuming that the geometry is represented by straight line in between two points. The error has been reported with respect to L2 norm.

One important aspect is the dependency of the points on the geometry to that of the results obtained. For this a convergence/error analysis has been done using 100 X 100 mesh points for the entire domain for variable number of points on the geometry. The result is presented in Fig. 3.16. It has been found that the solution depends on the number of points on the geometry up to 1000 points. After that the change in the numerical values obtained is insignificant.

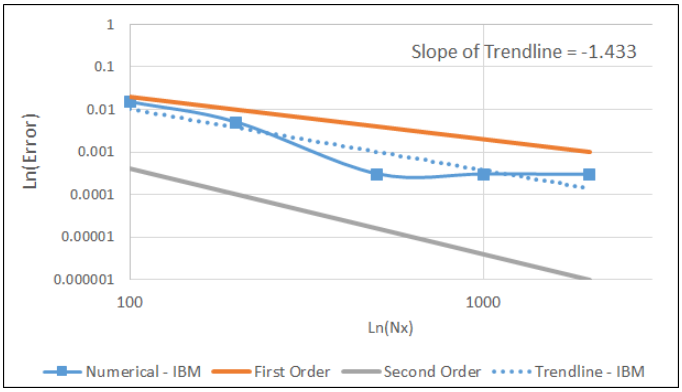


Figure 3.16 Convergence for 100 X 100 Mesh with variable number of points on the geometry for dT/dn. The error has been calculated using the L2 norm.

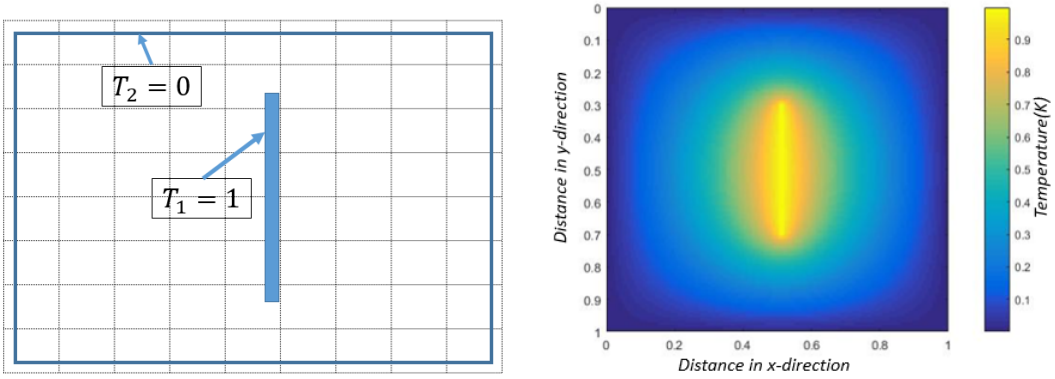
3.4 Advantages of the present methodology

As discussed in the previous sections, the presented method does not consider any contribution from the points that lie inside the solid domain. This is in contrast to the Ghost cell

approach, which can fail once the geometry or a part of the geometry is smaller than the mesh size.

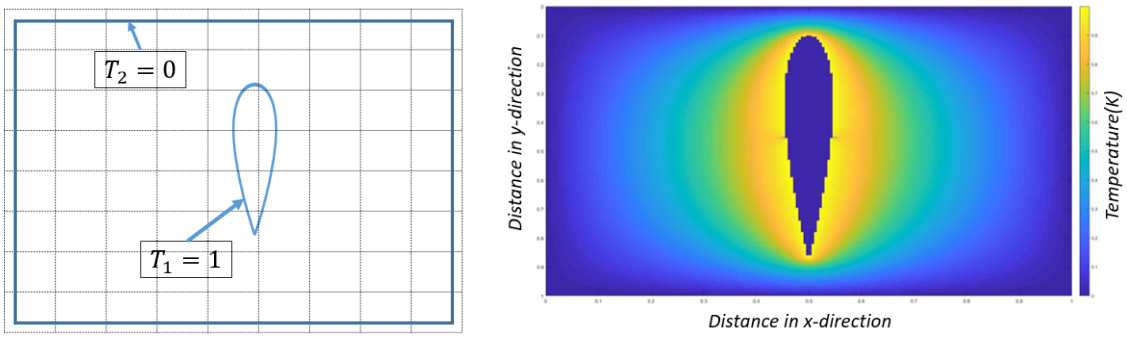
This can be understood by considering a problem statement in which the object is indeed smaller than the mesh size. Here, a thin plate, as shown in Fig. 3.17(a), has a temperature of $T_1 = 1$ and an outer boundary with a temperature of $T_2 = 0$. It should be noted that no cell centre lies inside the thin plate. Thus, the interpolation using the Ghost cell cannot be implemented for this case, while the interpolation methodology presented above can be easily implemented for this case.

The result for the numerical simulation is shown in the Fig. 3.17(b). The temperature varies between 0 and 1. No analytical solution exists for this case. Similarly, the schematic for an airfoil(NACA0012) shaped geometry, in Fig. 3.18(a), and its corresponding numerical result is shown in Fig. 3.18(b).



(a) Thin plate Schematic (b) Thin Plate numerical results

Figure 3.17 Schematic and numerical result for thin plate



(a) Airfoil geometry Schematic (b) Airfoil numerical results

Figure 3.18 Schematic and numerical result for Airfoil

3.5 Conclusion

The discrete representation of the geometry with a set of points is fairly justified with the results obtained for the concentric cylinder case with respect to the error analysis and the order of convergence obtained. Moreover, the results obtained also suggest that the version of the bi-linear interpolation implemented works well.

The variant of bi-linear interpolation used without the use of ghost cells, ensures that the method works even when the geometry, or the section of the geometry is smaller than the mesh size. The same method will be applied for the Euler equations in the subsequent chapters.

CHAPTER 4 EULER EQUATIONS

4.1 Introduction

This chapter deals with the Immersed Boundary method implementation for Euler Equations. Euler equations are a set of hyperbolic conservation laws that govern the dynamics of compressible flows for which the effect of heat flux, viscous forces and body forces are negligible. These equations can be represented in two ways, the first one is using the primitive variables like velocity, density and pressure and the second one is using the conserved variables. These conserved variables are mass density, ρ, the x-momentum component, ρu, y-momentum component, ρv, z-momentum component ρz and the total energy per unit mass E [3]. The five governing conservation laws are defined as:

$$\rho_t + (\rho u)_x + (\rho v)_y + (\rho w)_z = 0 \tag{4.1}$$

$$(\rho u)_t + (\rho u^2 + p)_x + (\rho uv)_y + (\rho uw)_z = 0 \tag{4.2}$$

$$(\rho v)_t + (\rho v^2 + p)_y + (\rho uv)_x + (\rho vw)_z = 0 \tag{4.3}$$

$$(\rho w)_t + (\rho w^2 + p)_z + (\rho uw)_x + (\rho vw)_y = 0 \tag{4.4}$$

$$E_t + [u(E+p)]_x + [v(E+p)]_y + [w(E+p)]_z = 0 \tag{4.5}$$

where E is the total energy per unit volume:

$$E = \rho(\frac{1}{2}\mathbf{V}^2 + e) \tag{4.6}$$

and,

$$\frac{1}{2}\mathbf{V}^2 = \frac{1}{2}\mathbf{V} \cdot \mathbf{V} = \frac{1}{2}(u^2 + v^2 + w^2) \tag{4.7}$$

The conservation laws expressed in Eqs. 4.1 to 4.5 can be represented in a compact form with a column vector of conserved variables U and flux vectors $F(U)$, $G(U)$ and $H(U)$ in x,y and z directions respectively. The compact form is:

$$U_t + F(U)_x + G(U)_y + H(U)_z = 0 \tag{4.8}$$

where,

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix}, F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(E+p) \end{bmatrix}, G = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v(E+p) \end{bmatrix}, H = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w(E+p) \end{bmatrix}$$

For 1-D problems, the terms for $v = 0$ and $w = 0$ are absent. So the final set of equations for 1-D Euler equations become:

$$\rho_t + (\rho u)_x = 0 \tag{4.9}$$

$$(\rho u)_t + (\rho u^2 + p)_x = 0 \tag{4.10}$$

$$E_t + [u(E+p)]_x = 0 \tag{4.11}$$

Similarly, the compact form is given as:

$$U_t + F(U)_x = 0 \tag{4.12}$$

where, $U = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix}, F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(E+p) \end{bmatrix}$

Let us first analyze the type of these equations and their numerical behaviour. For few cases, analytical solutions are available which will act as test cases and bench-marking for the IB implementation.

4.2 Euler Equations : Overview

4.2.1 Classification of Partial Differential Equations

Differential equations can be classified as parabolic, hyperbolic or elliptic which can be found out using eigenvalues.

For a quasi-linear system of equation,

$$a_1 \frac{\partial u}{\partial x} + b_1 \frac{\partial u}{\partial y} + c_1 \frac{\partial v}{\partial x} + d_1 \frac{\partial v}{\partial y} = 0 \tag{4.13}$$

$$a_2 \frac{\partial u}{\partial x} + b_2 \frac{\partial u}{\partial y} + c_2 \frac{\partial v}{\partial x} + c_2 \frac{\partial v}{\partial y} = 0 \tag{4.14}$$

Using, $W = \begin{bmatrix} u \\ v \end{bmatrix}$. Then the system of equations represented by Eqs. 4.13 and 4.14 can be written as:

$$\begin{bmatrix} a_1 & c_1 \\ a_2 & c_2 \end{bmatrix} \frac{\partial W}{\partial x} + \begin{bmatrix} b_1 & d_1 \\ b_2 & d_2 \end{bmatrix} \frac{\partial W}{\partial y} = 0 \tag{4.15}$$

$$\Big[K\Big] \frac{\partial W}{\partial x} + \Big[M\Big] \frac{\partial W}{\partial y} = 0 \tag{4.16}$$

Multiplying by $[K]^{-1}$,

$$\frac{\partial W}{\partial x} + \Big[K\Big]^{-1} \Big[M\Big] \frac{\partial W}{\partial y} = 0 \tag{4.17}$$

which finally gives,

$$\frac{\partial W}{\partial x} + \Big[N\Big] \frac{\partial W}{\partial y} = 0 \tag{4.18}$$

Eigenvalues for $[N]$ are obtained simply from the solution of the following equation:

$$|N - \lambda \mathbf{I}| = det(N - \lambda \mathbf{I}) = 0 \tag{4.19}$$

where $\mathbf{I}$ is the identity matrix

If the eigenvalue of N are all real, then the system of equation is hyperbolic. If the eigenvalues of N are all complex then the system of equation is elliptic.

4.2.2 Eigenvectors

A partial differential equation in compact form which is given as:

$$U_t + F(U)_x = 0 \tag{4.20}$$

where,

$$U = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}, F(U) = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \tag{4.21}$$

The jacobian matrix is represented as:

$$N(U) = \frac{\partial F}{\partial U} = \begin{bmatrix} \partial f_1/\partial u_1 \dots \partial f_n/\partial u_1 \\ \partial f_1/\partial u_2 \dots \partial f_n/\partial u_2 \\ \vdots \\ \partial f_1/\partial u_n \dots \partial f_n/\partial u_n \end{bmatrix} \tag{4.22}$$

The eigenvalues of the system are the zeros of the characteristic polynomial and for the differential equation presented in Eq. 4.20, as per [3], these are represented by,

$$|N - \lambda \mathbf{I}| = det \begin{bmatrix} 0 - \lambda & \rho_0 \\ a^2/\rho_0 & 0 - \lambda \end{bmatrix} \tag{4.23}$$

That is $\lambda^2 = a$, which gives;

$$\lambda = a; \lambda = -a \tag{4.24}$$

The goal now is to find the eigenvectors for the system of equations. Eigenvectors can be classified as right eigenvectors and left eigenvectors. A right eigenvector of a matrix $\mathbf{A}$ corresponding to an eigenvalue λ_i of $\mathbf{A}$ is a vector $K^i = [k_1^i \quad k_2^i \quad ...k_n^i]^T$ such that $AK^i = \lambda_i K^i$. Similarly, the left eigenvectors are defined as $L^i A = \lambda_i L^i$

4.2.3 Euler Equations: Eigenvalue and Eigenvector

The one-dimensional Euler Eqs., 5.1 to 4.11, in the conservative are,

$$U_t + F(U)_x = 0 \tag{4.25}$$

where,

$$U_t = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix}, \qquad F(U)_x = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(E + p) \end{bmatrix} \tag{4.26}$$

Also, $E = \rho(\frac{1}{2}u^2 + e)$ and $e = e(\rho, p) = \frac{p}{(\gamma - 1)\rho}$ and the sound speed can be written as:

$$a = \sqrt{\frac{\gamma p}{\rho}} \tag{4.27}$$

where, γ is the gas constant. The conservation laws from Eqs. 4.25 to 4.26 can be written as,

$$U_t + A(U)U_x = 0 \tag{4.28}$$

where $A(U)$ is the jacobian matrix defined as:

$$A(U) = \frac{\partial F}{\partial U} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \frac{\partial f_1}{\partial u_3} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \frac{\partial f_2}{\partial u_3} \\ \frac{\partial f_3}{\partial u_1} & \frac{\partial f_3}{\partial u_2} & \frac{\partial f_3}{\partial u_3} \end{bmatrix} \tag{4.29}$$

To find the values for the Jacobian matrix the f_i in the flux vector is expressed as a function of the conserved variables. Since $f_1 = \rho u$, so it is required to express pressure p as a function of conserved quantity giving,

$$p = (\gamma - 1)[u_3 - \frac{1}{2}(u_2)^2/u_1] \tag{4.30}$$

The flux vector can be written as:

$$F(U) = \begin{bmatrix} u_2 \\ \frac{1}{2}(3-\gamma)u_2^2/u_1 + (\gamma-1)u_3 \\ \gamma\frac{u_2 u_3}{u_1} - \frac{1}{2}(\gamma-1)\frac{u_2^2}{u_1^2} \end{bmatrix} \tag{4.31}$$

Finally, the Jacobian matrix gives,

$$A(U) = \begin{bmatrix} 0 & 1 & 0 \\ -\frac{1}{2}(\gamma-1)(\frac{u_2}{u_1})^2 & (3-\gamma)\frac{u_2}{u_1} & (\gamma-1) \\ \frac{\gamma u_2 u_3}{u_1^2} + (\gamma-1)(\frac{u_2}{u_1})^3 & \frac{\gamma u_3}{u_1} - \frac{3}{2}(\gamma-1)(\frac{u_2}{u_1})^2 & \gamma(\frac{u_2}{u_1}) \end{bmatrix} \tag{4.32}$$

In terms of the speed of sound, a, and velocity, u, the Jacobian matrix $A(U)$ can be written

as:

$$A(U) = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{2}(\gamma-3)u^2 & (3-\gamma)u & (\gamma-1) \\ \frac{1}{2}+(\gamma-2)u^3-\frac{a^2u}{\gamma-1} & \frac{3-2\gamma}{2}u^2+(\frac{a^2}{\gamma-1}) & \gamma u \end{bmatrix} \tag{4.33}$$

The eigenvalues and the corresponding eigenvectors for the jacobian matrix, A(U), can be calculated the same way as presented in sub-sections 4.2.1 and 4.2.2 as,

$$\lambda_1 = u - a, \quad \lambda_2 = u \quad \text{and} \quad \lambda_3 = u + a \tag{4.34}$$

and the corresponding eigenvectors are:

$$K^{(1)} = \begin{bmatrix} 1 \\ u-a \\ H-ua \end{bmatrix}, \quad K^{(2)} = \begin{bmatrix} 1 \\ u \\ \frac{1}{2}u^2 \end{bmatrix} \quad K^{(3)} = \begin{bmatrix} 1 \\ u+a \\ H+ua \end{bmatrix} \tag{4.35}$$

where H is the total enthalpy, which can be expressed as, $H = (E + p)/\rho$.
From the values of the eigenvalues and eigenvectors, it can be concluded that all the eigenvalues are real as well as distinct which proves that the Euler equations are strictly hyperbolic in nature as long as the sound speed a remains positive.

4.3 Riemann Problem: 1-D Euler

The Riemann problem is the initial value problem (IVP) for the conservation laws represented by Eqs. 4.25 and 4.26 with initial conditions:

$$U(x,0) \equiv U^0(x) = \begin{cases} U_L, & \text{if } x < 0 \\ U_R, & \text{if } x > 0 \end{cases}$$

The domain of interest in the (x, t) frame is $-\infty < x < \infty$ for $t > 0$. Usually the domain is fixed by L_L and L_R in left and right directions, respectively. For solving the present problem, $W = (\rho, u, p)^T$ will be used which is nothing but the primitive variables and are density, particle velocity and pressure, respectively.

The eigenvalues have been obtained in the previous section for the one-dimensional Euler equations and these values define the regions for the solution to be calculated. The Riemann problem described above is physically relevant to shock-tube problem in 1-D. As far as the

theoretical solution is concerned, different regions defined by the eigenvalues which has been presented in Fig. 4.1.

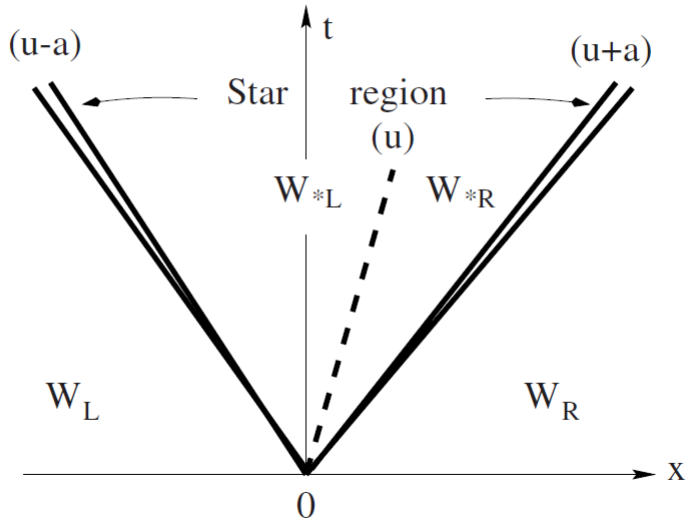


Figure 4.1 Various regions based on Eigenvalues [3]

The values of particle velocity u^* and p^* have the same values in the right and the left star regions the only thing that is varying is the density value, namely ρ_L^* and ρ_R^*. Since, the value for the pressure and particle velocity is constant in the star region, this will be used in calculating the ρ_L^* and ρ_R^*.

The solution for p^* can be obtained from Eq. 4.36

$$f(p, W_L, W_R) = f_L(p, W_L) + f_R(p, W_R) + \Delta u \quad \text{where,} \quad \Delta u = u_R - u_L \tag{4.36}$$

where, f_L is defined in [3] as:

$$f_L(p, W_R) = \begin{cases} (p - p_L)\left[\frac{A_L}{p+B_L}\right]^{\frac{1}{2}}, & \text{if } p > p_L \quad (shock) \\ \\ \frac{2a_L}{(\gamma-1)}\left[\left(\frac{p}{p_L}\right)^{\frac{\gamma-1}{2\gamma}} - 1\right], & \text{if } p < p_L \quad (rarefaction) \end{cases}$$

f_R is given by:

$$f_L(p, W_R) = \begin{cases} (p - p_R)\left[\frac{A_R}{p+B_R}\right]^{\frac{1}{2}}, & \text{if } p > p_R \quad (shock) \\ \\ \frac{2a_R}{(\gamma-1)}\left[\left(\frac{p}{p_R}\right)^{\frac{\gamma-1}{2\gamma}} - 1\right], & \text{if } p < p_R \quad (rarefaction) \end{cases}$$

And A_L, A_R, B_L and B_R are given by:

$$A_L = \frac{2}{(\gamma+1)\rho_L}, \quad B_L = \frac{\gamma-1}{(\gamma+1)}\rho_L, \quad A_R = \frac{2}{(\gamma+1)\rho_R}, \quad B_R = \frac{\gamma-1}{(\gamma+1)}\rho_R \tag{4.37}$$

The solution for the velocity in the star region is:

$$u_* = \frac{1}{2}(u_L + u_R) + \frac{1}{2}[f_R(p*) - f_L(p*)] \tag{4.38}$$

In order to solve for the p^* value, one needs to use an iterative scheme, like Newton-Ralphson. Then the ρ_L^* and ρ_R^* values are computed and depend on whether the wave is a shock or rarefaction. For this also, proper formulations and references can be found in [3].

4.3.1 Conservation Laws and Godunov method

A control volume represented by Fig. 4.2 can be used to represent the conservation law in the integral form [3]

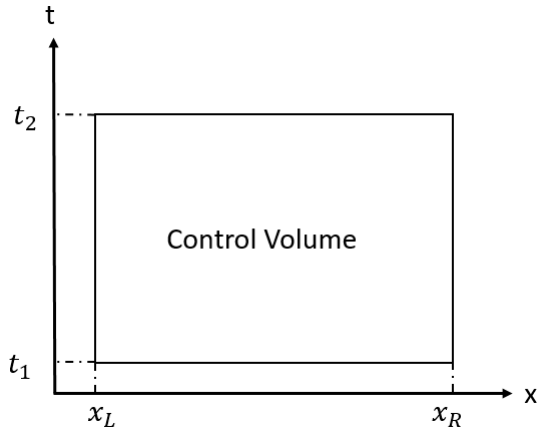


Figure 4.2 A control volume representation

Any conservation law represented in the form:

$$U_t + F(U)_x = 0 \tag{4.39}$$

For a generalised Initial-Boundary Value problem for a non-linear systems of hyperbolic conservation laws represented as:

$$PDEs: \quad U_t + F(U)_x = 0, ICs: \quad U(x,0) = U^{(0)}(x), BCs: \quad U(0,t) = U_1(t), U(L,t) = U_r(t) \tag{4.40}$$

where U is the vector of conserved variables and F is the Flux vector. It can be represented in the integral such that to apply the finite volume schemes as:

$$\frac{d}{dt}\int_{x_L}^{x_R} U(x,t)dx = F(U(x_L,t)) - F(U(x_R,t)) \tag{4.41}$$

This is one of the versions of the integral form. The other possible way of representing it is:

$$\int_{x_L}^{x_R} U(x,t_2)dx = \int_{x_L}^{x_R} U(x,t_1)dx + \int_{t_1}^{t_2} F(U(x_L,t))dt - \int_{t_1}^{t_2} F(U(x_R,t))dt \tag{4.42}$$

Since Euler equations can be represented in the conservation form, this information can be used to representing these in the finite volume form.

The discretisation of the domain is required to be done for the control volume domain represented in the Fig. 4.2. The domain $[0, L]$ can be discretised in M equidistant cells or finite volumes such that for a given cell I_i, the location of the cell cell boundaries are given by

$$x_{i-1/2} = (i-1)\Delta x \quad , \quad x_i = (i-1/2)\Delta x \quad \text{and} \quad x_{i+1/2} = i\Delta x \tag{4.43}$$

A piece-wise constant representation of the data is considered which is realised by defining cell averages:

$$U_i^n = \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} \tilde{U}(x,t_n)dx \tag{4.44}$$

where, $\tilde{U}(x,t_n)$ is the general initial data at time t^n and this leads to a desired constant piece-wise distribution of $U(x,t^n)$ with

$$U(x,t^n) = U_i^n, \quad \text{for } x \text{ in each cell} \quad I_i = [x_{i-1/2}, x_{i+1/2}] \tag{4.45}$$

Now, applying the integral form of the conservation equation, Eq. 4.42, to a control volume $[x_1, x_2] \times [t_1, t_2]$ where $x_1 = x_{i-1/2}$, $x_2 = x_{i+1/2}$, $t_1 = t^n$ and $t_2 = t^{n+1}$. Eq. 4.42 finally provides,

$$\int_{x_{i-1/2}}^{x_{i+1/2}} \tilde{U}(x, t_{n+1})dx = \int_{x_{i-1/2}}^{x_{i+1/2}} \tilde{U}(x, t_n)dx + \int_0^{\Delta t} F[\tilde{U}(x_{i-1/2}, t)]dt - \int_0^{\Delta t} F[\tilde{U}(x_{i+1/2}, t)]dt \quad (4.46)$$

When Δt is sufficiently small to avoid wave interaction, the global solution for $\tilde{U}(x,t)$ in the strip $0 \leq x \leq L$ and $t^n \leq t \leq t^{n+1}$ in terms of local solution can be defined as:

$$\tilde{U}(x,t) = U_{i+1/2}(\bar{x}/\bar{t}), \quad x\epsilon[x_i, x_{i+1}] \quad (4.47)$$

where the correspondence between local $\bar{x}, \bar{t}$ and global (x,t) is given by:

$$\bar{x} = x - x_{i+1/2}, \quad \bar{t} = t - t^n \quad (4.48)$$

With additional information of $\Delta t \leq \frac{\Delta x}{S^n_{max}}$ and Eq. 4.47 provides, one can write,

$$\tilde{U}(x_{i-1/2}, t) = U_{i-1/2}(0) = constant \quad (4.49)$$

and,

$$\tilde{U}(x_{i+1/2}, t) = U_{i+1/2}(0) = constant \quad (4.50)$$

where, $U_{i+1/2}(0)$ is the solution of the Riemann problem $RP(U^n_i, U^n_{i+1})$ along the ray $x/t = 0$ which is the t axis of the local frame centered at the cell. Similarly $U_{i-1/2}(0)$ is the solution of the Riemann problem $RP(U^n_{i-1}, U^n_i)$. Dividing Eq. 4.46 by Δx and taking the constants out of the integration, we finally get,

$$1/\Delta x \int_{x_{i-1/2}}^{x_{i+1/2}} \tilde{U}(x, t_{n+1})dx = 1/\Delta x \int_{x_{i-1/2}}^{x_{i+1/2}} \tilde{U}(x, t_n)dx + \frac{\Delta t}{\Delta x}[F(U_{i-1/2}(0)) - F(U_{i+1/2}(0))] \quad (4.51)$$

Finally in the compact form, this can be represented as

$$U^{n+1}_i = U^n_i + \frac{\Delta t}{\Delta x}[F_{i-1/2} - F_{i+1/2}] \quad (4.52)$$

where, the inter-cell flux is given by

$$F_{i+1/2} = F(U_{i+1/2}(0)), \tag{4.53}$$

and the time step is defined by,

$$\Delta t \leq \frac{\Delta x}{S^n_{max}} \tag{4.54}$$

where, S^n_{max} donates the maximum wave velocity throughout the domain. Generally, for numerical problems this is expressed as $S^n_{max} = max_i[\mid u^n_i \mid + \mid a^n_i \mid]$ where $a = (\frac{\gamma p}{\rho})^{1/2}$. u^n_i is the particle velocity and a^n_i is the wave speed for cell i.

There are various ways to find the Fluxes so as to march in time and finally calculate the primitive variables namely, p, ρ and u. One such method is called Roe scheme.

4.4 Roe Scheme

The Roe scheme is a numerical method for solving the Euler equations. It was first presented by [53] and since has been used and modified in a number of ways. For the present case, the original Roe scheme has been used as presented in [3].

The Roe scheme is based on the calculation of the numerical fluxes used for computing the conserved quantities. The Riemann problem is defined as:

$$U_t + F(U)_x = 0 \quad \text{and}$$
$$U(x,0) = \begin{cases} U_L, & \text{if } x < 0 \\ U_R, & \text{if } x > 0 \end{cases}$$

and can be recast by introducing a Jacobian matrix,

$$A(U) = \frac{\partial F}{\partial U} \tag{4.55}$$

Using the chain rule, it can be written as,

$$U_t + A(U)U_x = 0 \tag{4.56}$$

The jacobian matrix A(U) is replaced with a constant Jacobian matrix

$$\tilde{A} = \tilde{A}(U_L, U_R) \tag{4.57}$$

Finally, the approximate Riemann problem can be approximated as

$$U_t + \tilde{A}U_x = 0 \quad \text{with} \quad U(x,0) = \begin{cases} U_L, & \text{if } x < 0 \\ U_R, & \text{if } x > 0 \end{cases} \tag{4.58}$$

which is then solved exactly.

4.4.1 The Inter-cell Flux

The inter-cell flux, as per [3], is given as:

$$F_{i+\frac{1}{2}} = \frac{1}{2}(F_R + F_L) - \frac{1}{2}\sum_{i=1}^{m} \tilde{\alpha_i}|\tilde{\lambda_i}|\tilde{K^{(i)}} \tag{4.59}$$

where, $m = 3$ for one-dimensional case and F_R and F_L are the flux values for the right and left face of the i^{th} cell calculated from the primitive variables from the previous time step. $\tilde{\alpha}$ are the wave strengths computed in the next section and $\tilde{K}$ are the right eigenvectors as defined in Eq. 4.35 with averaged variables which are defined in the next section.

4.4.2 Roe Scheme : Euler Equations

The eigenvalues for the Jacobian matrix as represented in [3]:

$$\tilde{\lambda_1} = \tilde{u} - \tilde{a}, \quad \tilde{\lambda_2} = \tilde{u}, \quad \tilde{\lambda_3} = \tilde{u} + \tilde{a} \tag{4.60}$$

The corresponding Roe averages as expressed in [3] are:

$$\tilde{u} = \frac{\sqrt{\rho_L}u_L + \sqrt{\rho_R}u_R}{\sqrt{\rho_R} + \sqrt{\rho_R}} \tag{4.61}$$

$$\tilde{H} = \frac{\sqrt{\rho_L}H_L + \sqrt{\rho_R}H_R}{\sqrt{\rho_R} + \sqrt{\rho_R}} \tag{4.62}$$

$$\tilde{a} = ((\gamma - 1)(\tilde{H} - \frac{1}{2}\tilde{u}^2))^{0.5} \tag{4.63}$$

$$\Delta U = \sum_{i=1}^{3} \tilde{\alpha_i}\tilde{K^{(i)}} \tag{4.64}$$

$$\tilde{\alpha_1} + \tilde{\alpha_2} + \tilde{\alpha_5} = \Delta u_1 \tag{4.65}$$

$$\tilde{\alpha_1}(\tilde{u} - \tilde{a}) + \tilde{\alpha_2}\tilde{u} + \tilde{\alpha_5}(\tilde{u} + \tilde{a}) = \Delta u_2 \tag{4.66}$$

$$\tilde{\alpha_1}(\tilde{H} - \tilde{u}\tilde{a}) + \frac{1}{2}\tilde{\alpha_2}\tilde{u}^2 + \tilde{\alpha_5}(\tilde{H} + \tilde{u}\tilde{a}) = \Delta u_3 \tag{4.67}$$

Finally, the values of $\tilde{\alpha_1}$, $\tilde{\alpha_2}$ and $\tilde{\alpha_5}$, which will be used for calculating the final flux values, are given as

$$\tilde{\alpha_2} = \frac{\gamma - 1}{\tilde{a}^2}[\Delta u_1(\tilde{H} - \tilde{u}^2) + \tilde{u}\Delta u_2 - \overline{\Delta u_5}], \tag{4.68}$$

$$\tilde{\alpha_1} = \frac{1}{2\tilde{a}}[\Delta u_1(\tilde{u} + \tilde{a}) + \Delta u_2 - \tilde{a}\tilde{\alpha_2}], \tag{4.69}$$

$$\tilde{\alpha_5} = \Delta u_1 - (\tilde{\alpha_1} + \tilde{\alpha_2}) \tag{4.70}$$

Where,

$$\overline{\Delta U_5} = \Delta u_5 \tag{4.71}$$

Summary of the algorithm to compute the Roe numerical fluxes

Step 1: Compute the Roe average values for $\tilde{u}$, $\tilde{H}$ and $\tilde{a}$ according to Eqs. 4.62 and 4.61

Step 2: Compute the average eigenvalues

Step 3: Compute the average right eigenvectors $\tilde{K}^{(i)}$

Step 4: Compute the wave strength $\tilde{\alpha_i}$ according to the equations

Step 5: Use all the above quantities to calculate $F_{i+\frac{1}{2}}$

4.4.3 Entropy fix

The entropy fix is required when we have the rarefaction wave travelling in either left or right direction; the tip of the rarefaction wave travels with a certain speed and the tail with other speed such that the $u - a$ for the tip is positive and it is negative for the tail part. Then we have a discontinuity within this rarefaction wave which has not been taken care by the Roe scheme.

There are several ways to do an entropy fix. Harten-Hyman entropy fix [54] is one of them and will be used in this work.

4.5 Inviscid compressible Flow: Immersed Boundary Method

The Immersed Boundary Method has been implemented for inviscid compressible flow problems, Euler Equations, using the Roe scheme in a very simple way. Previous work on the

IB implementation for compressible inviscid flows includes [55] which focuses on the grid triangulation. The work by [56] focuses on the IB implementation for Euler equations in building-cube method. Notable work on sharp-interface IB implementation for compressible flows includes [57] and [58].

4.5.1 Meshing, Discretisation and Tagging

The 1-D Euler equations, Eqs. 5.1 to 4.11, are solved numerically using Roe scheme which is described in Sec. 4.4 using a uniform mesh. The tagging methodology, as described in Sec. 2.5, is not required for 1-D problems. The cell centres next to the boundary, on both side of the domain of interest are interface cells, cells inside the domain of interest are fluid cells and the cells outside the domain of interest are solid cells. This is described in Fig. 4.3.

Figure 4.3 Cell classification for one dimensional problem

4.5.2 Stationary boundary

The IB method is applied to the shock tube problem shown in Fig. 4.4. The distance between the cell centres are modified compared to the body-fitted approach which is presented in Fig. 4.5. The distance between the last cell centre and the wall is changed to $0.4Dx$ keeping the length of the domain the same and distributing the $0.1Dx$ equally to the rest of the cells. The $0.4Dx$ is an arbitrary length and any other length can be chosen for the demonstration purpose. For fluid cells the primitive variables, pressure, density and velocity, are calculated using the Roe scheme. For the interface cells, a reconstruction scheme is used.

Since the problem is 1-D, the reconstruction scheme is changed to linear interpolation instead of the bi-linear interpolation which was used for 2-D diffusion problems. For the interface cell in Fig. 4.5, a linear interpolation between the interface cell, the wall and the nearest fluid cell has to be carried out. For pressure and density a Neumann boundary condition of $\frac{\partial P}{\partial n} = 0$ and $\frac{\partial \rho}{\partial n} = 0$ is implemented and it is assumed that the pressure at the interface point and the wall are same.

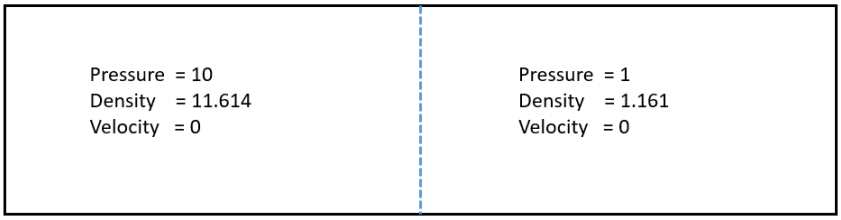


Figure 4.4 Shock tube schematic

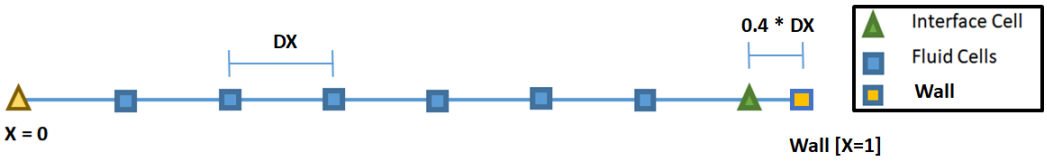


Figure 4.5 Discretized domain for 1-D IB implementation

The results of the shock tube problem with shock reflection from the right wall is presented in Fig. 4.6 and it has been compared to the traditional, body-fitted, approach. The total time of simulation is $t = 0.025s$.

4.5.3 Moving boundaries

Mainly two different scenarios are encountered when the geometry or the boundary starts moving, with respect to cell conversions, and are presented in subsequent sections. In moving boundary problems, one additional treatment is required when cells change type as the boundary moves past a given cell.

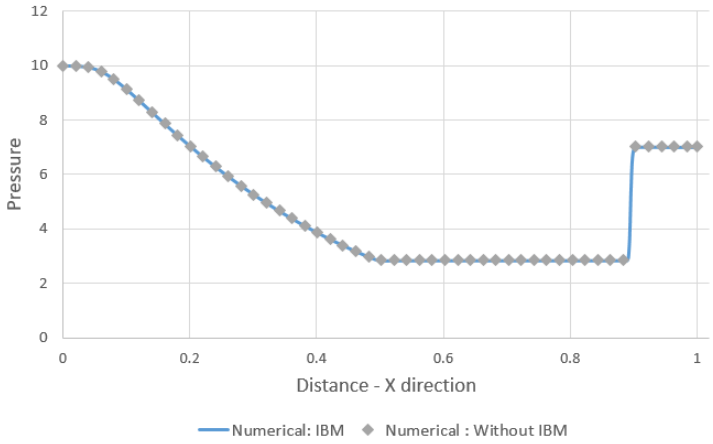
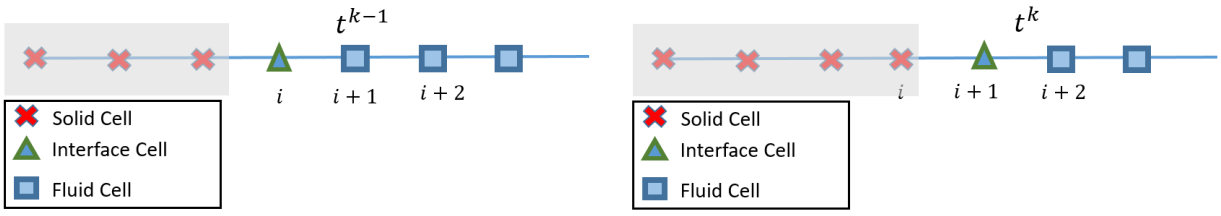


Figure 4.6 Pressure comparison: With and without IBM for the shock tube problem at $t = 0.025s$

Fluid-to-Solid cell conversion

Fluid-to-solid conversion is described in Fig. 4.7. The i^{th} cell number is an interface cell at time t^{k-1} and it becomes a solid cell at time t^k. Similarly, the $(i+1)^{th}$ cell which is a fluid cell at time t^{k-1} becomes an interface cell at time t^k. This kind of cell conversion does not pose any problem with respect to IB implementation as the time histories for all the cells for which the numerical scheme is used, are known beforehand. Moreover, the fluid cells for which there is a status change, fluid-to-interface, the primitive variables are interpolated using linear interpolation. Fig. 4.9 shows different cell centres with right wall in motion. The region between two boundaries has a pressure, $p = 10$, Velocity , $u = 0$ and density $\rho = 11.614$. The wall velocity is fixed at $U_w = 0.005m/s$. The problem schematic is shown in Fig. 4.8.

The wall movement is expected to produce a moving shock inside the tube which will travel towards left with the advancement of the wall.



(a) Cell status at time t^{k-1}

(b) Cell status at time t^k

Figure 4.7 Fluid-to-solid conversion

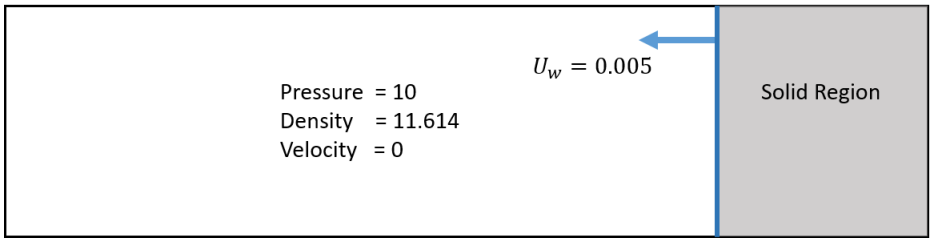


Figure 4.8 Wall movement towards left schematic

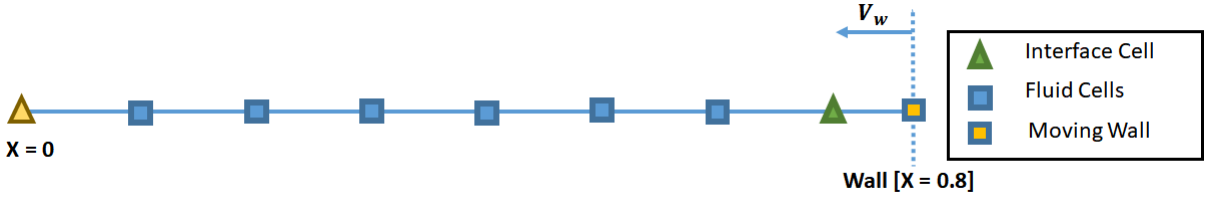
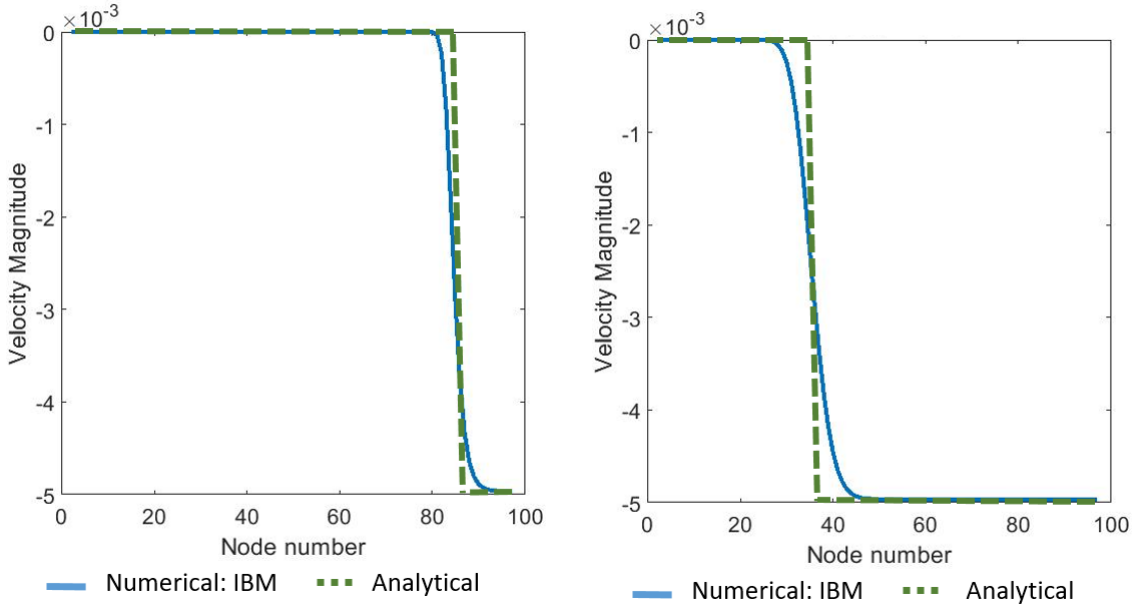


Figure 4.9 Discretized domain for wall movement towards left

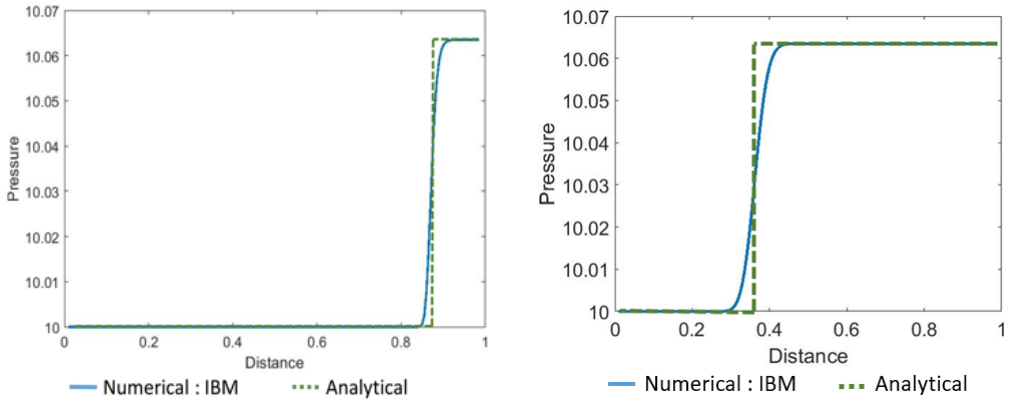
The numerical simulation results for the above mentioned case has been presented in Fig. 4.10.



(a) Velocity profile after t = 0.13 seconds

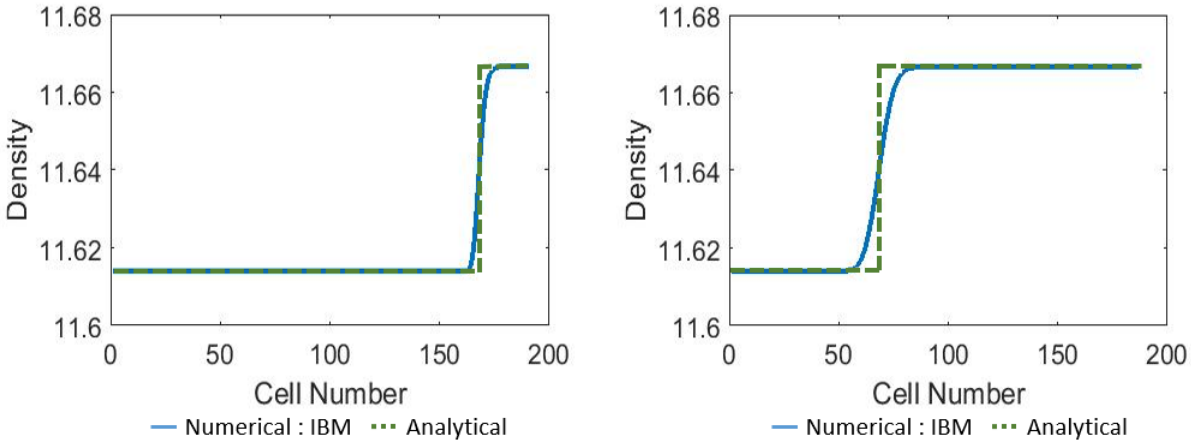
(b) Velocity profile at t = 0.58 seconds

Figure 4.10 Wall movement towards left at two different time steps



(a) Pressure profile after t = 0.13 seconds (b) Pressure profile at t = 0.58 seconds

Figure 4.11 Pressure plot for wall movement towards left at two different time steps



(a) Density profile after t = 0.13 seconds (b) Density profile at t = 0.58 seconds

Figure 4.12 Density profile for wall movement towards left at two different time steps

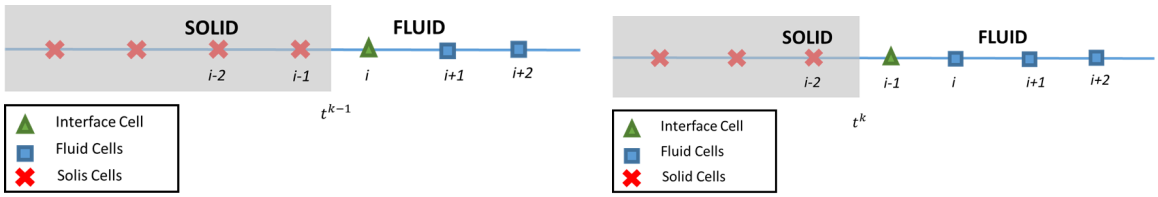
4.5.4 Moving boundaries: solid-to-fluid cell conversion

Solid-to-fluid cell conversion poses a challenge with respect to IB method implementation. In Fig. 4.13(a), the cell and wall status is shown for at t^{k-1} time step and as the wall moves towards right with the specified velocity for the next time step, t^k, the cell and wall status is as per Fig. 4.13(b). This transition involves two kinds of cell conversion:

1. **Interface-to-Fluid :** The interface cells in Fig. 4.13(a) are converted into fluid cells in Fig. 4.13(b). This poses a major challenge with respect to the IB implementation

as the values at time step t^k depends on the value calculated at time step t^{k-1}. For all the newly defined fluid points, the part of the Euler equations that depends on the derivatives from the previous time step will not be correct. Therefore, a special treatment for the primitive values at these cells are required.

2. **Solid-to-Fluid :** Solid cells (Ghost cells) in Fig. 4.13(a) are converted into interface cells in Fig. 4.13(b). This does not pose any problem with respect to the IB implementation as the solution for time step t^k will be reconstructed using an interpolation scheme and does not depend on its value at the previous time step t^{k-1}.



(a) Cell status at t^{k-1} (b) Cell status at t^k

Figure 4.13 Solid-to-Fluid conversion

There are different approaches to solve this problem. Among these methods, the Field Extension Method is the most common, which has been proposed by [4] and is described in the following section. A novel idea of the Two-level interpolation is introduced and used to solve the problems which are solved using Field Extension in order to verify and validate of the novel idea.

4.5.5 Field Extension Method

Yang et. al [4] explain the Field extension method which relies on the extrapolation of the primitive variables to the nearest ghost cell. The various steps involved are identified below,

Steps for Field Extension implementation

Step 1: Identify the Ghost cell, the nearest cell inside the solid, for a particular time step

Step 2: Extrapolate the values from the interface cell to this Ghost Cell

Step 3: Use this value for calculating the fluxes, which will be used to update the primitive variables for all the cells, including the interface cells, for the next time step

This way, the solid-to-fluid conversion of the cells is tackled. Since, the problem is 1-D, linear extrapolation is implemented using the i^{th} and $(i+1)^{th}$ points at the $(i-1)^{th}$ solid point, as per Fig. 4.13(a).

The schematic for the problem definition is presented in the Fig. 4.14. The region between two boundaries has a pressure, $P = 11.61$, Velocity , $U = 0$ and density $D = 10$. The wall velocity is fixed at $U_w = 0.005m/s$ towards the right.

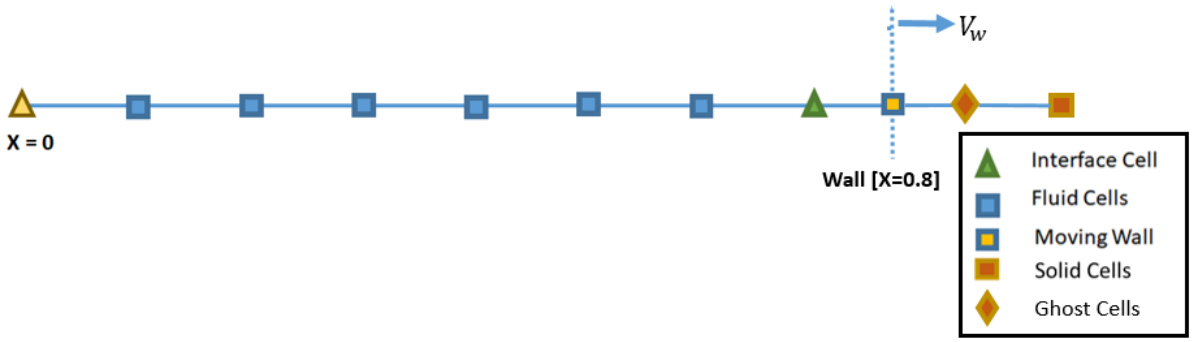


Figure 4.14 Wall movement towards right

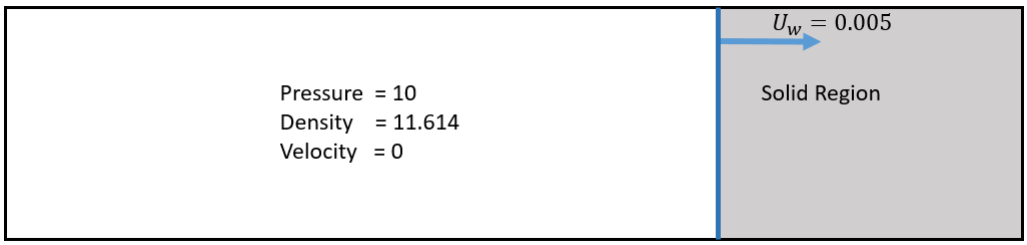
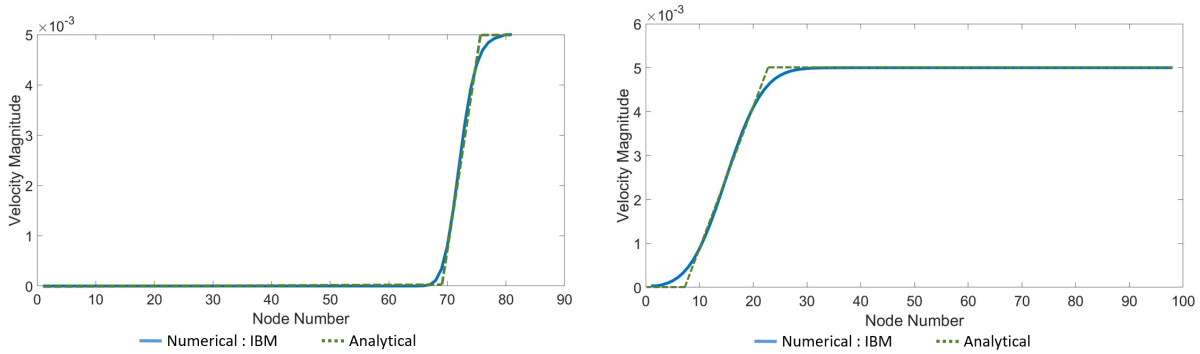


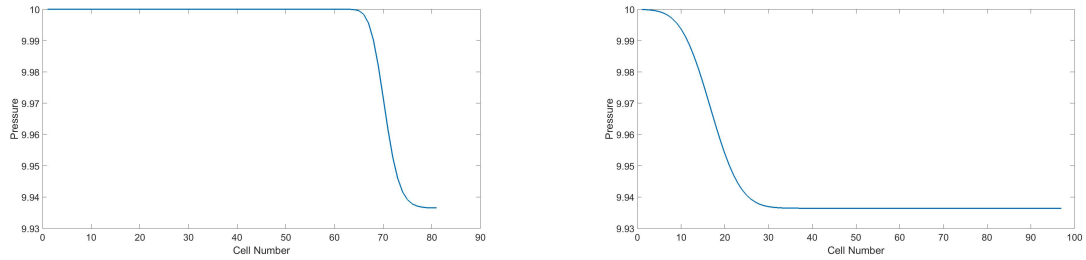
Figure 4.15 Schematic for wall movement towards right

The schematic for the wall movement is presented in Fig. 4.15 and the numerical solution for this problem is presented in Fig. 4.16, 4.17 and 4.18 for two different time steps.



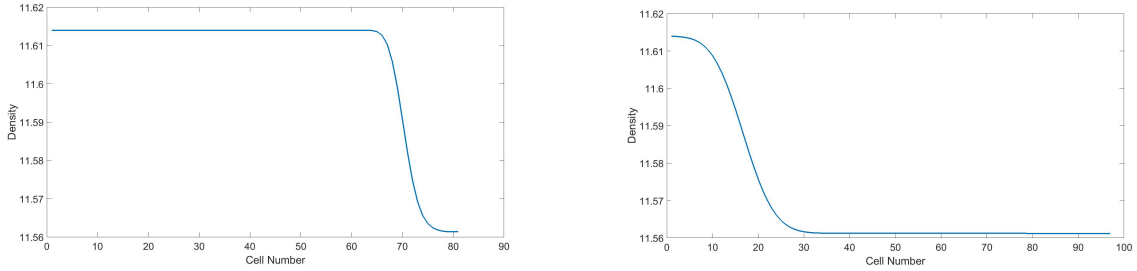
(a) Cell status at 0.08 seconds (b) Status after 6 seconds

Figure 4.16 Solid-to-Fluid conversion numerical result: velocity profile for two different time steps



(a) Pressure profile at 0.08 seconds (b) Pressure profile after 6 seconds

Figure 4.17 Solid-to-Fluid conversion numerical result: Pressure profile for two different time steps



(a) Density profile at 0.08 seconds (b) Density profile after 6 seconds

Figure 4.18 Solid-to-Fluid conversion numerical result: Density profile for two different time steps

4.5.6 Two-level Interpolation

One of the major drawbacks in the state-of-the-art methods in dealing with solid-to-fluid conversion of the cells is that these methods rely on the ghost cells. There can be a problem when there is no Ghost cell available because of the orientation, shape and size of the geometry. This will become more evident with respect to 2-D problems and is described in detail in the following chapter.

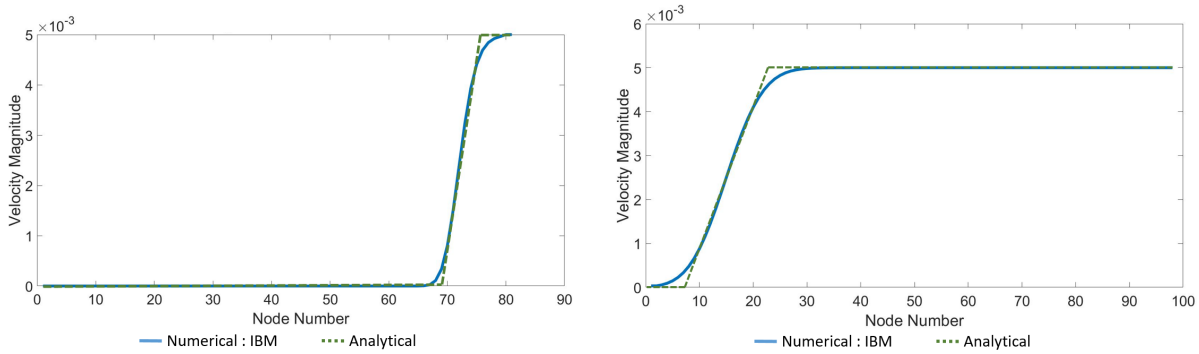
The idea is to get rid of the solution dependency on the ghost cells. One way to do that is to use interpolation for two cells, instead of just one. Therefore, the interpolation is done not only for the interface cells but also for the fluid cells for which interface cells are part of the numerical stencil. The modification rely solely on the fluid side by performing a Two-level interpolation.

None of the cells from solid region are selected for any field value calculation. As shown in Fig. 4.13, the values at the intercepted cell is calculated using the linear interpolation, considering boundary condition, $i+1$ and $i+2$ cell numbers using the equation:

$$\phi_i = a_0 + a_1 y + a_2 y^2 \tag{4.72}$$

where, values for a_0, a_1 and a_2 depends on the values of field variable ϕ at these points. Once the solid boundary crosses one cell, the field values for i^{th} cell at t^k has to be calculated by the linear interpolation using the boundary condition, $i+1$ and $i+2$ cells and for the $i-1$ cell, which is now an interface cell, field values have to be calculated using boundary condition, $i+1$ and $i+2$. Performing the interpolation for the i^{th} cell at t^k ensures that there is no oscillations in calculation of the primitive variables, but at the same time it induces errors related to the interpolation scheme selected.

Two-level interpolation approach has been to the same cases as in the Field Extension approach. The results obtained are similar to that of Field Extension approach as presented in Fig. 4.19



(a) Velocity profile at 0.08 seconds

(b) Velocity profile after 6 seconds

Figure 4.19 Solid-to-Fluid conversion numerical result: Density profile for two different time steps

4.5.7 Conclusion

The Field Extension Method method and the newly introduced Two-level interpolation method produces equally good results which can be concluded from the various cases presented in this chapter. The most interesting thing about the Two-Level interpolation method is that the solution relies solely in the fluid domain, which can be very advantageous with respect to cases when no ghost cells are available.

CHAPTER 5 2-D EULER EQUATIONS

This chapter deals with the solution of two-dimensional problems for the Euler equations. The 1-D problems described in the previous chapter has been extended to 2-D using Field Extension Method of [4] and Two level interpolation method.

5.1 Euler equations and Roe-Scheme implementation for 2-D

As described earlier, the 2-D Euler equations are given by:

$$U_t + F(U)_x + G(U)_y = 0 \tag{5.1}$$

where, U, F and G are given by:

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}, F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{bmatrix}, G = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{bmatrix}$$

Dimensional splitting has been used for Roe scheme implementation. As described in [3], sweeping is done direction wise. For x-sweep the following equations are solved:

$$\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}_t + \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{bmatrix}_x = 0$$

and for y-sweep, it is required to solve:

$$\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}_t + \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{bmatrix}_x = 0$$

Explicitly, for x-sweep, the equations can be written as,

$$U_{i,j}^{n+1/2} = U_{i,j}^n + \frac{\Delta t}{\Delta x}[F_{i-\frac{1}{2},j}^n - F_{i+\frac{1}{2},j}^n] \tag{5.2}$$

and for the y-sweep:

$$U_{i,j}^{n+1} = U_{i,j}^{n+\frac{1}{2}} + \frac{\Delta t}{\Delta x}[G_{i-\frac{1}{2},j}^{n+\frac{1}{2}} - G_{i+\frac{1}{2},j}^{n+\frac{1}{2}}] \tag{5.3}$$

For calculating the time step, the maximum wave speed in both the directions have to be considered.

5.1.1 Tagging and Solver coupling

The Tagging steps for 2-D Euler problems is similar to that described in the 2-D diffusion equation chapter. Whereas the diffusion part was solved implicitly, using a final matrix system, an explicit Roe-scheme is used for solving the Euler equations. Because of the explicit nature of the solver, tagging implementation is slightly different compared to the diffusion part.

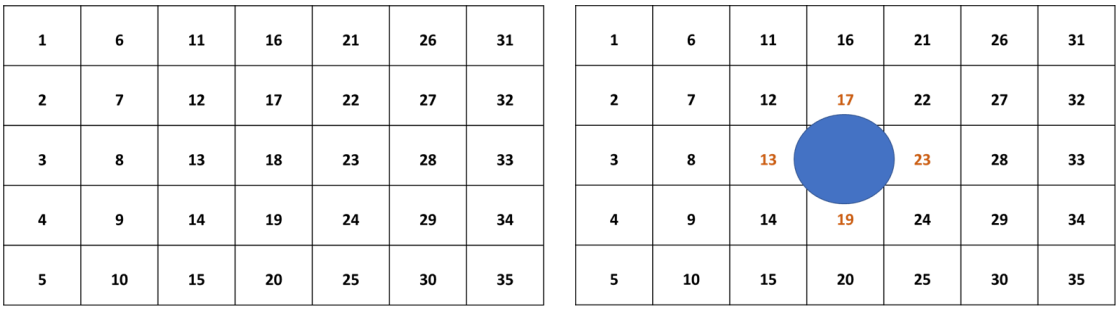
Steps for Tagging for 2-D inviscid compressible flow case

Step 1: Identify the cell type; inside or outside using the cross-product.

Step 2: Identify the intercepted cells using the neighbours check for the cells which are of interest i.e, if the inside of the geometry is solved, as in the case of shock tube, the intercepted cells will lie inside otherwise outside, nearest to the boundary.It is similar to the tagging algorithm defined in the Algorithm 1

Step 3: Coupling to the explicit solver

In order to understand the coupling, the data structure that has been generated for the tagging part has to be understood. The cells are numbered as described in Fig. 5.1(b) and this numbering is maintained throughout. At no point in the process of IB implementation, this numbering is altered including tagging and explicit solver call. Using the cell identification part, inside or outside, the cell numbers are stored in a list. For example, if we are solving outside of the geometry, a list of intercepted cells and a list of cells lying outside of the geometry, excluding the intercepted cells, are maintained. The explicit solver is used for this list and the the interpolation schemes are used for the intercepted cells. This process is repeated for each time step for a moving geometry.



(a) Cell Numbering without object

(b) Cell numbering with object with interface cells represented in orange color

Figure 5.1 Cell Numbering without and with object

The List for the External Cells, for which Roe scheme is used is presented in Table 5.1, the list for solid cells have been shown in Table 5.2 and the list for the interface cells have been presented in Table 5.3

Table 5.1 List of Fluid Cells; without interface Cells

Cell Number	X-Coordinate	Y-Coordinate	Z-Coordinate
1	X_1	Y_1	0
2	X_2	Y_2	0
3	X_3	Y_3	0
⋮	⋮	⋮	⋮
12	X_{12}	Y_{12}	0
14	X_{14}	Y_{14}	0
⋮	⋮	⋮	⋮
16	X_{16}	Y_{16}	0
⋮	⋮	⋮	⋮
20	X_{20}	Y_{20}	0
⋮	⋮	⋮	⋮
22	X_{22}	Y_{22}	0
24	X_{24}	Y_{24}	0
⋮	⋮	⋮	⋮
35	X_{35}	Y_{35}	0

The geometry is represented by a set of points, the density of the points can be dependent or independent on the curvature of the geometry. The tagging methodology is robust enough to handle both kinds of cases. An example for the geometry is presented in Fig. 5.2. Here, points representing the circle are equidistant.

Table 5.2 List of Solid Cells; Inside Geometry

Cell Number	X-Coordinate	Y-Coordinate	Z-Coordinate
18	X_{18}	Y_{18}	0

Table 5.3 List of Intercepted/Interface Cells

Cell Number	X-Coordinate	Y-Coordinate	Z-Coordinate
13	X_{13}	Y_{13}	0
17	X_{17}	Y_{17}	0
19	X_{19}	Y_{19}	0
23	X_{23}	Y_{23}	0

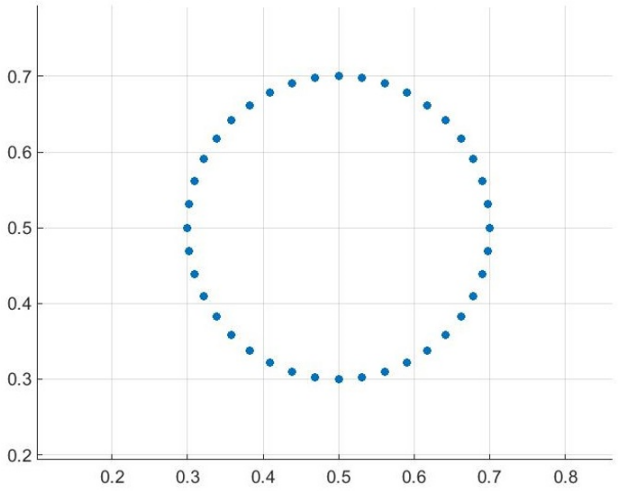


Figure 5.2 Geometry representation using points

The list of external cells is found out for the problem presented Fig. 5.1(b) and then the solver function loops on this list finding the fluxes, which is finally used to find the primitive variables. For interface cells interpolation scheme is used to directly find the variables, u, v, p and ρ. Thus, finding all the parameters for the fluid region.

5.2 Interpolation scheme

The interpolation scheme initially selected is bi-linear, as described in section 2.6 and is illustrated in Fig. 5.3. The difference in interpolation function implementation from the diffusion problem is the boundary condition. In order to implement the Neumann boundary

condition for pressure and density, it is assumed that the values calculated at the interface cells are equal to the boundary values, which helps in simplifying the calculation and it accurately represent the boundary condition. Therefore in Fig. 5.3, the primitive variables, p and ρ, calculated at point P are equal to those at Q. For U and V the interpolation scheme is same as that of section 2.6.

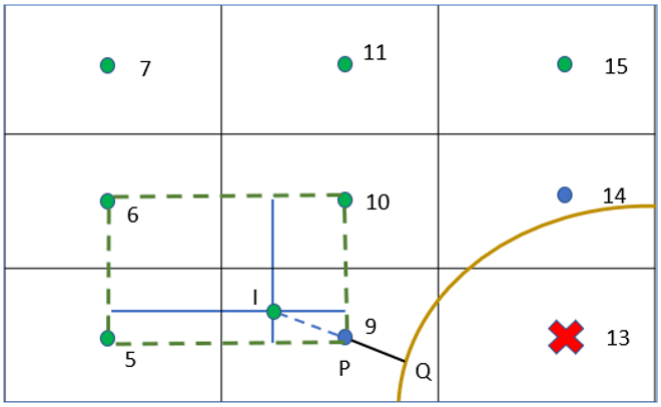


Figure 5.3 Bi-linear interpolation

5.3 2-D IBM with Roe scheme

The Immersed Boundary (IB) implementation for the 2-D induces complexity with respect to geometry representation, tagging and interpolation scheme implementation. For two-dimensional case, like the diffusion case, bi-linear interpolation scheme is implemented. The fluxes have to be calculated for the interior cells(if solving internal flows) and the primitive variables have to be calculated using the bi-linear interpolation for the interface cells.

In order to proceed in a step-wise manner, at first the Roe-Scheme is required to be converted to tackle 2-D problem. For a finite volume scheme, 2-D Roe fluxes are represented in a way described in Fig. 5.4. As described earlier, dimensional splitting has been used for calculating the fluxes and finally, the primitive variables p, u, v and ρ are calculated. The steps for Roe's scheme for the IB implementation are described below,

Steps for Roe Scheme implementation for IB method

Step 1: Calculate fluxes for the Fluid cells, as listed in Table 5.1

Step 2: Calculate the primitive variables using the fluxes calculated in step 1 using the Eqs. 5.4 and 5.5 explicitly.

For x sweep,

$$U_{i,j}^{n+1/2} = U_{i,j}^{n} + \frac{\Delta t}{\Delta x}[F_{i-\frac{1}{2},j}^{n} - F_{i+\frac{1}{2},j}^{n}] \tag{5.4}$$

and y-sweep:

$$U_{i,j}^{n+1} = U_{i,j}^{n+\frac{1}{2}} + \frac{\Delta t}{\Delta y}[G_{i-\frac{1}{2},j}^{n+\frac{1}{2}} - G_{i+\frac{1}{2},j}^{n+\frac{1}{2}}] \tag{5.5}$$

where U is given by, $\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}$

Step 3: Use interpolation scheme, bi-linear, for calculating p, u, v and ρ using the neighbouring fluid cells and the boundary condition.

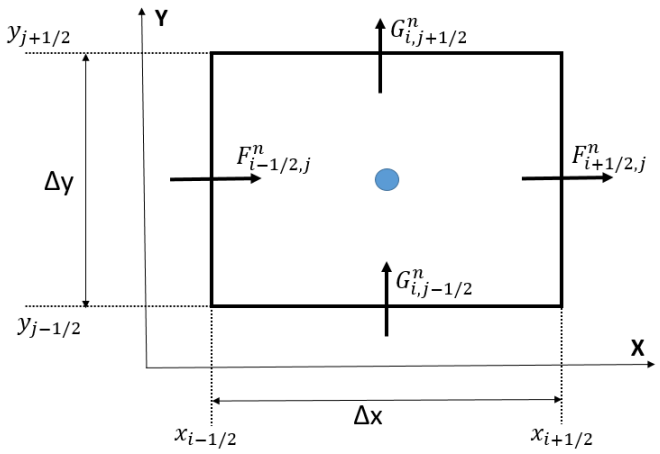


Figure 5.4 Finite Volume discretisation of Cartesian domain. Typical computing cell $I_{i,j}$ with four intercell fluxes

5.4 2-D IBM with stationary objects

The case for a shock tube with stationary walls can be used as a test case for the implementation of 2-D IB methodology. At first the shock tube is aligned with the coordinate axes. This simplifies the implementation with respect to the choice of the interpolation scheme. Linear interpolation can be implemented as the geometry boundary lines are either perpendicular or parallel to the mesh lines.

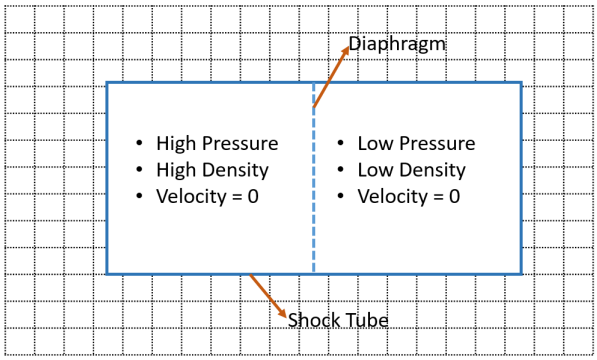
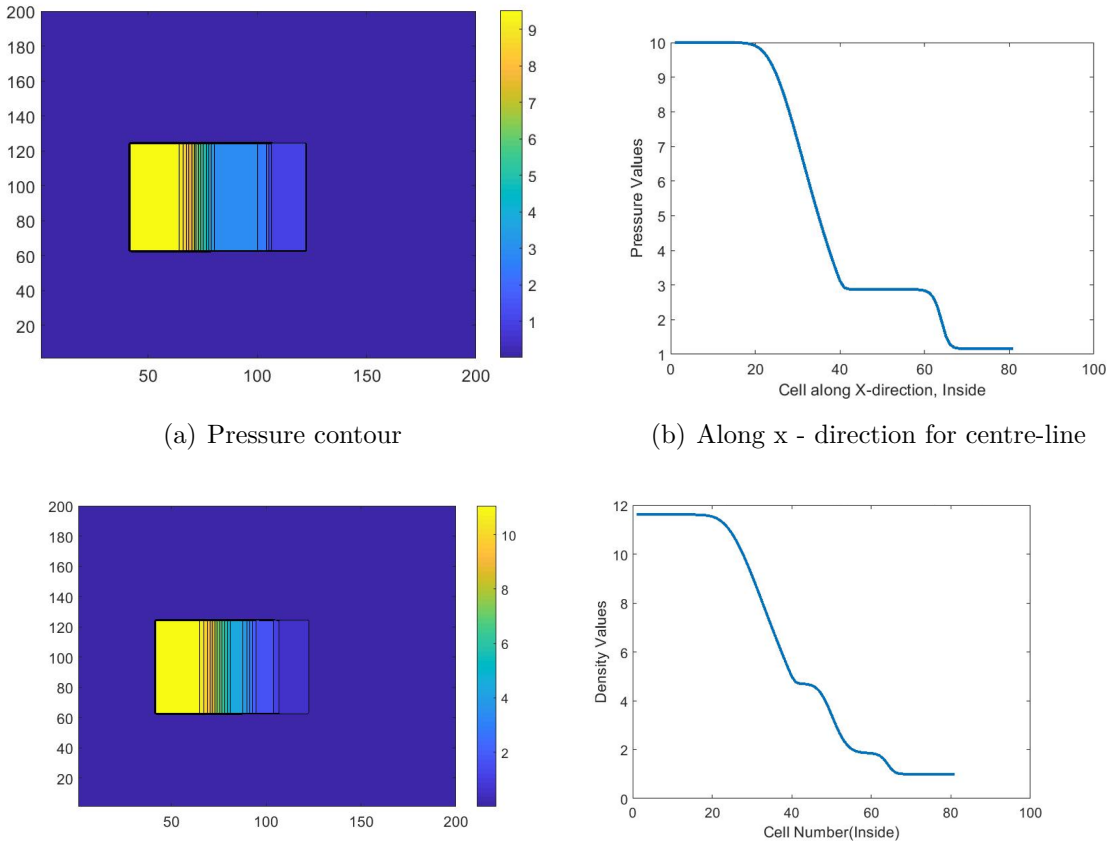


Figure 5.5 Shock Tube case schematic

The initial condition and the case set-up is presented in Fig. 5.5 where, a shock tube is immersed in the background mesh. The domain of interest is inside the shock tube. The left $Pressure = 10$, $Density = 11.16$, $X - Velocity = 0$ and $Y - Velocity = 0$ and correspondingly for the right side $Pressure = 1.0$, $Density = 1.16$, $X - Velocity = 0$ and $Y - Velocity = 0$. The results for this case after time $t = 0.0627 seconds$ is presented in Fig. 5.6(a), for pressure and Fig. 5.6(c) for density.

The next step is to include the bi-linear interpolation to replace the linear interpolation. This is done by inclining the shock tube such that the result for the interpolation is independent of the inclination. The problem schematic is presented in Fig. 5.7 in which the angle of inclination is fixed to 30^0.



(a) Pressure contour

(b) Along x - direction for centre-line

(c) Density contour

(d) Along x - direction for centre-line

Figure 5.6 a) Pressure contour, b) pressure variation along x-direction at the middle of the shock tube, c) Density contour and d) density variation along x-direction at the middle of the shock tube

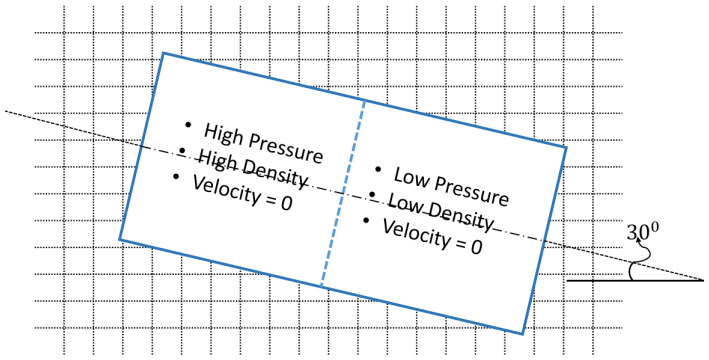


Figure 5.7 Inclined Shock Tube schematic with an inclination of 30^0

For the diffusion case, one final matrix was solved incorporating the coefficients corresponding

to the Interface cells with respect to the Bi-linear interpolation. Since the Roe Scheme is implemented explicitly, the individual implementation of the Bi-linear for each interface cell might give erroneous results. One particular case has been described in Fig. 5.8. To interpolate the values at IN_2, it will be required to use the value of IN_1. If the value of IN_1 is still unknown, this will induce error in the calculation of primitive variables at IN_2. Once there is an error in any one of the interface cell calculation, this can affect the interpolated values at rest of the interface cells while advancing in time. Therefore, there is a need to modify the way interpolation function is implemented.

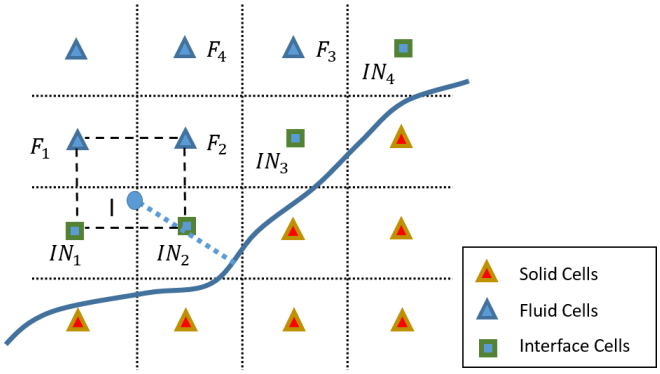


Figure 5.8 Problem with individual cell interpolation

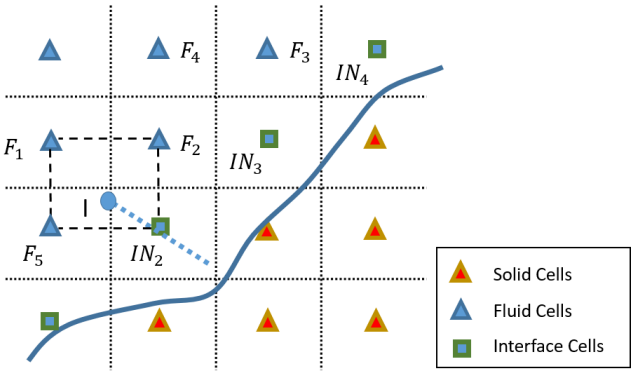


Figure 5.9 Implicit interpolation demonstration

The implicit definition of the interpolation function is one of the easiest way to solve the problem presented above. This yields a matrix system for all the interface cells is obtained and thus the values at all interface cells are calculated in one step. This will remove the problem observed in Fig. 5.8. Considering IN_2 and IN_3 in Fig. 5.9, the equation for bi-linear

interpolation for primitive variable represented as $f(x, y)$ can be written as,

$$f(x,y) = B_{11}f(x_{F_5}, y_{F_5}) + B_{21}f(x_{IN_2}, y_{IN_2}) + B_{12}f(x_{F_1}, y_{F_1}) + B_{22}f(x_{F_2}, y_{F_2}) \tag{5.6}$$

since, the values of pressure and density at I is equal to IN_2, Eq 5.7 will become,

$$f(x,y)(1 - B_{21}) = B_{11}f(x_{F_5}, y_{F_5}) + B_{12}f(x_{F_1}, y_{F_1}) + B_{22}f(x_{F_2}, y_{F_2}) \tag{5.7}$$

where, the coefficients B_{11}, B_{21}, B_{12} and B_{22} can be found by solving the following matrix system,

$$\begin{bmatrix} B_{11} \\ B_{21} \\ B_{12} \\ B_{22} \end{bmatrix} = \left(\begin{bmatrix} 1 & x_{F_5} & y_{F_5} & x_{F_5}y_{F_5} \\ 1 & x_{IN_2} & y_{IN_2} & x_{IN_2}y_{IN_2} \\ 1 & x_{F_1} & y_{F_1} & x_{F_1}y_{F_1} \\ 1 & x_{F_2} & y_{F_2} & x_{F_2}x_{F_2} \end{bmatrix}^{-1} \right)^T \begin{bmatrix} 1 \\ x \\ y \\ xy \end{bmatrix} \tag{5.8}$$

For a two point system, the implicit interpolation scheme can be implemented easily. Considering interface cells IN_3 and IN_2 and assuming local coefficients for IN_3 as C_{11}, C_{21}, C_{12} and C_{22}, the matrix system to find the primitive variables looks like,

$$\begin{bmatrix} (1 - B_{21}) & 0 \\ 0 & (1 - C_{21}) \end{bmatrix} \begin{bmatrix} f(x_{IN_2}, y_{IN_2}) \\ f(x_{IN_3}, y_{IN_3}) \end{bmatrix} = \begin{bmatrix} B_{11}f(x_{F_5}, y_{F_5}) + B_{12}f(x_{F_1}, y_{F_1}) + B_{22}f(x_{F_2}, y_{F_2}) \\ C_{11}f(x_{F_2}, y_{F_2}) + C_{12}f(x_{F_4}, y_{F_4}) + C_{22}f(x_{F_3}, y_{F_3}) \end{bmatrix} \tag{5.9}$$

Therefore, for a problem with N interface cells, the left side matrix will be $[N * N]$.

One more interpolation approach can be used namely, linear interpolation using three points. This again has to be implemented implicitly, with a matrix system. Since this uses only three points, one being the interface cell, it is easier to implement compared to bi-linear interpolation and is less computationally expensive.

5.4.1 Boundary condition implementation

The Neumann boundary condition for pressure and density is very straight forward. As shown in Fig. 5.10, the values are interpolated at point I, assuming that the primitive variable value does not change in the normal direction such that *pressure* or *density* at point I is same as that of P and Q. For velocity, in order to implement slip boundary condition, a two-step procedure is implemented:

1. The image point I is removed from the interpolation stencil of Fig. 5.10 and the velocity is extrapolated at point P using the points 5, 6 and 10.

2. The velocity U is divided into two components, namely tangent and normal to the surface as shown in Fig. 5.11(a). The normal component is neglected and the tangential component is broken into x and Y-directions contributing to the velocity in those directions accordingly. This is repeated for V.

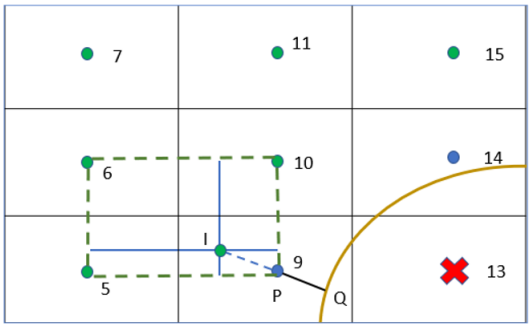
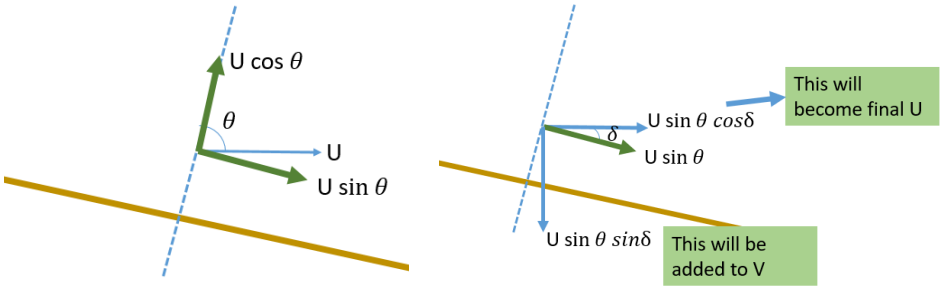


Figure 5.10 Bi-linear interpolation



(a) Normal and tangential velocity component

(b) Final velocity components

Figure 5.11 Slip boundary condition implementation

5.4.2 Results and discussions

Some of the results obtained are presented in Fig. 5.13. It is interesting to note that the quality of the solution depends on the interpolation scheme selected and the way of its implementation. The convergence for the inclined shock tube has been represented in Fig. 5.12.

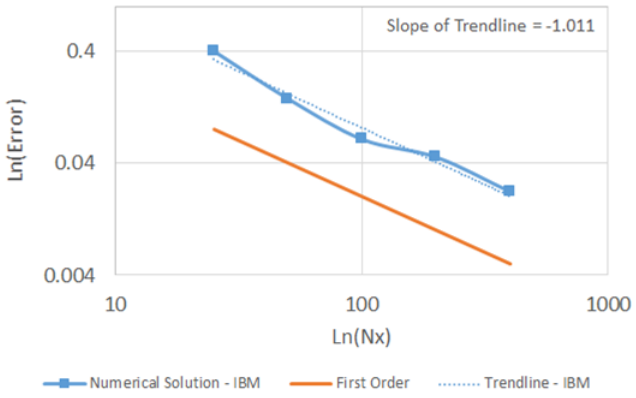
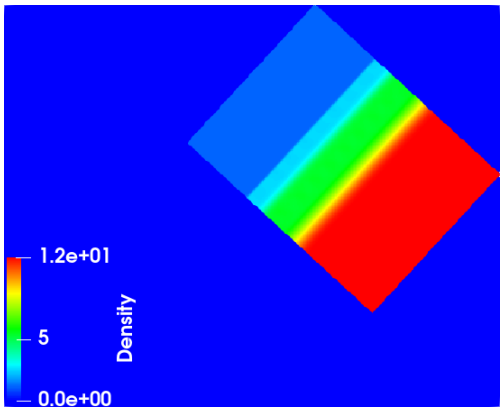
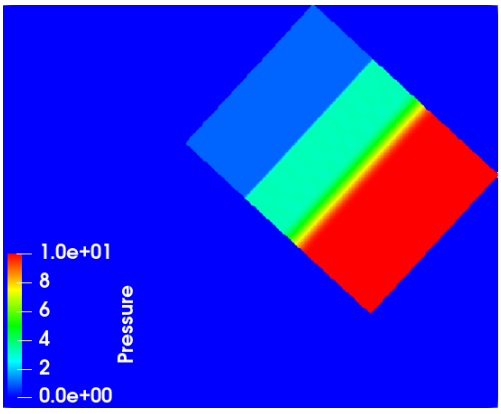


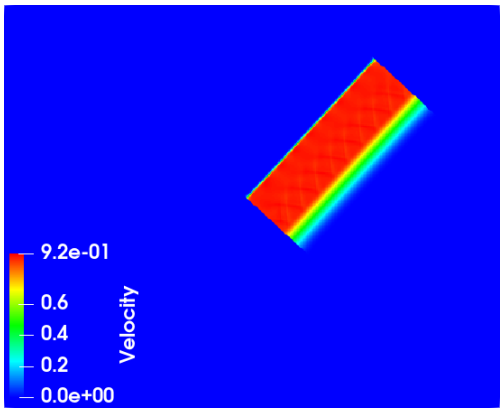
Figure 5.12 Pressure convergence for inclined shock tube with implicit bi-linear interpolation for an angle of rotation of 140^0



(a) Density variation inside the shock tube



(b) Pressure variation inside the shock tube



(c) Velocity variation inside the shock tube

Figure 5.13 Pressure, Velocity and Density contours for inclined shock tube for 500 X 500 background mesh with the angle of inclination of 140^0

5.5 Problems with shock-tube for corner cells

For the two-dimensional case, the problem arises with the sharp corner cells, especially with the four corners of the shock tube problem solved in the previous section. The non-availability of the fluid cells to fill the coefficients in the implicit bi-linear equation matrix, leads to the inaccurate calculation of the primitive variables for these cells, as demonstrated in Fig 5.14. Here, in order to implement bi-linear, S_1 should be either a fluid cell or an interface cell. Conditions like this makes bi-linear interpolation a non-ideal choice for sharp corner problems.

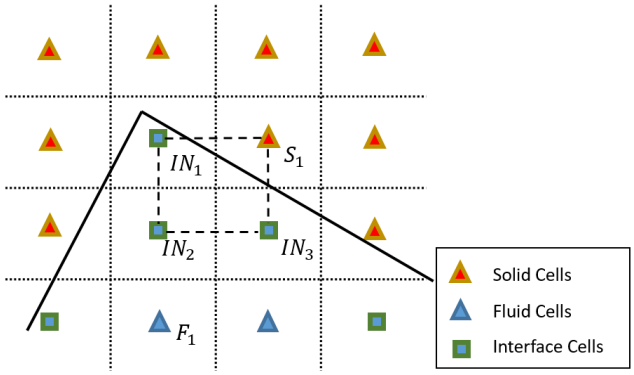


Figure 5.14 Demonstration of corner cell problem for Bi-linear interpolation

In order to get rid of this problem, one suggestion is to initialise the four corner cells with the expected values for each time step. One problem with this is when the wave reaches the boundary, primitive variable values are not be known beforehand. Another approach includes, using a linear interpolation with three points. Even at the corners, it is ensured that three nearest points are available from the fluid and interface regions and the interpolation is done implicitly. Mathematically, it is similar to the bi-linear interpolation expressed in Eq. 5.8 and Fig. 5.9 except that the fourth point has to be dropped.

5.6 2-D IBM with moving boundaries

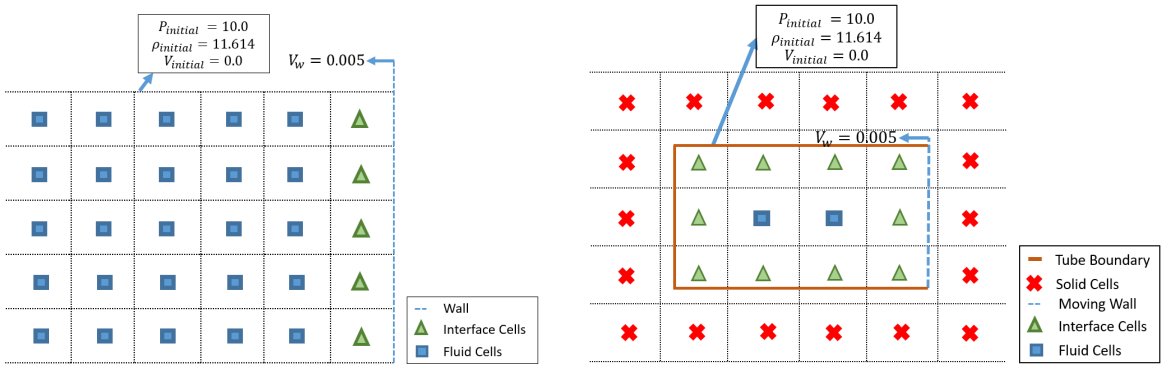
Two-dimensional moving boundaries problems for IB method is presented with respect to the shock tube. The reason for the selection of the problem is the ease to implement IB methods and check for the errors and bugs, whenever encountered. As described in 1-D inviscid compressible flows, there are two different possibilities encountered for moving boundaries.

1. The cell is an interface cell, and with the movement of the boundary is converted into a solid cell.

2. The cell which contains the boundary, is a solid cell and gets converted into a fluid cell because of boundary movement.

5.6.1 Moving Boundaries: Fluid-to-Solid conversion

The fluid-to-solid conversion implementation is straightforward and does not require any special treatment for IB implementation. Fluid cell to solid cell conversion is implemented in two steps. The first problem statement consists of a tube with initial values of pressure, density and velocity as described in Fig. 5.15(a), where the right wall of the tube moves towards left with a velocity of $V_{wall} = 0.005\ m/s$. As per the schematic, the IB method is implemented for the right wall only. Moreover the movement if perpendicular to the mesh, therefore linear interpolation is enough to interpolate the values for the interface cells.



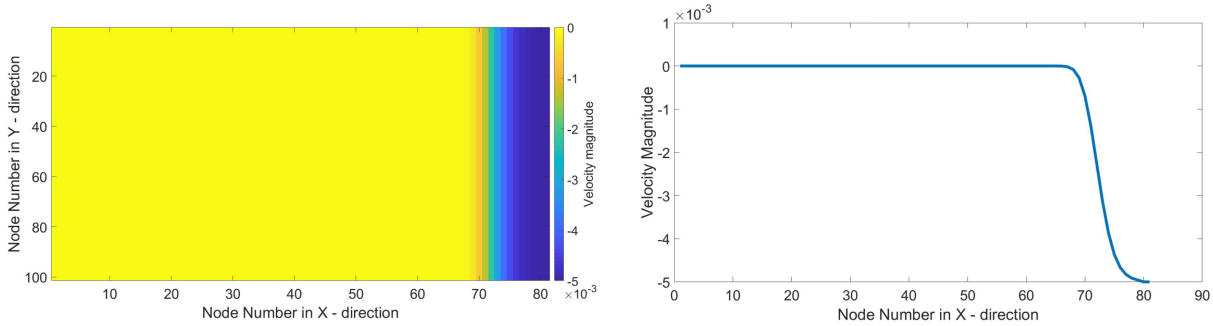
(a) Wall movement for the whole domain

(b) Tube immersed in the domain

Figure 5.15 Wall movement towards left

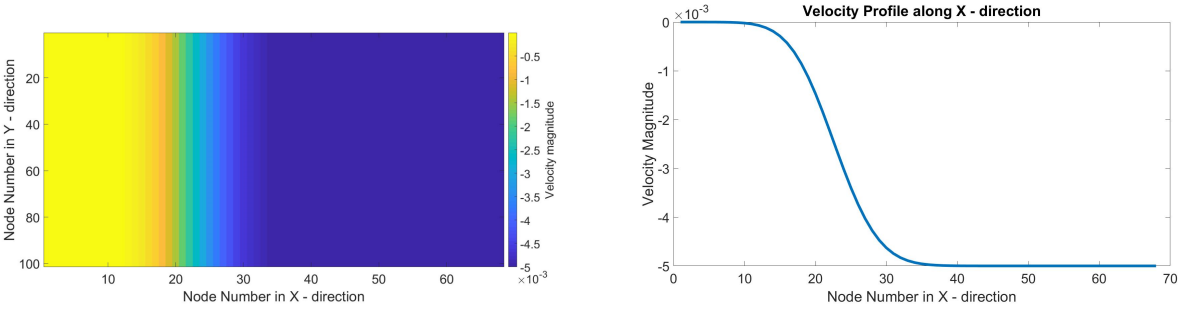
Numerical solution for this problem is presented in Fig. 5.16 and Fig. 5.17.

The second problem is defined in Fig. 5.15(b) where, the whole tube is immersed in an outer domain. Since the area of interest is inside the tube, everything outside the tube is neglected. The tube is initialised with the values of $p = 10$, $\rho = 11.614$ and $u = 0$. Numerical solution for this problem is presented in Fig. 5.18.



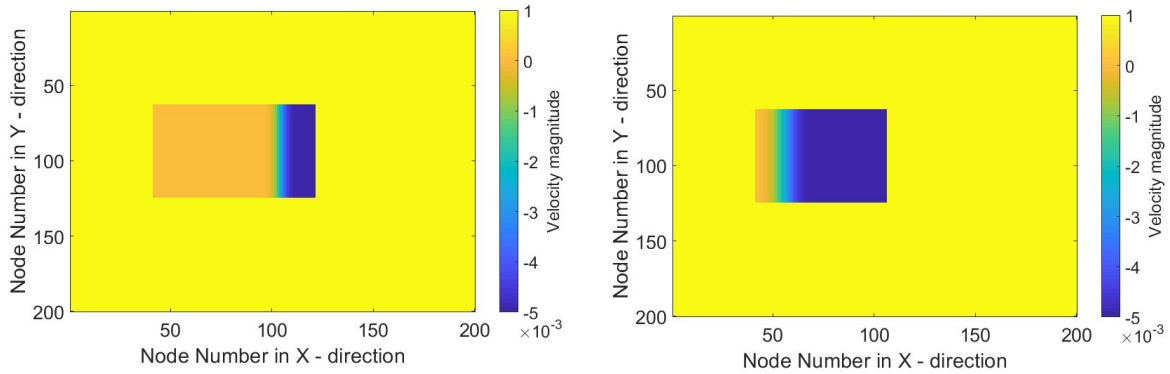
(a) Velocity profile (b) Velocity along x - direction for centre-line

Figure 5.16 Wall movement towards left at t = 0.08 seconds



(a) Velocity profile (b) Velocity along x - direction for centre-line

Figure 5.17 Wall movement towards left at t = 2 seconds



(a) Wall movement towards left: at time t = 0.08 seconds (b) Wall movement towards left: at time t = 0.3 second

Figure 5.18 Wall movement towards left: Tube immersed in domain

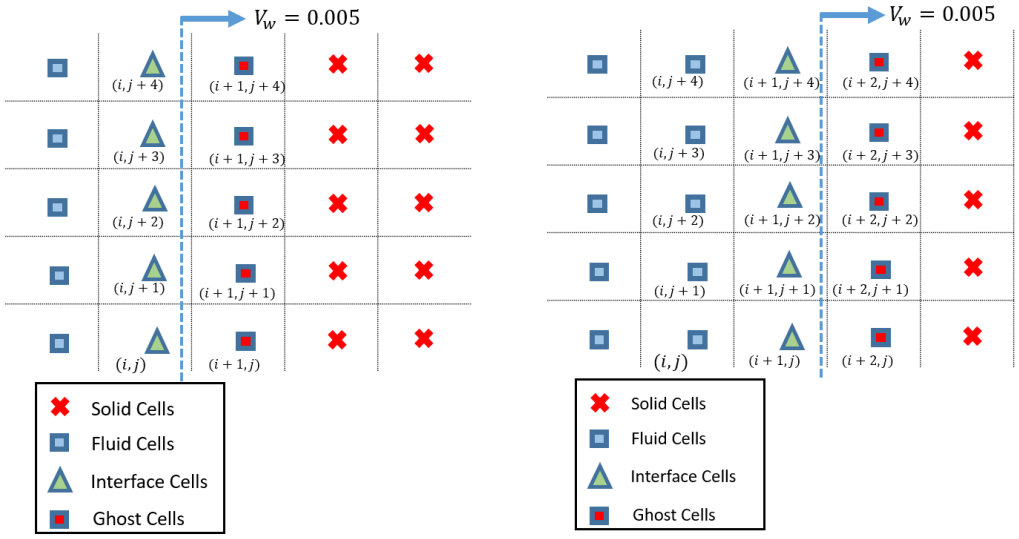
5.6.2 Moving Boundaries: Solid-to-Fluid conversion

For a typical 2-D problem, solid cell to fluid cell conversion poses a challenge. In Fig. 5.19(a), the cell and wall status is shown for at t^k time step and as the wall moves towards right with the specified velocity for the next time step, t^{k+1}, the cell and wall status is as per Fig. 5.19(b). This transition involves two kinds of cell conversion,

1. **Interface-to-Fluid :** The interface cells in Fig. 5.19(a) are converted into fluid cells in Fig. 5.19(b). This pose a major challenge with respect to the IB implementation as the values at time step t^{k+1} depends on the value calculated at time step t^k. For all the newly defined fluid points, the part of the Euler equations that depends on the derivatives from the previous time step will not be correct. Therefore, a special treatment for the primitive values at these cells are required.

2. **Solid-to-Fluid :** Solid cells (Ghost cells) in Fig. 5.19(a) are converted into interface cells in Fig. 5.19(b). This, does not pose any problem with respect to the IB implementation as the solution for time step t^{k+1} will be reconstructed using an interpolation scheme and does not depend on its value at the previous time step t^k

In order to deal with the above mentioned problem, one approach called Field Extension technique has been described in [4] and this approach is implemented for simplified wall movement problems in the next sections. One novel idea of Two-level interpolation is introduced and is used to solve the same problems which are solved using Field Extension in order to verify the validity of the novel idea.

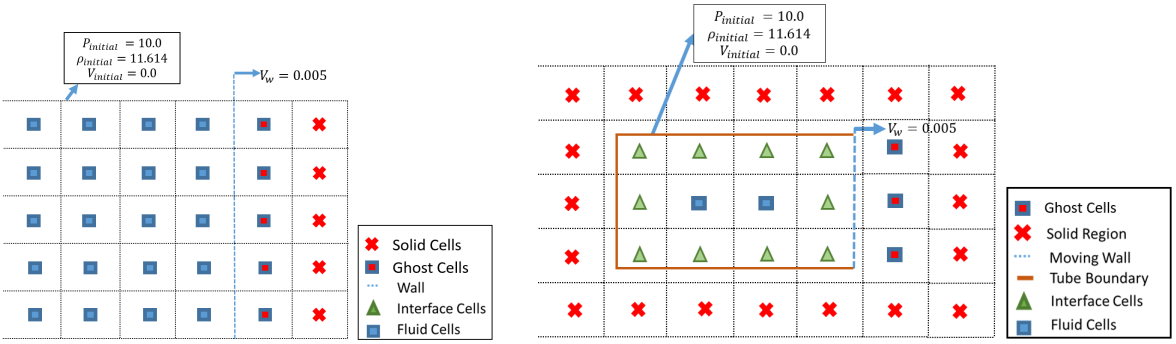


(a) Cell status at time t^k seconds (b) Cell status at time t^{k+1} second

Figure 5.19 Wall movement towards right: cell status at a) t^k time step and b) t^{k+1} time step

For Field Extension as well as for Two level interpolation, IB method is implemented in two steps. The problem schematic for fluid-to-solid conversion is described in Fig. 5.20(a) where, there is a wall on the right side of the tube. Once, this wall starts moving the solid cells that lie to the right of the wall becomes fluid cells, in an iterative manner. Since the wall lies vertically, along y-axis, linear interpolation is enough to interpolate the values at interface cells.

The schematic for the second problem is presented in Fig. 5.20(b), where the tube is immersed in an outer domain with right wall moving towards right. Here again, the solid cells get converted into fluid cells with the movement of the wall.



(a) Wall movement for the entire domain (b) Wall movement for tube immersed in an outer domain

Figure 5.20 Problem schematic: Wall movement towards right

Field Extension Implementation in 2-D

Field Extension approach for a 2-D problem, is similar to the one presented for 1-D problems in the previous chapter in which velocity, pressure and density fields are extended to the solid phase at the end of each time step. This involves, extrapolating the primitive variables to the Ghost cells, as described in Fig. 5.19. This way, not only the values of the primitive variables but also their derivatives will have physical values, as described in [4].

In [4], the field extension is described as using linear interpolation(extrapolation) using three points as shown in Fig. 5.21. It should be noted here that the pseudo-fluid points are nothing but the ghost cells. In the present case, the moving boundary is aligned with the mesh, therefore a simple linear interpolation is enough to simulate the case. Therefore for Fig. 5.20, a linear interpolation using the nearest fluid cell, interface cell and the boundary point is done. It should be noted that, the line joining these three points is always normal to the geometry and passes through the nearest ghost cell.

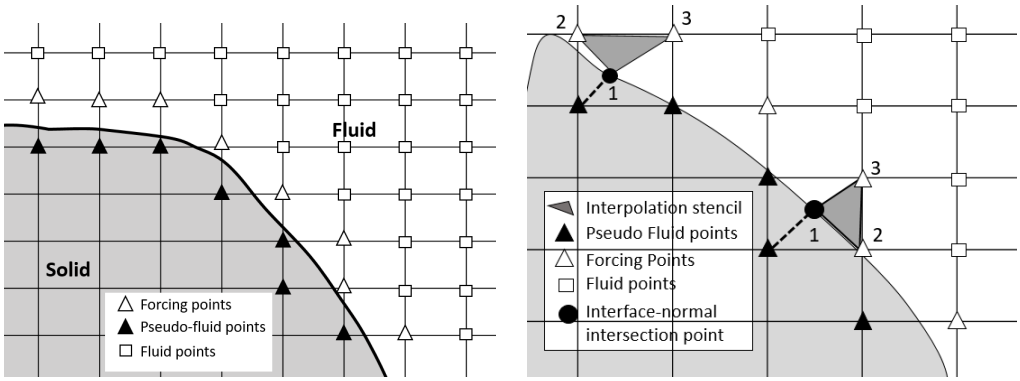
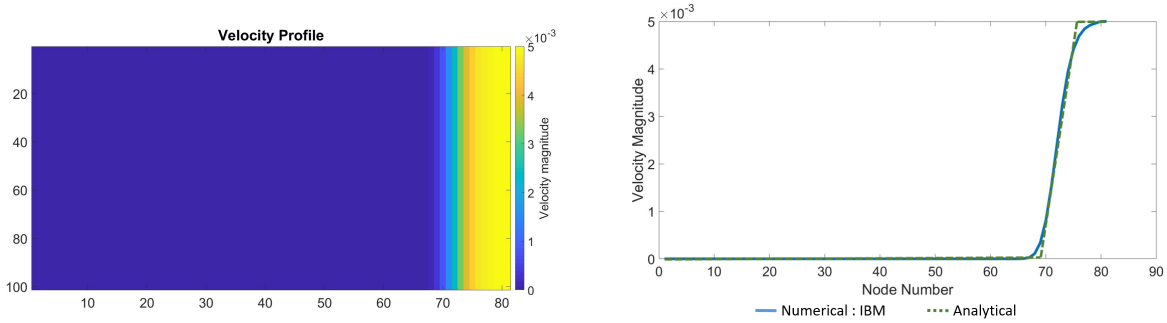


Figure 5.21 The field-extension procedure: (a) identification of the pseudo-fluid points/ghost cells where the solution will be extended; (b) possible extrapolation stencils [4]

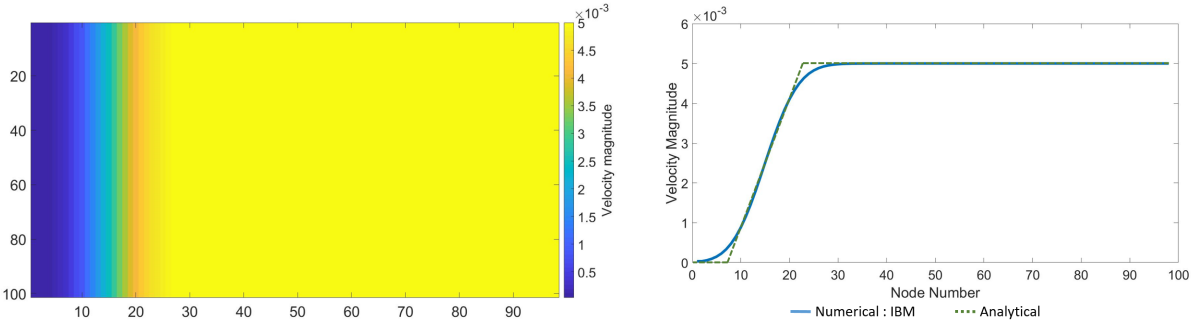
The first case which is represented in Fig. 5.20(a) is simulated with the field extension technique and the results have been reported in Fig. 5.22 and Fig. 5.23 for two different time intervals. The velocity profile is smooth and has no fluctuations at any point along the x-direction, which is as expected.

The results for the case presented in Fig. 5.20(b) in which the tube is immersed in the domain is presented in Fig. 5.24 for two different time steps.



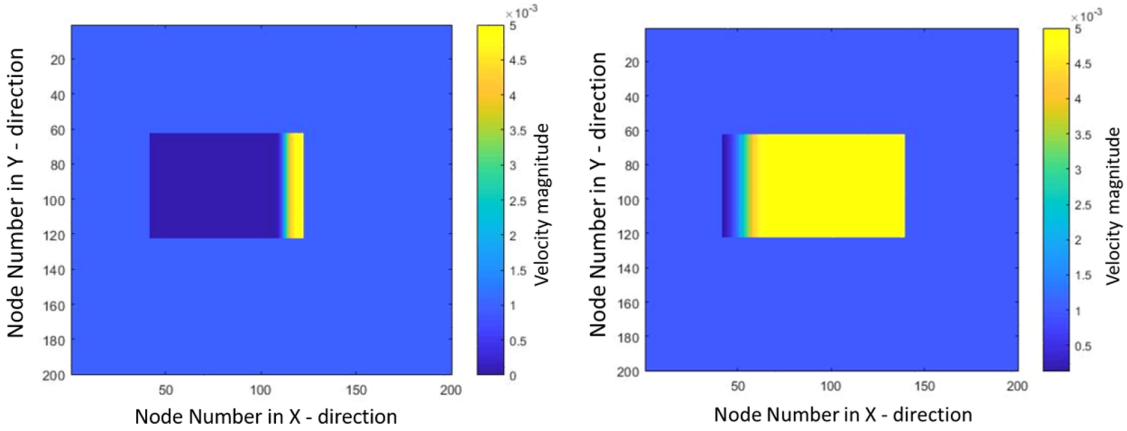
(a) For the entire domain (b) Along x - direction for a particular y

Figure 5.22 Wall movement towards right: Whole domain at t = 0.08 seconds



(a) For the entire domain (b) Along x - direction for a particular y

Figure 5.23 Wall movement towards right: Whole domain at t = 6 seconds



(a) At time t = 0.08 second (b) At time t = 3 seconds

Figure 5.24 Wall movement towards right: Tube immersed in domain

Two-level interpolation

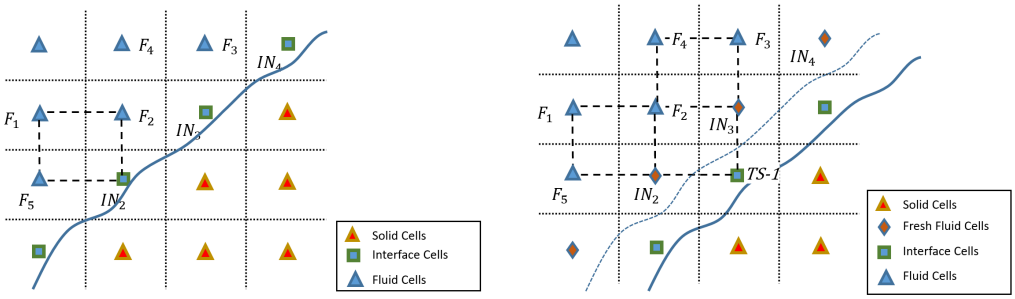
The two-level interpolation technique, for a 2-D problem, include,

1. Keeping track of the interface cells at t^k time step

2. Use interpolation scheme for the freshly cleared cells at t^{k+1} time step, which were ghost cells at t^k, as well as for the cells which were interface cells at t^k.

Therefore in Fig. 5.19, the interface cells in Fig. 5.19(a) and the interface cells in Fig. 5.19(b) have to be interpolated between the fluid points and the boundary boundary. In other words, the primitive variables for cells till $i-1^{th}$ column is calculated using finite volume Roe scheme and interpolated between the fluid points and the boundary points for i^{th} and $i+1^{th}$ columns. For a typical 2-D problem, similar to the one presented in Fig. 5.21, two level interpolation is presented in Fig. 5.25.

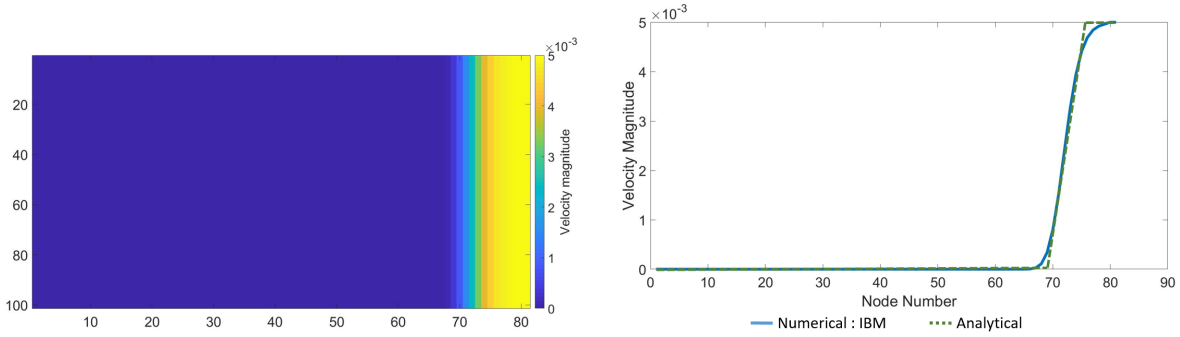
The stencils have been shown with dotted lines with respect to bi-linear interpolation. It can be noted that, for freshly cleared cell TS_1, in order to interpolate primitive variables values, calculated values of IN_2 and IN_3 are used. This might induce some error as these values are themselves interpolated from the fluid cells. The idea here is to show that this novel method works and produce good results. These induced errors can be eliminated by using more sophisticated interpolation schemes like Radial Basis functions or Least-square.



(a) At time t^k (b) At time t^{k+1}

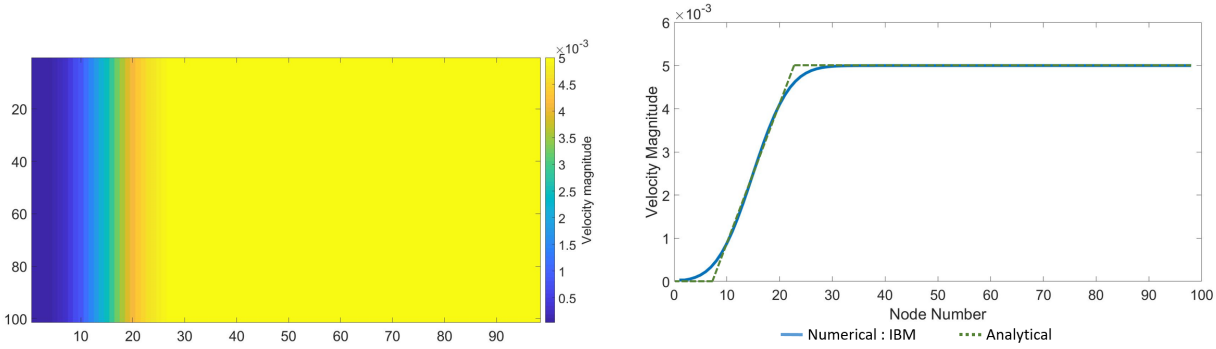
Figure 5.25 Generalised Two-level interpolation for two dimension

The two problems solved using the field-extension method shown in Fig. 5.20, are solved using two-level as well so as to compare the two cases. The results for the first case is presented in Fig. 5.26 and Fig. 5.27. The result for the second problem is presented in Fig. 5.28.



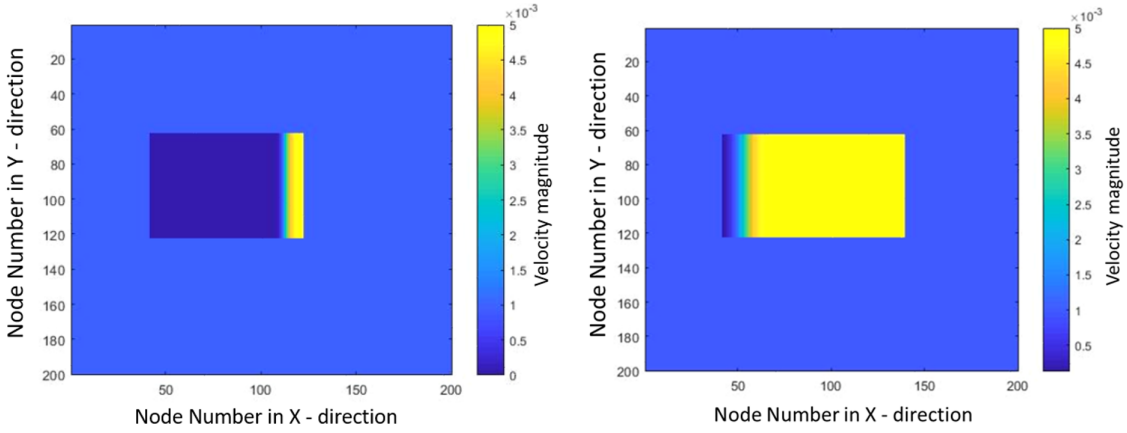
(a) For the entire domain (b) Along x - direction for a the centre line

Figure 5.26 Solid-to-Fluid: Whole domain at t = 0.08 seconds using two-level interpolation



(a) For the entire domain (b) Along x - direction for a the centre line

Figure 5.27 Solid-to-Fluid: Whole domain at t = 0.6 seconds using two-level interpolation



(a) At time t = 0.08 second (b) Along x - direction for a the centre line

Figure 5.28 Solid-to-Fluid: Tube immersed in domain using Two-level interpolation

5.6.3 Advantages of methods implemented

Two-level interpolation

Though, problems solved using two-level interpolation for comparison with field-extension are simple but complex enough to validate that the method works. One of the main expected advantage of two-level method is to deal with thin geometries or part of the geometry which is thinner than the mesh size. These problems are encountered especially when airfoils cases are solved. The trailing edge of the airfoils are thin and for a given mesh size, there is a very high probability that there are no cell centres inside the thin part. This makes methods dependent on Ghost cell inefficient, or unable at times, to capture such geometries or parts of the geometry. This problem is eliminated when the numerical scheme is independent on the solid points and are dependent only on the fluid points as described in Fig. 5.29.

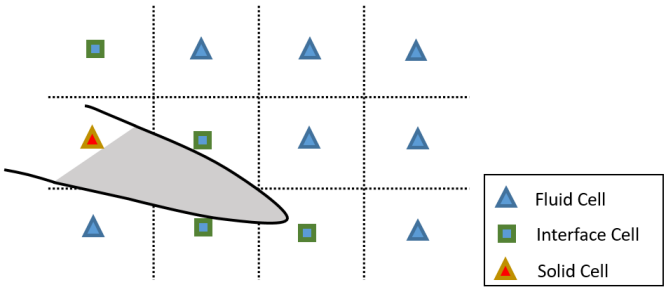


Figure 5.29 Demonstration of inefficiency of ghost cell method to capture the shaded portion of geometry

Implicit interpolation

The implicit interpolation scheme implemented for the rotated shock-tube problem solves the problems encountered in [4] where, there is a constraint that the interpolation stencil should not include another interface cell. This is achieved by gradually moving the virtual point in the interpolation stencil further away from the boundary along the normal direction as represented in Fig. 5.30. There is a chance that this can induce local errors which can propagate with time.

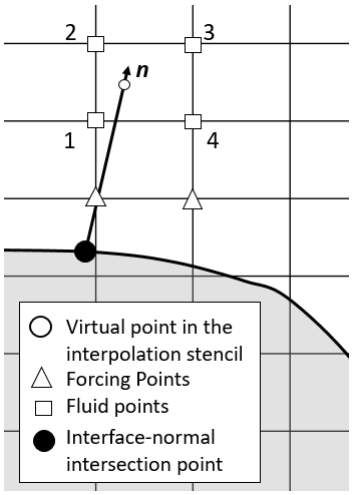


Figure 5.30 Problem with explicit interpolation as used in [4]

With the use of implicit interpolation, this problem is not encountered.

5.7 Conclusion

The modeling of moving geometry, which is a critical part with respect to IB implementation has been solved for 2-D problems. The implementation of fluid-to-solid conversion is straightforward. The problem for the Solid cell to Fluid Cell conversion has been dealt in two different ways. The first one is as per the [4], Field Extension technique and the second one is the "Two Level" interpolation, which is a novel idea. The results obtained using two level of interpolation is as good as the Field Extension which is evident from the results obtained above. Moreover, the use of more sophisticated interpolation schemes is expected to increase the effectiveness of the method proposed for more complex problems.

CHAPTER 6 CONCLUSION

6.1 Summary of Works

The work presented here deals with the IB implementation for two different kinds of problems, diffusion and Euler Equations. The diffusion case, being simpler, act as a verification and validation for the implementation of the IB method and helps in finding out the challenges and the corresponding solutions.

The main contribution of the diffusion case is the problem identification for the wrong normal calculation, which helped in correct implementation of the Boundary Conditions for the compressible flow cases. Moreover, the interpolation/reconstruction has been implemented using bi-linear interpolation.

For Euler equation cases, present methods are explored especially with respect to one-sided interpolation. The complexity of the problems are increased in a step-wise manner starting from 1-D to 2-D problems. It is ensured that the geometry is captured correctly, even when the size of the geometry is less than the mesh size.

For moving boundaries, one of the present methods is explored which relies on ghost cell and it has been found that the group of methods relying on the Ghost Cells are ineffective to deal with the geometries thinner than the mesh size. Therefore, a new method is introduced which relies on one-sided interpolation even for moving boundaries which is achieved by considering two cells from the fluid side for interpolation.

Various reconstruction schemes are tried for interpolating the values at the interface cells. For the diffusion equation, a bi-linear scheme is implemented where, a system of equations are solved for the Laplace problem and the coefficients obtained from the interpolation are inserted in the same system of equations. Here, the interpolation is implicitly coupled to the solver.

Linear interpolation has been used for the 1-D Euler equations. Since the Euler equations are solved explicitly using Roe's scheme, the bi-linear interpolation can not be implemented for 2-D Euler case, the way it was solved for the diffusion equation. Instead, an implicit interpolation has been used, otherwise the interpolation provides erroneous results. The idea of implicit implementation of the interpolation schemes also helps in solving the problems reported in [4], where the author moves away from the boundary in search of the fluid points, thus inducing inaccuracy during interpolation.

In some cases, like the edges of the shock tube, bi-linear interpolation cannot be applied because of the non-availability of the interpolation stencil. Instead, a linear interpolation using three points needs to be implemented which get rid of the problems faced by bi-linear implementation. These problems need to be analyzed before selecting the interpolation scheme.

6.2 Limitations

Present work has certain limitations which includes:

1. Geometry Representation: Adequate number of points should be used for defining the geometry, especially with respect to the curvature of the geometry. For example, to represent the leading edge of the airfoil there should be sufficient number of points to do it. This is also dependent on the mesh size being used to solve the problem.

2. Sharp Corners: For the sharp corners, as in the case of the inclined chock tube, the reconstruction scheme can be ambiguous, depending on its way of implementation.

3. Two different Boundary condition in close proximity: Again this can be problematic with respect to the IB implementation finding the weights for the interface cells with respect to the Boundary conditions is big problem. This can be amplified when the two different boundaries form a sharp edge.

6.3 Future Research

Ongoing and future work includes implementation of different Interpolation schemes namely Radial Basis Function, Least-Square and Inverse distance weighting and comparing all these interpolation scheme with respect to ease of implementation and accuracy.

Moreover, for the compressible flow case, problems including supersonic inlet with simple objects, eg. cylinder, will be implemented in order to compare the solution with various experimental solutions and previously cited cases.

REFERENCES

[1] G. I. R. Mittal, "Immersed boundary methods," *Annual Review of Fluid Mechanics*, vol. 37, pp. 239–261, 2005.

[2] F. Nikfarjam, Y. Cheny, and O. Botella, "The ls-stag immersed boundary/cut-cell method for non-newtonian flows in 3d extruded geometries," *Computer Physics Communications*, vol. 226, pp. 67 – 80, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010465518300109

[3] E. F. Toro, *Riemann solvers and numerical methods for fluid dynamics: A practical introduction*, 2009.

[4] E. B. Jianming Yang, "An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries," *Journal of Computational Physics*, vol. 215, pp. 12, 40, 2005.

[5] C. S. Peskin, "Flow patterns around heart valves: A numerical method," *Journal of Computational Physics*, vol. 10, no. 2, pp. 250, 271, 1972.

[6] M. Kumar, S. Roy, and M. S. Ali, "An efficient immersed boundary algorithm for simulation of flows in curved and moving geometries," *Computers & Fluids*, vol. 129, pp. 159–178, Apr. 2016. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0045793016300275

[7] Z. Chen *et al.*, "Immersed boundary-simplified thermal lattice boltzmann method for incompressible thermal flows," *Physics of Fluids*, vol. 32, no. 1, 2020.

[8] M. K. R. S. Md. Sujaat Ali, Manish Jaiswal, "A computational study of flow over an airfoil at low reynolds number using immersed boundary method," in *42nd National Conference on Fluid Mechanics and Fluid Power*, 2015.

[9] M. Linnick and H. Fasel, "A high-order immersed boundary method for unsteady incompressible flow calculations," in *41st Aerospace Sciences Meeting and Exhibit*, 2003.

[10] F. Muldoon and S. Acharya, "Mass conservation in the immersed boundary method," in *Proceedings of the American Society of Mechanical Engineers Fluids Engineering Division Summer Conference*, vol. 1 Part A, 2005, pp. 411 – 419.

[11] M. Kumar and S. Roy, "A sharp interface immersed boundary method for moving geometries with mass conservation and smooth pressure variation," *Computers and Fluids*, vol. 137, pp. 15 – 35, 2016.

[12] R. Boukharfane *et al.*, "A combined ghost-point-forcing / direct-forcing immersed boundary method (ibm) for compressible flow simulations," *Computers and Fluids*, vol. 162, pp. 91 – 112, 2018.

[13] H. Choung, V. Saravanan, and S. Lee, "Jump-reduced immersed boundary method for compressible flow," *International Journal for Numerical Methods in Fluids*, 2020.

[14] D. Boffi *et al.*, "Mass preserving distributed langrage multiplier approach to immersed boundary method," in *Computational Methods for Coupled Problems in Science and Engineering V - A Conference Celebrating the 60th Birthday of Eugenio Onate, Coupled Problems 2013*, 2013, pp. 323 – 334.

[15] A. M. Roma, "A Multilevel Self-Adaptive Version of the Immersed Boundary Method," PhD thesis, Courant Institute of Mathematical Sciences - New York University, United States, Jan. 1996, university Microfilms #9621828.

[16] S. Popinet, "Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries," *Journal of Computational Physics*, vol. 190, no. 2, pp. 572–600, Sep. 2003. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0021999103002985

[17] A. van Nimwegen, K. Schutte, and L. Portela, "Direct numerical simulation of turbulent flow in pipes with an arbitrary roughness topography using a combined momentum–mass source immersed boundary method," *Computers Fluids*, vol. 108, pp. 92 – 106, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045793014004307

[18] S. Roy and S. Acharya, "Scalar mixing in a turbulent stirred tank with pitched blade turbine: Role of impeller speed perturbation," *Chemical Engineering Research and Design*, vol. 90, no. 7, pp. 884 – 898, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0263876211003947

[19] J. Nam and F. Lien, "Assessment of ghost-cell based cut-cell method for large-eddy simulations of compressible flows at high reynolds number," *International Journal of Heat and Fluid Flow*, vol. 53, pp. 1 – 14, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0142727X15000089

[20] F. Picano, W.-P. Breugem, and L. Brandt, "Turbulent channel flow of dense suspensions of neutrally buoyant spheres," *Journal of Fluid Mechanics*, vol. 764, pp. 463 – 487, 2015.

[21] A. Prosperetti, "Life and death by boundary conditions," *Journal of Fluid Mechanics*, vol. 768, p. 1–4, 2015.

[22] W. S. H.SUdaykumar, R Mittal, "Computation of solid–liquid phase fronts in the sharp interface limit on fixed grids," *Journal of Computational Physics*, vol. 153, pp. 535, 574, 1999.

[23] P. H.S.Udaykumar, R.Mittal, "A sharp interface cartesian grid method for simulating flows with complex moving boundaries," *Journal of Computational Physics*, vol. 174, pp. 345, 380, 2001.

[24] L. Zhu and C. S. Peskin, "Interaction of two flapping filaments in a flowing soap film," *Physics of Fluids*, vol. 15, no. 7, pp. 1954–1960, 2003. [Online]. Available: https://doi.org/10.1063/1.1582476

[25] . M. A. Fauci, L. J., "Sperm motility in the presence of boundaries," *Bulletin of mathematical biology*, no. 57(5), p. 679–699, 1995.

[26] K. Khadra *et al.*, "Fictitious domain approach for numerical modelling of navier-stokes equations," *International Journal for Numerical Methods in Fluids*, vol. 34, no. 8, pp. 651 – 684, 2000.

[27] N. S. Dhamankar, G. A. Blaisdell, and A. S. Lyrintzis, "Implementation of a wall-modeled sharp immersed boundary method in a high-order large eddy simulation tool for jet aeroacoustics," vol. 0, 2016.

[28] R. Boukharfane *et al.*, "A combined ghost-point-forcing / direct-forcing immersed boundary method (ibm) for compressible flow simulations," *Computers and Fluids*, vol. 162, pp. 91 – 112, 2018.

[29] J. Liu *et al.*, "A new immersed boundary method for compressible navier-stokes equations," *International Journal of Computational Fluid Dynamics*, vol. 27, no. 3, pp. 151 – 163, 2013.

[30] Y. Zhang and C. Zhou, "An immersed boundary method for simulation of inviscid compressible flows," *International Journal for Numerical Methods in Fluids*, vol. 74, no. 11, pp. 775 – 793, 2014.

[31] K. Luo *et al.*, "A ghost-cell immersed boundary method for simulations of heat transfer in compressible flows under different boundary conditions," *International Journal of Heat and Mass Transfer*, vol. 92, pp. 708 – 717, 2016.

[32] M. Al-Marouf and R. Samtaney, "A versatile embedded boundary adaptive mesh method for compressible flow in complex geometry," *Journal of Computational Physics*, vol. 337, pp. 339–378, May 2017.

[33] R. Boukharfane *et al.*, "A combined ghost-point-forcing / direct-forcing immersed boundary method (IBM) for compressible flow simulations," *Computers & Fluids*, vol. 162, pp. 91–112, Jan. 2018.

[34] S. Brahmachary *et al.*, "A sharp-interface immersed boundary framework for simulations of high-speed inviscid compressible flows: Sharp Interface Immersed Boundary Solver for Compressible Flows," *International Journal for Numerical Methods in Fluids*, vol. 86, no. 12, pp. 770–791, Apr. 2018.

[35] F. Capizzano, "A Compressible Flow Simulation System Based on Cartesian Grids with Anisotropic Refinements," in *45th AIAA Aerospace Sciences Meeting and Exhibit*. Reno, Nevada: American Institute of Aeronautics and Astronautics, Jan. 2007.

[36] C. Chi, B. J. Lee, and H. G. Im, "An improved ghost-cell immersed boundary method for compressible flow simulations," *International Journal for Numerical Methods in Fluids*, vol. 83, no. 2, pp. 132–148, Jan. 2017.

[37] R. Ghias, R. Mittal, and T. Lund, "A Non-Body Conformal Grid Method for Simulation of Compressible Flows with Complex Immersed Boundaries," in *42nd AIAA Aerospace Sciences Meeting and Exhibit*. Reno, Nevada: American Institute of Aeronautics and Astronautics, Jan. 2004.

[38] R. Ghias, R. Mittal, and H. Dong, "A sharp interface immersed boundary method for compressible viscous flows," *Journal of Computational Physics*, vol. 225, no. 1, pp. 528–553, Jul. 2007.

[39] A. Kapahi *et al.*, "Parallel, sharp interface Eulerian approach to high-speed multi-material flows," *Computers & Fluids*, vol. 83, pp. 144–156, Aug. 2013.

[40] A. Kapahi, S. Sambasivan, and H. Udaykumar, "A three-dimensional sharp interface Cartesian grid method for solving high speed multi-material impact, penetration and fragmentation problems," *Journal of Computational Physics*, vol. 241, pp. 308–332, May 2013.

[41] F. Toja-Silva, J. Favier, and A. Pinelli, "Radial basis function (rbf)-based interpolation and spreading for the immersed boundary method," *Computers and Fluids*, vol. 105, pp. 66 – 75, 2014.

[42] J.-J. Xin, F.-L. Shi, and Q. Jin, "Numerical simulation of complex immersed boundary flow by a radial basis function ghost cell method," *Wuli Xuebao/Acta Physica Sinica*, vol. 66, no. 4, 2017.

[43] V. Shankar *et al.*, "Augmenting the immersed boundary method with radial basis functions (rbfs) for the modeling of platelets in hemodynamic flows," *International Journal for Numerical Methods in Fluids*, vol. 79, no. 10, pp. 536 – 557, 2015.

[44] D. Kirshman and F. Liu, "A gridless boundary condition method for the solution of the Euler equations on embedded Cartesian meshes with multigrid," *Journal of Computational Physics*, vol. 201, no. 1, pp. 119–147, Nov. 2004.

[45] H. Uddin, R. Kramer, and C. Pantano, "A Cartesian-based embedded geometry technique with adaptive high-order finite differences for compressible flow around complex geometries," *Journal of Computational Physics*, vol. 262, pp. 379–407, Apr. 2014.

[46] Y. Qu and R. C. Batra, "Constrained moving least-squares immersed boundary method for fluid-structure interaction analysis," *International Journal for Numerical Methods in Fluids*, vol. 85, no. 12, pp. 675–692, Dec. 2017.

[47] D. De Marinis *et al.*, "Improving a conjugate-heat-transfer immersed-boundary method," *International Journal of Numerical Methods for Heat & Fluid Flow*, vol. 26, no. 3/4, pp. 1272–1288, May 2016.

[48] C. Santarelli, T. Kempe, and J. Frohlich, "An immersed boundary method and a ghost cell method for the simulation of heat transfer problems," 2014, pp. 423 – 426.

[49] A. Pacheco-Vega, J. R. Pacheco, and T. Rodic, "A general scheme for the boundary conditions in convective and diffusive heat transfer with immersed boundary methods," *Journal of Heat Transfer*, vol. 129, no. 11, pp. 1506 – 1516, 2007.

[50] J. C. Song, J. Ahn, and J. S. Lee, "An immersed-boundary method for conjugate heat transfer analysis," *Journal of Mechanical Science and Technology*, vol. 31, no. 5, pp. 2287 – 2294, 2017.

[51] T. Bergman *et al.*, *Fundamentals of Heat and Mass Transfer*. Wiley, 2017.

[52] H. K. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics - the finite volume method.* Addison-Wesley-Longman, 1995.

[53] P. L. Roe, "Approximate riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, vol. 135, no. 2, pp. 250 – 250, 1997. [Online]. Available: http://dx.doi.org/10.1006/jcph.1997.5705

[54] A. Harten and J. Hyman, "Self adjusting grid methods for one-dimensional hyperbolic conservation laws," *Journal of Computational Physics*, vol. 50, no. 2, pp. 235 – 69, 1983/05/.

[55] Y. Zhang and C. Zhou, "An immersed boundary method for simulation of inviscid compressible flows," *International Journal for Numerical Methods in Fluids*, vol. 74, no. 11, pp. 775 – 793, 2014.

[56] K. Nakahashi, "Immersed boundary method for compressible euler equations in the building-cube method," Honolulu, HI, United states, 2011, pp. American Institute for Aeronautics and Astronautics (AIAA) –.

[57] S. Brahmachary *et al.*, "A sharp-interface immersed boundary framework for simulations of high-speed inviscid compressible flows," *International Journal for Numerical Methods in Fluids*, vol. 86, no. 12, pp. 770 – 791, 2018.

[58] Y. Qu, R. Shi, and R. C. Batra, "An immersed boundary formulation for simulating high-speed compressible viscous flows with moving solids," *Journal of Computational Physics*, vol. 354, pp. 672 – 691, 2018.