POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

An Immersed Boundary Method approach using hierarchical and overlapping grids for unsteady aerodynamics

MOUSTAFA AWAD
Département de génie mécanique

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie mécanique

Décembre 2021

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

An Immersed Boundary Method approach using hierarchical and overlapping grids for unsteady aerodynamics

présentée par **Moustafa AWAD**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Stéphane ÉTIENNE, président
Jean-Yves TRÉPANIER, membre et directeur de recherche
Guillaume PERNAUDAT, membre et codirecteur de recherche
Bruno SAVARD, membre
Sivakumaran NADARAJAH, membre externe

DEDICATION

To my parents, my wife, my two little kids,
To all my friends at Polytechnique, I will miss you. . .

À mes parents, mon épouse, mes deux petits gamins,
À tous mes amis de Polytechnique, vous me manquerez. . .

ACKNOWLEDGEMENTS

First of all, I would like to thank ALLAH for the accomplished work I have reached and the achievements under the supervision comity. A great appreciated thanks to my research directors Prof. Jean-Yves Trépanier, Prof. Ricardo Camarero, and Dr. Guillaume Pernaudat. Great thanks for your guidance and support through the research. I have the honor to be supervised by such comity like you.

I appreciate and thank all the research group members, especially Renan SOUSA and Sujaat Ali, for the real meaning of brotherhood, friendship, and support.

I would like to thank the mechanical engineering department of Polytechnique Montreal for its support, assistance, and the afforded computational facilities.

A special thanks to my adorable wife, Noheir El Faham, and my two buddies Omar & Yassine. I express my gratitude to my family, my mother, my dad, and my sisters.

My deep thanks to all my friends, and anyone helped me to direct this work.

RÉSUMÉ

La simulation numérique d'écoulements compressibles autours d'objets mobiles comprend de nombreuses applications, comme la séparation de magasins à grande vitesse, le suivi de débris, les jets à un nombre de Mach élevé, et l'écoulement à l'intérieur des disjoncteurs haute-tension. La modélisation numérique de l'ensemble de ces applications se heurte principalement à certains défis : les complexités géométriques, l'écoulement compressible à haute vitesse, et de la prise en compte du mouvement des objets. A ce sujet, cette recherche vise à relever ces défis en développant un algorithme pour simuler les écoulements compressibles sur des géométries complexes en mouvement relatif.

Ce travail présente le développement d'un code numérique basé sur l'intégration de la méthode des frontières immergées (Immersed Boundary Method, IBM) avec des maillages superposés (Overset Grid) hiérarchiques pour simuler des problèmes d'écoulement compressible non visqueux. L'approche proposée réduit la complexité de la simulation numérique par rapport aux approches conventionnelles avec des maillages ajustés aux frontières.

La méthode des frontières immergées choisie est la méthodologie (Sharp Interface, SI), où la géométrie est représentée avec précision en fonction de la résolution de la grille Eulérienne. Cette méthode est adaptée pour représenter des géométries rigides. La résolution de la grille est contrôlée par un raffinement local basé sur l'approche hiérarchique linéaire. Les quadtrees linéaires sont une structure de données efficace pour représenter les grilles hiérarchiques pour une représentation précise de géométries complexes avec un faible coût de traitement. Des contrôles de grille supplémentaires sont présentés, en tant que préadaptation du maillage, et pour adapter localement le domaine de calcul aux variations physiques auparavant étudiées.

Une approche de marquage robuste est présentée, basée sur le parcours de la géométrie, comme l'une des contributions de cette thèse. Cette approche est basée sur un bouclage sur la courbe géométrique plutôt que sur l'ensemble de la grille Eulérienne pour identifier la frontière immergée. Cette approche prend en charge l'identification de géométries complexes sur des maillages cartésiens et hiérarchiques. Les informations de la géométrie immergée générées par l'approche de marquage sont transférées vers d'autres parties de l'algorithme pour intégrer le module des maillages superposés et le solveur fluide.

La prise en compte du mouvement des objets est réalisée par l'utilisation de maillages superposés où la topologie immergée et la grille se déplacent dans un repère solidaire. Cette approche permet d'effectuer un marquage unique et constant du maillage même avec le déplacement de l'objet. En effet, on élimine le besoin de refaire le marquage de la grille à chaque pas de temps. Les applications avec des objets mobiles sont jusqu'à présent limitées aux mouvements de translation sans collisions entre les parois.

Une approche de reconstruction implicite par la méthode des moindres carrés a été mise en œuvre permettant de reconstruire la solution d'écoulement. Ce schéma prend en compte précisément la reconstruction des conditions aux limites pour les équations d'Euler au niveau des cellules d'interface avec la frontière immergée, ainsi que la reconstruction entre les maillages superposés pour transférer la solution d'écoulement entre celles via une couche de cellules interpolées autour de la frontière de la superposition. Le schéma de reconstruction a été vérifié analytiquement et numériquement pour les deux cas (pour l'application des conditions aux limites et la reconstruction pour le transfert de la solution entre les maillages superposés).

Un résoluteur d'écoulement à schéma de volumes finis a été implémenté sur la base d'un schéma de Roe, où les calculs des flux sont généralisés à la discrétisation de l'espace Cartésien et hiérarchique. La méthode mise en œuvre est du premier ordre dans l'espace et dans le temps, avec une intégration temporelle explicite. De plus, ce résoluteur est mis à jour avec la formulation Arbitrary Lagrangian-Eulerian (ALE) pour prendre en compte les vitesses de grille introduites par le module de maillages superposés.

Un ensemble de cas de test de vérification sont menés et classés par type de configuration spatial (maillage unique ou maillages superposés), type d'écoulement (régime permanent ou instationnaire) et le mouvement de la géométrie (stationnaire, en mouvement à vitesse uniforme ou en mouvement accéléré) . Les cas tests réalisés vérifient l'intégration de chaque module du code développé et évaluent l'ordre de précision des schémas implémentés. Les résultats de l'algorithme développé sont limités à l'ordre de convergence du résoluteur (premier ordre) dans le cas d'écoulements sans discontinuités, et un ordre de convergence autour de (0.5) est atteint pour les cas d'écoulements discontinus ce qui concorde avec des configurations similaires disponibles dans la littérature.

Cette recherche constitue, comme l'une des principales contributions de ce travail, la première version d'un algorithme entièrement basé sur l'approche IBM intégrée avec les maillages superposés hiérarchiques. De plus, ce jalon est extensible pour prendre en charge les mouvements complexes qui incluent la collision des géométries, les implémentations de schémas d'ordre élevé et les extensions 3D. La réalisation de ces extensions est en cours par l'équipe de recherche permettant de repousser les limitations de la méthodologie et du code développés et d'ouvrir de nouvelles perspectives pour des contributions scientifiques supplémentaires.

ABSTRACT

Numerical simulation of compressible flows over moving bodies can be found in numerous applications, ranging from internal and external flows (store separation at high speed, debris tracking, flying bodies at high Mach number, flow inside circuit breakers, *etc*). Numerical modeling of this wide range of applications raises some challenges when these three elements are combined; a) Geometric complexity. b) Flow complexity. and c) Body motion. This research addresses these challenges by the development of an algorithm able to simulate compressible flows over complex geometries in relative motion.

This work presents the development of a numerical code based on the integration of the Immersed Boundary Method (IBM) with hierarchical Overset grids to simulate inviscid compressible flow problems. The proposed approach reduces the complexity of the numerical simulation compared to conventional body-fitted approaches.

The immersed boundary is represented by the Sharp Interface (SI) method, where the geometry is accurately represented as a function of the Eulerian grid resolution. This method is suitable to represent rigid geometries. The grid resolution is controlled by a local grid refinement based on linear hierarchical grids. Linear quad-trees is an efficient data structure to represent hierarchical grids that can describe complex geometries accurately with low computational cost. Additional grid controls are presented, as grid pre-adaptation, to locally pre-adapt the computational domain for complex flow cases.

A robust tagging approach is presented, based on a "Geometry Marching" procedure, as one of the contributions of this thesis. This approach is based on looping over the domain boundaries rather than over the entire Eulerian grid to identify the immersed boundary. The proposed tagging approach supports the identification of complex geometries on Cartesian and hierarchical grids. The information of the immersed geometry thus generated is transferred to other parts of the algorithm to integrate the overset module and flow solver.

The body motion is introduced by the integration of overset grids, where every moving geometry is immersed on an overset tagged grid. The implementation of this approach allows to perform the mesh tagging only once on the overset grid, and this tagging remains constant with grid motion. This integration eliminates the re-tagging of the geometry every time step, where the overset grid will move entirely with fixed tagged geometry. The applied cases for overset grids are limited to translation motions, with a non-colliding geometries.

A second-order implicit least-square representation is implemented to reconstruct the flow solution. This is applied to the reconstruction of boundary conditions for Euler's equations at the interface cells of the immersed boundary, as well as the transfer of the flow solution between overlapping grids via a layer of interpolated cells around the overset boundary. The reconstruction scheme is verified analytically and numerically for both cases (boundary condition and overset reconstruction).

A cell-centered finite-volume scheme flow solver is implemented based on a flux difference splitting scheme (Roe's Scheme), where flux calculations are generalized to Cartesian and hierarchical space discretization. The implemented method is first-order in space and time, with an explicit temporal integration. In addition, the flow solver is updated with the Arbitrary Lagrangian-Eulerian (ALE) formulation to take into account the grid motion introduced by the overset module.

A set of verification test cases (fifteen test cases) are conducted and categorized by the type of space configuration (single grids/overset grids), flow type (steady/unsteady), and body motion (stationary/uniform speed motion/accelerated motion). These test cases verify the integration of each module of the overall developed code and assess the order of accuracy of the implemented schemes. The results are limited to the solver order of convergence (first-order) in case of flows without discontinuities, and an order of convergence around 0.5 is attained for the cases of discontinuous flows, these results agree solutions for similar configurations available in the literature.

As one of the main contributions of this work, this research succeeded in developing the first version of an entirely IBM-based algorithm integrated with hierarchical overset grids. In addition, this milestone is extendable to support complex body motion that includes body collisions, high-order scheme implementations, and 3D extensions. These extensions are currently in progress by the research team to push forward the limitations of the developed code and open new sights for additional scientific contributions.

TABLE OF CONTENTS

DEDICATION . . . iii

ACKNOWLEDGEMENTS . . . iv

RÉSUMÉ . . . v

ABSTRACT . . . vii

TABLE OF CONTENTS . . . ix

LIST OF TABLES . . . xiii

LIST OF FIGURES . . . xiv

LIST OF SYMBOLS AND ACRONYMS . . . xix

LIST OF APPENDICES . . . xxi

CHAPTER 1 INTRODUCTION . . . 1
- 1.1 Context . . . 1
- 1.2 Numerical treatment of unsteady compressible flows with bodies in relative motion . . . 2
  - 1.2.1 Dynamic meshes . . . 2
  - 1.2.2 Overset grids . . . 5
  - 1.2.3 Cartesian grid approach (Cut-cell approach) . . . 6
  - 1.2.4 Numerical treatments conclusion . . . 6
- 1.3 The Immersed Boundary Method . . . 7
- 1.4 Objectives of the present work . . . 8
- 1.5 Research originality . . . 9
- 1.6 Thesis structure . . . 9

CHAPTER 2 LITERATURE REVIEW . . . 11
- 2.1 Background of the Immersed Boundary Method . . . 11
  - 2.1.1 Diffused Interface Method (DIM) . . . 11
  - 2.1.2 Sharp Interface Method (SIM) . . . 12
- 2.2 Eulerian mesh . . . 13

2.2.1 Adaptive Mesh Refinement (AMR) 14
2.2.2 Non-uniform Cartesian grid refinement 16
2.2.3 Unstructured grid refinement 16
2.3 Tagging 17
2.3.1 The Ray Tracing approach 18
2.3.2 The Level Set approach 19
2.3.3 The explicit minimum distance approach 20
2.4 Reconstruction schemes 21
2.5 IBM with moving boundaries 23
2.5.1 Field Extension approach 23
2.5.2 Direct reconstruction approach 24
2.5.3 Overset grids 24
2.6 Critical literature 25

CHAPTER 3 HIERARCHICAL GRID GENERATION - OVERSET GRID AND TAGGING 28
3.1 Global methodology 28
3.1.1 Geometry/Discrete topology 29
3.1.2 Hierarchical grid 29
3.1.3 Mesh tagging 29
3.1.4 Overset grid 30
3.1.5 Refinement criteria and grid controls 30
3.1.6 Solution reconstruction 31
3.1.7 Flow solver 31
3.2 Geometry/Discrete topology 31
3.3 Hierarchical grid 34
3.3.1 Numbering scheme 36
3.3.2 Grid balancing 38
3.3.3 Neighbors localization 41
3.3.4 Side list generation of hierarchical grid 45
3.4 Mesh tagging 46
3.4.1 Basic terminologies and definitions 47
3.4.2 Properties of intersected cells 49
3.4.3 Geometry marching algorithm 50
3.4.4 Types of intersected cells 55
3.4.5 Types of Interface Cells (IC) 56

3.4.6 Face discretization 58
3.5 Overset grid 59
3.6 Refinement criteria 63
3.7 Grid management and control Add-ons 70

CHAPTER 4 FLOW SOLVER 73
4.1 Governing equations 73
4.2 Numerical scheme 74
4.3 Reconstruction scheme 78
4.3.1 Reconstruction methodology 79
4.3.2 Reconstruction verification 85
4.4 Boundary conditions for Euler equations 86
4.5 Treatment of overset grid motion 91

CHAPTER 5 VERIFICATION AND VALIDATION OF THE ALGORITHM . . . 93
5.1 Algorithm implementation 93
5.2 General aspects 93
5.3 Test cases classification 94
5.4 Steady flows on single grids 95
5.4.1 Subsonic flow in tube (CAT11) 96
5.4.2 Supersonic flow in tube (CAT12) 97
5.4.3 Subsonic Rayleigh flow (CAT13) 99
5.4.4 Supersonic Rayleigh flow (CAT14) 102
5.4.5 Subsonic radial flow (CAT15) 105
5.4.6 Confluence of two supersonic flows (CAT16) 107
5.4.7 Supersonic flow over a bump (CAT17) 113
5.5 Unsteady flows on a single grid 115
5.5.1 Shock tube flow (CAT21) 116
5.5.2 Tube in impulsive motion (CAT22) 119
5.5.3 Unsteady supersonic flow over a Wedge (CAT23) 123
5.6 Steady flows on overset grids 127
5.6.1 Supersonic tube flow with stationary overset grid (CAT31) 127
5.6.2 Supersonic tube flow with moving overset grid (CAT32) 132
5.7 Unsteady flows on overset grids 135
5.7.1 Unsteady supersonic flow over a wedge with stationary overset grid (CAT41) 135
5.7.2 Unsteady supersonic flow over a wedge with moving overset grid (CAT42) 141

5.7.3 Unsteady supersonic flow over three moving accelerated wedges on overset grids (CAT43) 144

CHAPTER 6 CONCLUSION 150

6.1 Summary of work 150

6.2 Limitations 151

6.3 Future research 152

REFERENCES 154

APPENDICES 167

LIST OF TABLES

Table 3.1 Siblings pair of a parent cell and the corresponding indices in four cardinal directions 43
Table 3.2 Example of a vertical side list 46
Table 3.3 List of cell index vs.`cellID` 46
Table 4.1 Reconstruction types of boundary conditions 91
Table 5.1 Classification of test cases 94
Table 5.2 Test cases - Category#1: Steady flows on a single grid 94
Table 5.3 Test cases - Category#2: Unsteady flows on a single grid 95
Table 5.4 Test cases - Category#3: Steady flows on overset grids 95
Table 5.5 Test cases - Category#4: Unsteady flows on overset grids 95
Table 5.6 Operating conditions of tube in impulsive motion 120
Table 5.7 Operating conditions of supersonic wedge on single grid 125
Table 5.8 Operating conditions of moving supersonic wedge on overset grid . . 142
Table A.1 Hash-table information of a given key (`cellID`) in hierarchical grid. . 167
Table A.2 Hash-table tagging information of a given key `faceID`. 168

LIST OF FIGURES

Figure 1.1 The Gripen E aircraft performing the separation test of the drop fuel tank [1]. 1
Figure 1.2 Interaction between the different approaches used for testing and assessment of store separation [2]. 2
Figure 1.3 Tetrahedral computational domain and simulation contours of rocket booster separation using dynamic meshes [3]. 3
Figure 1.4 Adaptive Cartesian grid with static pressure contours for the simulation of a Space Launch System [4]. 3
Figure 1.5 Cross-section view of a simplified circuit breaker, indicating different stationary and moving parts [5]. 4
Figure 1.6 Cross-section view of the relief valve, with contours of Mach number and pressure [6]. 4
Figure 1.7 Overset grids for debris tracking simulation [7]. 5
Figure 1.8 Overset grids for flow simulation over the 30P30N airfoil [8]. 6
Figure 1.9 Classification of cells resulting from tagging procedure. 8
Figure 2.1 IBM Classification. 11
Figure 2.2 Boundary condition reconstruction with Sharp Interface Method (SIM) [9]. 13
Figure 2.3 Illustration of quadtree levels representation [10]. 15
Figure 2.4 Local anisotropic grid refinement [11]. 16
Figure 2.5 Local non-uniform grid refinement [12]. 17
Figure 2.6 Unstructured grid refinement for IBM with the contours of density distribution over a wedge [13]. 18
Figure 2.7 Illustration of ray tracing the concept [14]. 19
Figure 2.8 Illustration of Level Set function [15]. 20
Figure 2.9 Illustration of explicit tagging. 21
Figure 2.10 Reconstruction stencil of an interface cell based on three fluid cells and two boundary condition points. 22
Figure 2.11 Field Extension Approach for FCC Treatment. 24
Figure 2.12 Direct Interpolation Approach for FCC Treatment. 25
Figure 2.13 Overset grids around a mackerel, the geometry is descretized by a body-fitted tetrahedral elements [16]. 26
Figure 3.1 Representation of the immersed geometry. 32
Figure 3.2 Entities of discrete topology. 33

Figure 3.3 Quadtree representation with two levels of refinement. Refined siblings' location relative to the parent cell is indicated (SW, SE, NW, NE). . 35
Figure 3.4 Siblings indices generated from a parent cell refinement. 37
Figure 3.5 Numbering scheme adopted to represent a linear quadtree data structure. 38
Figure 3.6 Hierarchical grid 2:1 balancing rule. 39
Figure 3.7 Example of procedures of grid balance, using the level difference between neighbors. 40
Figure 3.8 Neighboring possibilities of balanced grid. 42
Figure 3.9 Numbering of sides of a hierarchical grid. 46
Figure 3.10 Cells classification of a tagged immersed topology. 48
Figure 3.11 Properties of an intersected cell. 49
Figure 3.12 Geometry marching procedure to find intersected cells. 51
Figure 3.13 Geometry marching procedure on a Cartesian grid. 53
Figure 3.14 Geometry marching in hierarchical grid. 54
Figure 3.15 Illustration of two cases of intersected cells. 55
Figure 3.16 Four possibilities of intersected cells. 56
Figure 3.17 one corner cut vs. two corners cut Intersected Interface Cell (IIC). . . 57
Figure 3.18 Determination of Non-Intersected Interface Cell (NIIC) by two-corners cut Interface Outside Cell (IOC) of cell W. 57
Figure 3.19 Interface tagging verification by neighborhood check. 58
Figure 3.20 The four steps of the tagging algorithm. 59
Figure 3.21 Overset supported configuration. 60
Figure 3.22 Overset treatment for overlapped grids. 62
Figure 3.23 Representation of the interpolated cells conditioning for 1D overset configuration. 63
Figure 3.24 Application of the first criteria of refinement to limit to 4 the number of geometric points inside a cell. 66
Figure 3.25 Application of refinement criteria #2. 67
Figure 3.26 Application of the proximity refinement criteria. 69
Figure 3.27 Five levels of refinement with vs. without uniform refinement around a NACA 0012 profile. 70
Figure 3.28 Grid pre-adaptation with three levels of refinement following the adaptation topology. 71
Figure 3.29 Representation of doubling the grid resolution will not lead to a grid topology finer by a factor 1/2 for all grid elements. 72

Figure 3.30 The application of $RefineUniformGrid$ function to generate the same grid topology with $1/2^N$ factors of reduction. 72
Figure 4.1 Cell-centered control volume definition in Cartesian grids. 74
Figure 4.2 Evaluation of average properties at the control volume boundaries. . . 75
Figure 4.3 Flux representation in hierarchical grids in x-direction. 77
Figure 4.4 Reconstruction stencil of interface cell P. 79
Figure 4.5 Reconstruction of Dirichlet and Neumann Boundary Condition (BC)'s of cell P from East side. 80
Figure 4.6 Dirichlet and Neumann BC reconstruction for cell P. 81
Figure 4.7 Illustration to implicitly reconstruct the gray cells (7,8 and 9). 84
Figure 4.8 Verification of reconstruction scheme using analytical function. 86
Figure 4.9 Possible configurations for BC reconstruction. 87
Figure 4.10 Slip boundary condition application by vector projection. 89
Figure 4.11 Reconstruction stencil on a 2-D overset grid. 90
Figure 5.1 Subsonic tube test setup and computational domain. 96
Figure 5.2 Comparison of numerical solution of Mach number for the subsonic flow in a tube. 97
Figure 5.3 Comparison of numerical solution of pressure for the subsonic flow in a tube. 97
Figure 5.4 Supersonic tube test setup. 98
Figure 5.5 Comparison of numerical solution of Mach number for the supersonic flow in a tube. 98
Figure 5.6 Comparison of numerical solution of Pressure for the supersonic flow in a tube. 99
Figure 5.7 Subsonic Rayleigh flow test setup and operating boundary conditions. 100
Figure 5.8 Mach number variation for subsonic Rayleigh flow on a 10x10 grid. . 101
Figure 5.9 Mach number variation for subsonic Rayleigh flow on a 80x80 grid. . 101
Figure 5.10 Mach number L_2 norm convergence for subsonic Rayleigh flow. 102
Figure 5.11 Test case setup for supersonic Rayleigh flow. 103
Figure 5.12 Supersonic Rayleigh flow solution on a 10x10 grid. 103
Figure 5.13 Supersonic Rayleigh flow solution on a 160x160 grid. 104
Figure 5.14 L_2 norm convergence for supersonic Rayleigh flow. 104
Figure 5.15 Test case setup for subsonic radial flow. 105
Figure 5.16 Pressure distribution of IBM vs. unstructured solvers. 106
Figure 5.17 Pressure distribution of IBM vs. analytical solution on a Cartesian grid of 85×85 (average element length of 0.0247 m). 107

Figure 5.18 Density and pressure L_2 norm convergence for IBM vs. unstructured solvers. 107
Figure 5.19 Test case configuration, flow angles, and boundary conditions for the confluence of two supersonic flows. 108
Figure 5.20 Computational domains of pre-adapted refined grids for confluence test case. 109
Figure 5.21 Pressure distribution along the outlet section for different grid resolutions. 110
Figure 5.22 Pressure error along the outlet section for different grid resolutions. . 111
Figure 5.23 Pressure L_2 norm convergence for supersonic confluence of two flows. 112
Figure 5.24 Numerical results on a Level 3 grid. 112
Figure 5.25 Test case setup for a supersonic bump configuration. 113
Figure 5.26 Distribution of physical properties over a supersonic bump $M = 1.4$. . 114
Figure 5.27 Mach number distribution on upper and lower walls. 115
Figure 5.28 Shock tube configuration. 116
Figure 5.29 Pressure distribution at mid-section of a shock tube at different times, grid size 81×81. 117
Figure 5.30 u-velocity distribution at mid-section of a shock tube at different time frame, grid size 81×81. 117
Figure 5.31 Density distribution at mid-section of a shock tube at different time frame, grid size 81×81. 117
Figure 5.32 Order of convergence of L_2 norm of physical quantities of shock tube. 118
Figure 5.33 Schematic representation of a tube in impulsive motion with operating zones. 119
Figure 5.34 Flow properties at mid-section of a tube in impulsive motion for a grid resolution of 200×20 at $t = 0.0089s$. 121
Figure 5.35 Flow properties L_2 norm convergence of a tube in impulsive motion. . 122
Figure 5.36 2D flow properties distribution in a tube in impulsive motion, grid 400×40 at $t = 0.0089s$. 123
Figure 5.37 Computational domain of supersonic wedge test case. 124
Figure 5.38 Immersed wedge topology in Cartesian grid for different grid resolutions. 124
Figure 5.39 Density distribution of the shock development over a wedge at Mach 2, grid resolution of 90×80. 126
Figure 5.40 Flowfield properties distribution of the shock development over a wedge at Mach 2, grid resolution is 360×320 at steady state conditions. . . 127
Figure 5.41 Computational domain of a stationary overset supersonic tube. 128
Figure 5.42 Overset computational domain for supersonic flow tube. 129

Figure 5.43 Solution error of flow properties on the reference grid. 130
Figure 5.44 Solution error of flow properties on the overlapping grid. 131
Figure 5.45 Solution of flow properties on the overset grids. 132
Figure 5.46 Computational domain of moving overset supersonic tube. 133
Figure 5.47 Pressure error on reference grid at initial position, mid-trajectory, and at the end of trajectory. The position of the hole changes with the mobile grid motion. 133
Figure 5.48 Pressure error on overset grid at initial position, mid-trajectory, and at the end of trajectory. 134
Figure 5.49 Pressure distributions on reference grid at initial position, mid-trajectory, and at the end of trajectory. The position of the hole changes with the mobile grid motion. 134
Figure 5.50 Pressure distributions on overset grid at initial position, mid-trajectory, and at the end of trajectory. 135
Figure 5.51 Computational domain dimensions for a symmetric wedge on an overset grid. 136
Figure 5.52 Computational domain with perfectly aligned overset grids. 137
Figure 5.53 Grid offset of interpolated cell centers. 138
Figure 5.54 Density distribution on single and reference grids, $n_t = 100$. 139
Figure 5.55 Density distribution on single and overset Cartesian grids, $n_t = 100$. . 139
Figure 5.56 Pressure distribution on single and reference girds, $n_t = 100$. 140
Figure 5.57 Pressure distribution on single and overset Cartesian girds, $n_t = 100$. 140
Figure 5.58 Overset reconstruction error of flow properties. Outer loop: overset grid. Inner loop: reference grid. 141
Figure 5.59 Computational domain of moving overset supersonic tube. 142
Figure 5.60 Computational domain discretization at initial position $t = 0$. 143
Figure 5.61 Unsteady density distribution of supersonic moving wedge at Mach 2, grid resolution $1260 \times 160/\ 120 \times 80$ at different time steps. 144
Figure 5.62 Overset grids trajectories for three wedges, aligned initially at $x = 0$. 145
Figure 5.63 Trajectory and corresponding velocity profiles for the three moving wedges 146
Figure 5.64 Computational overset domain at $t = 0$. 147
Figure 5.65 Unsteady density distribution of an accelerated three wedges. 148
Figure 5.65 Unsteady density distribution of an accelerated three wedges. 149
Figure A.1 Hierarchical grid element definition. 168
Figure A.2 UML diagram of the developed code 169

LIST OF SYMBOLS AND ACRONYMS

ALE	Arbitrary Lagrangian Eulerian
AMR	Adaptive Mesh Refinement
API	Application Programming Interface
BC	Boundary Condition
BI	Body Intercept
BLI	Bi-Linear Interpolation
CFD	Computational Fluid Dynamics
DDF	Discrete Delta Function
DIM	Diffused Interface Method
ENO	Essentially Non-Oscillatory
FC	Face Cell
FCC	Fresh Cleared Cells
FIC	Fluid Interface Cell
FSI	Fluid Structure Interaction
GC	Ghost Cell
GCL	Geometric Conservation Laws
HPC	High Performance Computing
IBM	Immersed Boundary Method
IBVP	Initial Boundary Value Problem
IC	Interface Cell
IDW	Inverse Distance Weighting
IIC	Intersected Interface Cell
IOC	Interface Outside Cell
LSI	Least Square Interpolation
MUSCL	Monotone Upstream-Centered Scheme for Conservation Laws
NIIC	Non-Intersected Interface Cell
OC	Outer Cell
PDE	Partial Differential Equation
RBF	Radial Basis Function
SIM	Sharp Interface Method
SLS	Space Launch System
TLI	Tri-Linear Interpolation
TVD	Total Variance Diminishing

UML	Unified Modeling Language
WENO	Weighted Essentially Non-Oscillatory

LIST OF APPENDICES

Appendix A Algorithm Implementation 167

CHAPTER 1 INTRODUCTION

1.1 Context

Numerical simulation of unsteady compressible flow fields past moving geometries with adequate cost and accuracy is a highly demanding problem. The difficulties confronting numerical approaches are often linked to flow complexities (shockwaves, wakes, separated flow, dynamic stall, *etc.*) and/or geometric complexities (thin bodies, body proximity in relatively small tolerances, complex shapes, moving bodies, *etc.*).

Numerous applications illustrate these complexities and the importance of numerical simulations to solve these kinds of problems. The store separation from aircraft is an example. This could be any payload released or separated from the aircraft during its flight (e.g., a drop tank, an unmanned areal vehicle, flares, *etc.*), as illustrated in Fig. 1.1. The difficulty of the separation procedure resides on the complexity of the flowfield surrounding the store, especially as it is exposed to a high flow speed, near the supersonic regime with shockwaves and high dynamic pressures. In addition, the non-uniformity of the flow around the store and its interaction with shockwaves may lead to an abrupt change in the store's attitude after separation, which in turn may cause post-separation hazards.

The integration of Computational Fluid Dynamics (CFD) with flight tests and experimental wind tunnel tests has proved a cost effective and time management complement. The interaction between these three approaches, Fig. 1.2, illustrates the importance of the CFD approach to investigate unsteady aerodynamic compressible flow problems with bodies in relative motion.

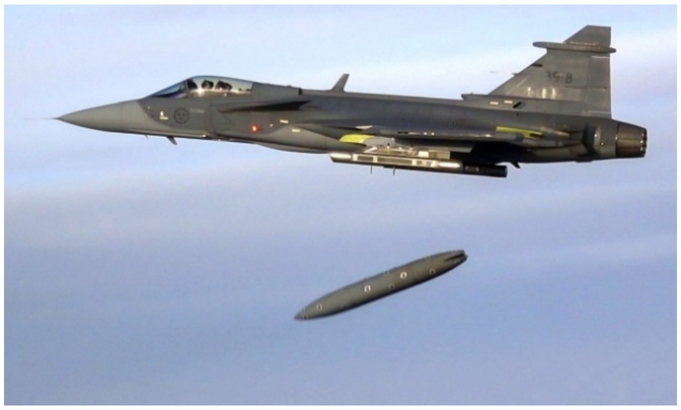


Figure 1.1 The Gripen E aircraft performing the separation test of the drop fuel tank [1].

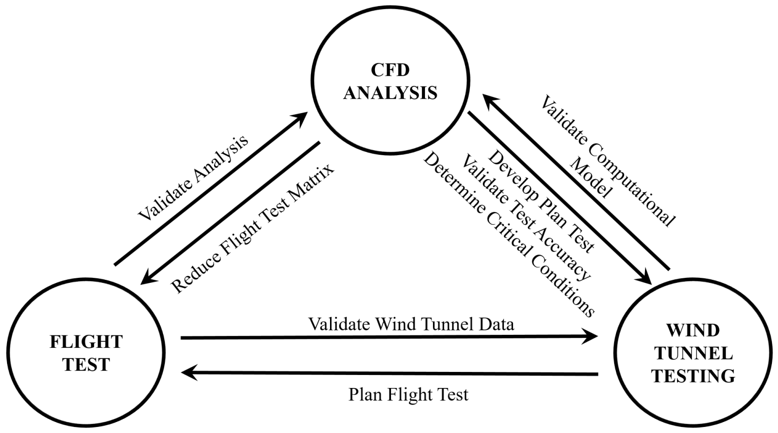


Figure 1.2 Interaction between the different approaches used for testing and assessment of store separation [2].

With CFD as a complementary pivot approach, the present research introduces a numerical tool to simulate 2D unsteady compressible inviscid flows over moving bodies. This tool is based on the integration of the Immersed Boundary Method (IBM) and overset grids, as a step toward the development of an optimized and functional numerical tool in terms of precision and complexity to perform the simulation.

1.2 Numerical treatment of unsteady compressible flows with bodies in relative motion

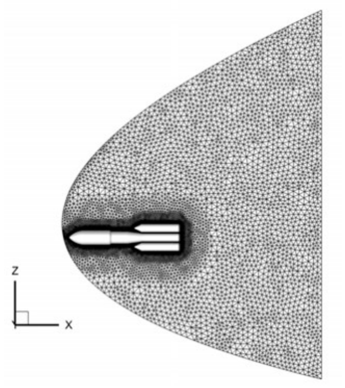
This section reviews different numerical techniques used to solve unsteady compressible flows with boundaries in relative motion. First, the body-fitted numerical approaches will be discussed, namely, the dynamic meshes, the overset grids, and the Cartesian grid method.

1.2.1 Dynamic meshes

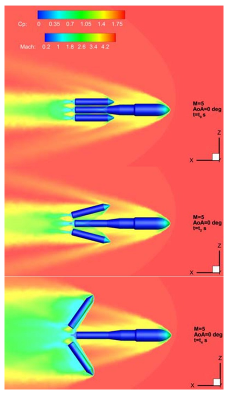
In this approach, the computational domain is discretized into a single or multiple zones, where the grid connectivity is updated with the boundary motion. The movement of the boundary inside the grid is treated by adapting the mesh every time step. This grid update is carried out by different techniques:

- Complete global re-meshing of the computational domain.
- Local refining/coarsening of the mesh cells within applying certain quality criteria.

Both techniques have demonstrated good results for 2D and 3D configurations [3, 17, 18], as illustrated in Fig. 1.3.



(a) (a) Hypersonic domain using triangular mesh cells.



(b) Contours of Mach number and pressure coefficient.

Figure 1.3 Tetrahedral computational domain and simulation contours of rocket booster separation using dynamic meshes [3].

The adaptive meshing is also implemented with a Cartesian grid discretization, as shown in Fig. 1.4. This approach has been applied to simulate the separation of solid rocket boosters from an Space Launch System (SLS) problem [4].

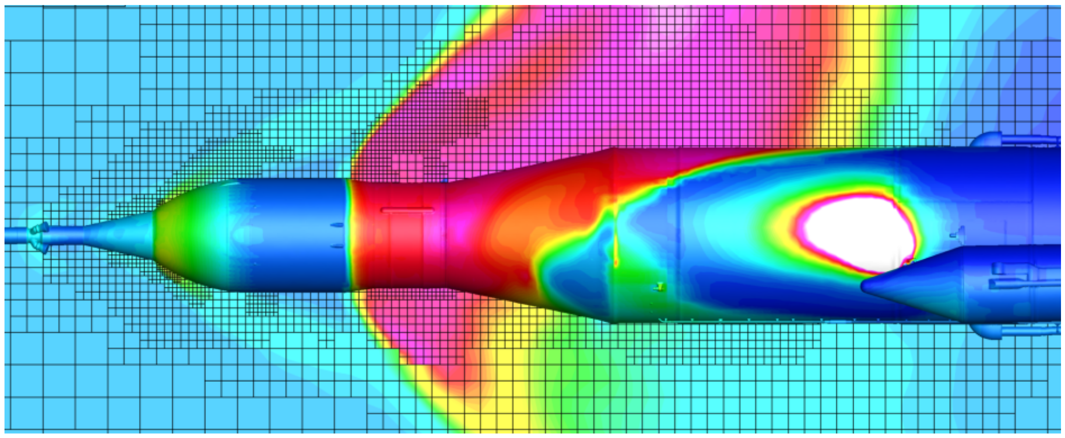


Figure 1.4 Adaptive Cartesian grid with static pressure contours for the simulation of a Space Launch System [4].

Another industrial applications dealing with complex internal flows, is the flow inside high voltage circuit breakers used in electrical power stations. These involve multi-physics interaction of high speed compressible flow of plasma, and combined various energy exchange phenomena such as thermal radiation. The cross-section view of a simplified circuit breaker is illustrated in Fig. 1.5, where the moving and non-moving parts are indicated. A compressible flow simulation is conducted [5] to simulate the airflow through the chamber of the circuit breaker.

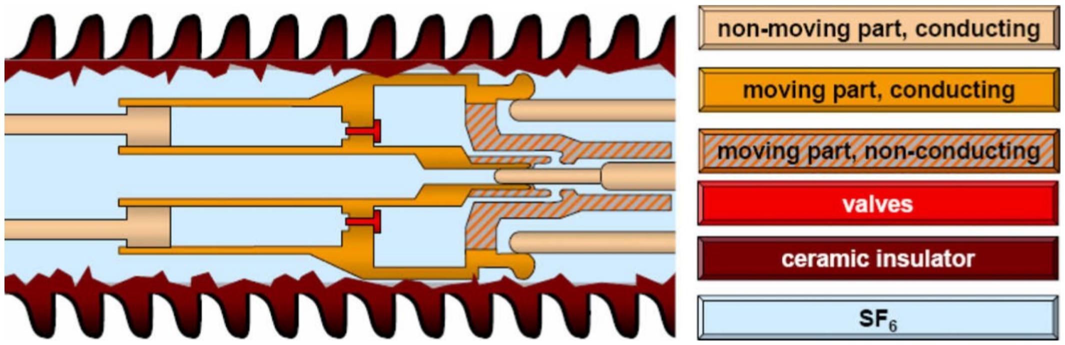


Figure 1.5 Cross-section view of a simplified circuit breaker, indicating different stationary and moving parts [5].

Additional examples of dynamic mesh flow simulations with moving bodies are the compressible flow simulation of pneumatic relief valves [6, 19], shown in Fig. 1.6 respectively.

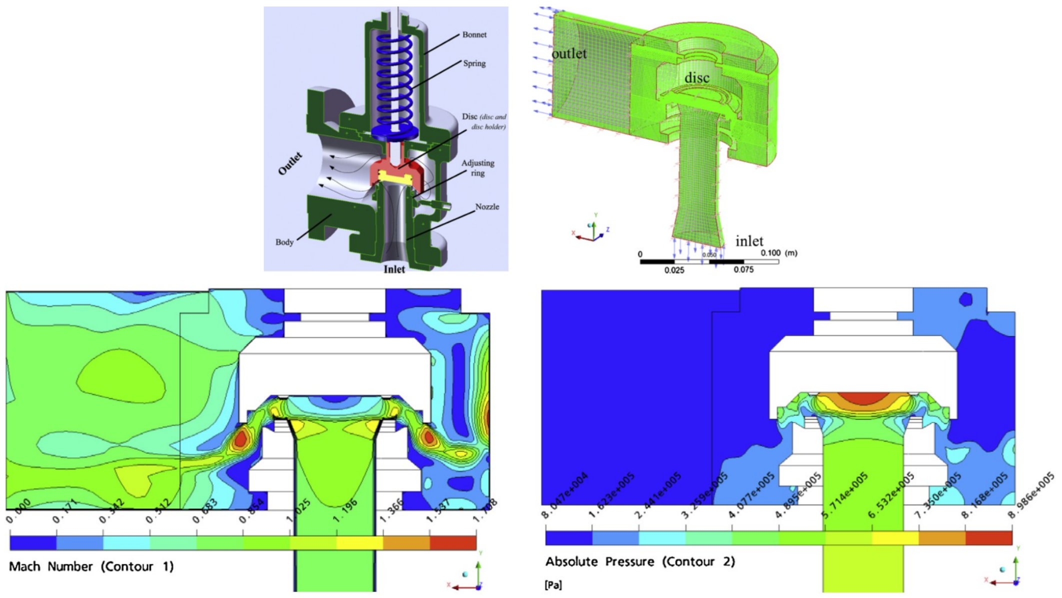
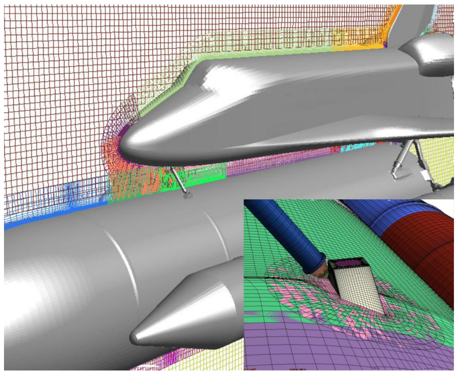


Figure 1.6 Cross-section view of the relief valve, with contours of Mach number and pressure [6].

1.2.2 Overset grids

The approach of overset grids or Chimera grids [20] generally relies on overlapping grids to decompose the domain of interest, where the interpolation accomplishes the communication between overlapping grids within the overlapped layer. This approach allows the motion of the grids relative to each other while the topology of each grid remains constant, avoiding the process of re-meshing at each time step. The communication between the overlapping grids and the treatment of the data transfer between grids is known as "grid assembly," whereby a procedure of hole creation is applied to remove the cells containing the solid boundary from the background grid. This approach was first proposed by [20] and applied to store separation simulations at compressible flow conditions [21–27].

The implementation of structured overset grids makes grid generation easier, in addition to its definitive advantage for moving boundaries configurations. Hence, it is adopted to simulate a Space Launch System (SLS) during payload separation [7, 28, 29]. Structured overset grids were also implemented to investigate a space shuttle accident [7] caused by foam debris; whereby the debris trajectories that hit the leading edge of the space shuttle wing were simulated, as shown in Fig. 1.7.


(a) Overset grids for SLS simulation

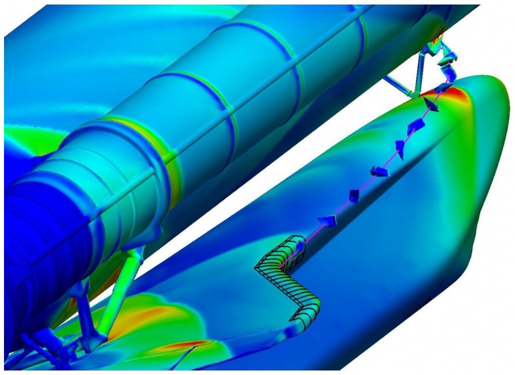

(b) Debris trajectory

Figure 1.7 Overset grids for debris tracking simulation [7].

Other applications of the overset grids to simulate aerodynamic problems with moving boundaries include flows over flaps and leading-edge slats [8, 26], Fig. 1.8.

The extension of the unstructured overset approach makes the grid generation process more flexible, but complicates the solver and increases the computational effort [27].

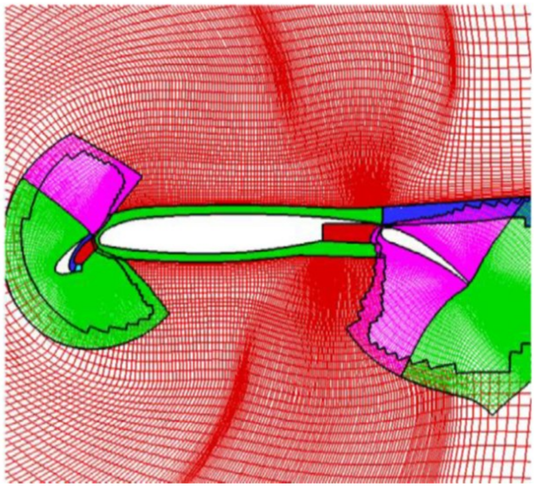


Figure 1.8 Overset grids for flow simulation over the 30P30N airfoil [8].

1.2.3 Cartesian grid approach (Cut-cell approach)

The Cartesian grid approach was first introduced by [30] and consists of submerging the geometry in a Cartesian grid. Local grid refinement is then performed after the identification of the immersed boundary inside the grid, and applying a cut-cell procedure to adapt the cells to the boundary. Generally speaking, it can be considered as a non-traditional body-fitted approach, that accelerates the meshing procedure, and leads to more efficient computations. This approach is characterized by its cost effectiveness, resulting from the effort related to grid generation [31–37].

1.2.4 Numerical treatments conclusion

From the review of the previous numerical approaches, the advantages can be summarized as:

- The Cartesian grid approach is helpful in order to facilitate grid generation, where grids over complex geometries can be generated more easily than traditional body-fitted approaches.
- Overset grids eliminate the re-gridding at every time step during the simulation, which reduces the computational cost compared to dynamic meshes.
- Relying on structured grids will make the computational cost more efficient in terms of data structure handling.

The analysis of these advantages leads to propose a non-body fitted approach that supports the solution for two-dimensional compressible inviscid flows over stationary and moving bodies. It is based on the integration of the Immersed Boundary Method with overset grids.

1.3 The Immersed Boundary Method

The Immersed Boundary Method (IBM) is a methodology introduced by Peskin [38] to overcome the burden of generating body-fitted grids. Primarily introduced to simulate the blood flow in heart artery, the methodology was extended to cover multiple Fluid Structure Interaction (FSI) applications. In this approach, the boundary is represented by a set of Lagrangian points immersed in an Eulerian grid, 1.9. The Eulerian grid lines do not necessarily conform to the body contour, unlike the body-fitted approaches.

This non-body-fitted methodology has high potential because of its capability to deal with complex shapes efficiently. In order to solve an IBM-based problem, the immersed boundary is supposed to enclose a face that represents the computational domain. The immersed boundary location within the grid is identified via a tagging procedure, which is performed to separate the computational Eulerian nodes into two types, inside and outside the face. A detailed discussion about the different types of cell tagging will be presented in Chapter 2. A typical illustration of different cell types used in the IBM is illustrated in Fig. 1.9. The nodes adjacent to the immersed boundary are treated differently; they have been identified as interface interior cells if the cell lies inside the face or outside interface cell if the cell lies outside the face.

Moreover, benefiting from Cartesian grids, adaptive grid refinement can be achieved by using hierarchical grids (quad-tree) to control the grid resolution in the vicinity of the boundary for better resolution to describe the immersed boundary.

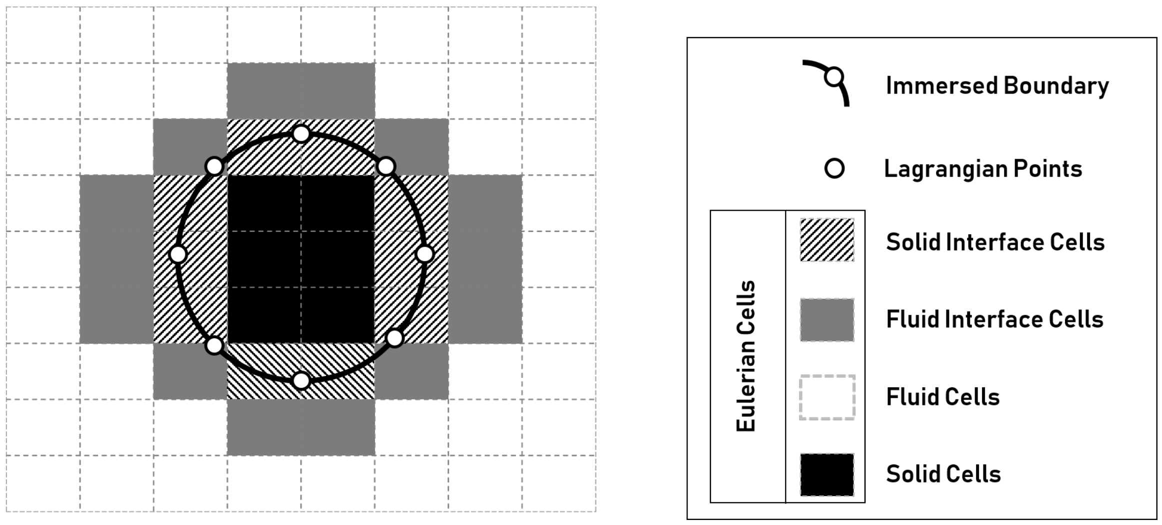


Figure 1.9 Classification of cells resulting from tagging procedure.

The difference between body-fitted methodologies and the IBM approach is the way the boundary conditions are applied. Since the IBM is a non-body-fitted approach, these are applied by reconstructing the solution from the geometric boundary to the nearest cell, called the interface cell. This characteristic makes the reconstruction scheme used in such a methodology critical for the accuracy of the results. After reconstructing the boundary conditions to the interface cells, the solution is resolved to the entire domain through a solver.

The IBM has been developed and applied to numerous applications that span compressible and incompressible flows with stationary or moving boundaries. A detailed review will be presented in the next chapter.

1.4 Objectives of the present work

The present research's principal goal is to develop an IBM algorithm integrated with hierarchical-overset grids to simulate unsteady aerodynamic inviscid compressible flows over 2D bodies in relative motion. The detailed objectives are as follows;

1. Development of a robust and efficient tagging approach for the geometry representation.
2. Development of a hierarchical grid generator based on curvature and proximity for the adaptation of the grid to complex geometry and complex flows.
3. Integration of the hierarchical grid to an Overset grid strategy to introduce the body motion.

4. Implementation of a robust reconstruction scheme for the application of boundary conditions.

A set of hypotheses are considered for the research problem:

- The flow viscosity effect is neglected so that Euler's equations are employed to model the flow.
- The moving of bodies always remains within the computational domain.
- Only translation movement will be implemented in the present work.

1.5 Research originality

The originality of the present work comes from different aspects :

- An original and efficient tagging approach that does not require looping over the entire domain to localize the immersed boundary to reduce the computational cost. This approach loops only over the boundary segments to identify the intersected cells. This is in contrast to the double loop on all cells and all boundary segments needed with the Ray-Tracing approach. So the proposed approach has an order of complexity lower than the Ray-Tracing.
- A robust reconstruction scheme based on an implicit least-square reconstruction scheme that allows reconstructing the boundary conditions of inviscid compressible flow-based problems efficiently.
- The integration of a two-dimensional IBM approach with the Overset-hierarchical grids to treat bodies in motion in compressible flows.

To the author's best knowledge, generally, the proposed integration of hierarchical-overlapping grids with an IBM scheme has not been reported to solve compressible flows yet.

1.6 Thesis structure

This document is organized into five chapters as follows:

- **Chapter 1:** This chapter presents the research context, the field of study, the problem definition, and how previous researchers treat this problem. Finally, the proposed strategy is presented, with the research objectives, originality, and thesis structure.

- **Chapter 2:** The literature on the proposed methodology is reviewed, and the progress achieved on each element of the proposed methodology is discussed. This chapter answers the questions: Why the proposed methodology is adopted? What is the state of the art of the field? How will the proposed methodology fill the gap between the elements?
- **Chapter 3:** This chapter presents the first part of the methodology, which includes:
  - The generation of hierarchical grids, the numbering scheme of quadtree meshes, neighbor localization, and grid balance.
  - The development of a tagging approach, the employment of tagging to apply local grid refinements using the hierarchical grid algorithm, the definition of refinement criteria and other grid control tools.
  - The integration of the developed algorithm with an overset strategy in mind and the generalization of data structure between the different components.
- **Chapter 4:** This chapter is the second part of the methodology and presents the development of a fluid solver based on Roe's scheme, the reconstruction scheme based on an implicit averaged least-square approach to apply boundary conditions. Next, the different types of boundary conditions for Euler equations are presented, and how there are treated by the reconstruction scheme. Finally, the introduction of moving grid terms in the Euler equations by means of the Arbitrary Lagrangian Eulerian (ALE) approach, to link the overset grid module to the solver.
- **Chapter 5:** This chapter presents the computer programming of the proposed methodology (class architecture), as well as the verification test cases conducted to verify and assess the order of accuracy of the proposed methodology. The results of the test cases are then presented and discussed.
- **Chapter 6:** It concludes the findings of the research project, the gains, the limitations and proposes future work opportunities.

CHAPTER 2 LITERATURE REVIEW

In the literature, there is a large body of studies and researches in diverse topics of aerodynamics using the IBM in different applications and different flow conditions. This review will focus on the application of IBM to compressible flows with moving boundaries.

2.1 Background of the Immersed Boundary Method

The IBM is categorized into two types according to the application of the boundary conditions: the Diffused Interface Method (DIM) and the Sharp Interface Method (SIM), as shown in Fig.2.1.

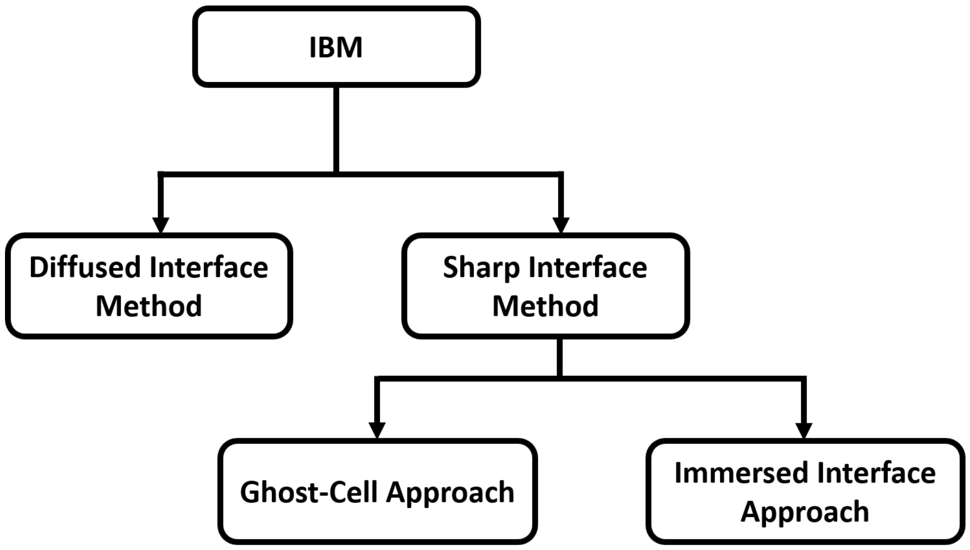


Figure 2.1 IBM Classification.

2.1.1 Diffused Interface Method (DIM)

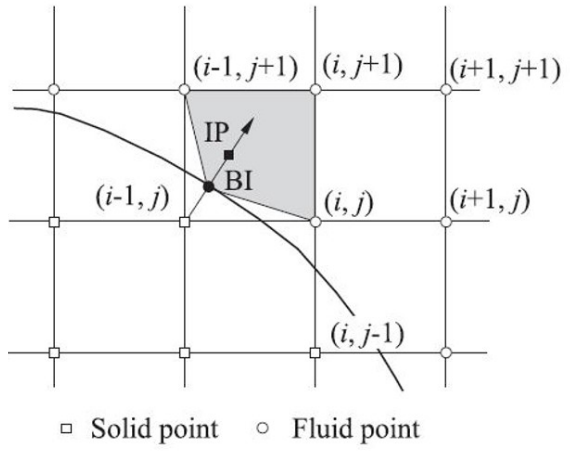
The DIM, a.k.a continuous forcing approach, accounts for the immersed boundary by a fictitious force or a penalization term that simulates the presence of the boundary on the background grid. In this methodology, introduced by Peskin [38], the boundary is smeared on the background grid nodes and not sharply identified. The representation of an elastic boundary is described by an external force field applied in the flow equations solved at the Eulerian grid points. The structure elasticity was modeled by mass-less Lagrangian points interconnected together by spring forces, with predefined stiffnesses. The connection between the flow and the structured discretization was made via a Discrete Delta Function (DDF), in such a way that the fluid velocity on the Eulerian background grid is interpolated to the Lagrangian boundary nodes, and the momentum is then forced from the boundary node to the nearest cells on the background Eulerian grid. Further developments were made by Kim and Peskin [39] to apply to a rigid body by considering the connection between Lagrangian points as springs with large stiffness. However, this did not result in a good behavior in the rigid limit for complex geometries [40], and drove the material stiffness representation to be considered as an ad hoc crucial parameter [41].

A feedback-continuous forcing approach has been presented by Goldstein *et al.* [42], based on the correction of the flow velocity to satisfy the boundary condition via a large set of adjusting parameters. This imposes another constraint to this approach, in addition to its considerable restriction for the time step size [43]. Applying that method to rigid boundaries has shown some shortcomings in computations [44] to simulate compressible flow over a cylinder. This was corrected by Qiu *et al.* [45]. Another shortcoming of the diffused interface approach was reported by Edwards *et al.* [46], which faced challenges for the simulations of rigid moving boundaries.

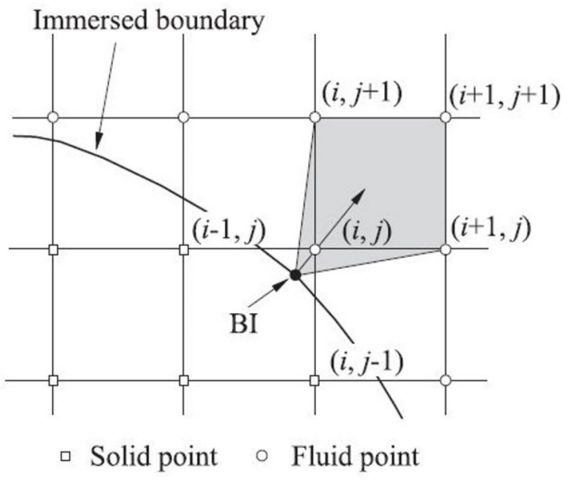
Most of the applications of the continuous forcing approach were applied in biological applications [38, 47–50], elastic boundary and multi-phase flows [44, 51], or porous medium simulations as in [52]. So that, this method is not commonly recommended for the application on rigid boundaries [43].

2.1.2 Sharp Interface Method (SIM)

The SIM, introduced by Mohd-Yusof [53], removed the time step constraint posed by the diffused interface method. This approach was initially applied for stationary boundaries and, often, to efficiently span stationary/moving and rigid/elastic boundaries [9]. In this method, also known as the discrete forcing approach, the boundary is sharply described, where the boundary condition is directly imposed. The flow variables are then reconstructed in the nearby cells using an interpolation scheme without relying on the formulation of material stiffness. This approach is globally seen as a local reconstruction approach that satisfies the boundary condition without momentum forcing [54]. The interpolation stencil in such a method uses the Eulerian grid in the vicinity of the immersed boundary. If a point adjacent to the immersed boundary lies in the fluid zone, it will be called Fluid Interface Cell (FIC), and if it lies in the solid zone, it will be called Ghost Cell (GC), as illustrated in Fig. 2.2. The flow variables are reconstructed on these points such that the boundary condition is exactly satisfied at the immersed boundary.



(a) Ghost-Cell approach.



(b) Immersed Interface approach.

Figure 2.2 Boundary condition reconstruction with Sharp Interface Method (SIM) [9].

So, depending on the way the interface points are considered as the unknown for the reconstruction stencil, the SIM can be further classified into :

- Ghost-Cell Approach: The solution is extrapolated to the solid ghost cell via the normal vector that passes through the ghost cell center toward the point of intersection of the normal vector with the boundary interface, named the Body Intercept (BI) point, as shown in Fig. 2.2a. The normal vector in this approach is based on the known surface information.

- Immersed Interface Approach: The solution is reconstructed directly at the fluid interface cell, relying only on the fluid neighbors and solid boundary information, as shown in Fig. 2.2b.

This classification is illustrated in Fig. 2.1. Several classifications were found to categorize the sharp interface method. Ones include the cut-cell approach as a descendant from the sharp interface method, such as Mittal *et al.* [40] and Sotiropoulos *et al.* [55]. Nevertheless, without loss of generality, the IBM is essentially a non-body-fitted approach, which leads to easily exclude the cut-cell from this classification. This perspective is similar to Yang's [41].

2.2 Eulerian mesh

The IBM has moved the difficulties related to grid generation to the tagging process. The generation of body-fitted mesh, especially with complex geometries, requires more robust meshing algorithms and more complex programming [41]. Classically, the solution on a Cartesian grid removes these constraints. However, it has some difficulties when dealing with steep curvature geometries or high gradients flows [56]. In order to fit these geometric and physical gradients, the corresponding fine grid resolution would be propagated to the entire computational domain, which in turn drives this refinement methodology inefficient. Furthermore, Peskin and McQueen [57] observed during applying a uniform Cartesian grid to their simulation that the order of convergence of the cells adjacent to the boundary was of first order, while the numerical order of convergence of the entire remaining domain cells was second order. These observations spur the essential consideration of a local refinement near the boundary to engender more grid points to capture the geometric and physical steep variations.

Numerous local grid refinement techniques have been proposed to describe the sharp interface better, enhancing the methodology's global accuracy. The reviewed approaches for local grid refinement are Adaptive Mesh Refinement (AMR), non-uniform Cartesian grid refinement, and unstructured grid refinement.

2.2.1 Adaptive Mesh Refinement (AMR)

The first implementation of an adaptive grid methodology in the context of IBM was made by Roma *et al.* [58], who employed a two-dimensional hierarchical adaptive grid refinement near the boundary. Thereafter, this AMR approach was widely adopted by several researchers to adapt the mesh resolution near the immersed boundary locally [11, 15, 49, 58–83].

Basically, the AMR is based on the successive and nested subdivision of each cell to quadrants (2D) or octants (3D), such that the *parent* cell is divided into four/eight *siblings* or sub-cells. Each subdivision occurs in a given level that describe the generation of the new *siblings*; this nested subdivision process takes place until fulfilling prescribed refinement criteria, which is often linked to a geometric feature (e.g., boundary curvature, proximity to another boundary, user-defined level, ...etc.) and/or physical feature (e.g., gradients, location of shock-wave, ...etc.) [84]. Fig. 2.3 illustrates the discretization in the quadtree, and its corresponding generated grid.

The wide use of the AMR promotes the integration of multiple grid levels, such that each refined level could be considered a Cartesian grid with a prescribed resolution, and hence several grids with several levels could be distributed over processor nodes, which could result in faster computations [82]. Furthermore, this progress allows efficient end-user libraries that handle the data passage through the hierarchical tree, such as AMReX [85], BoxLib [86], and Chombo library [87].

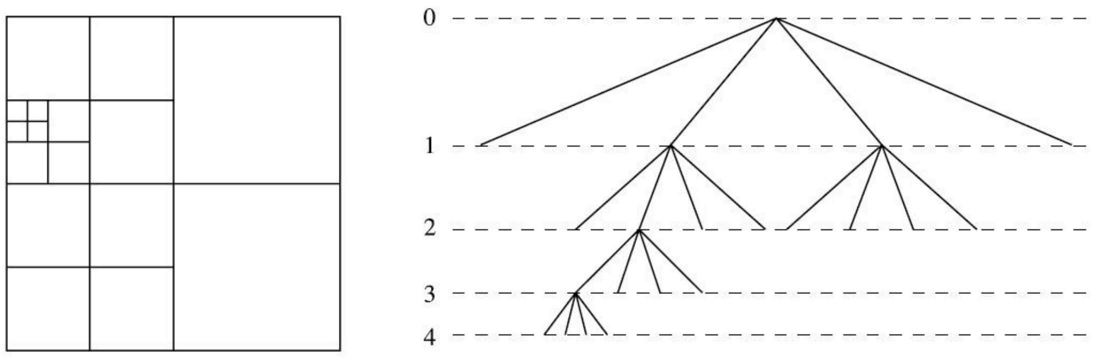
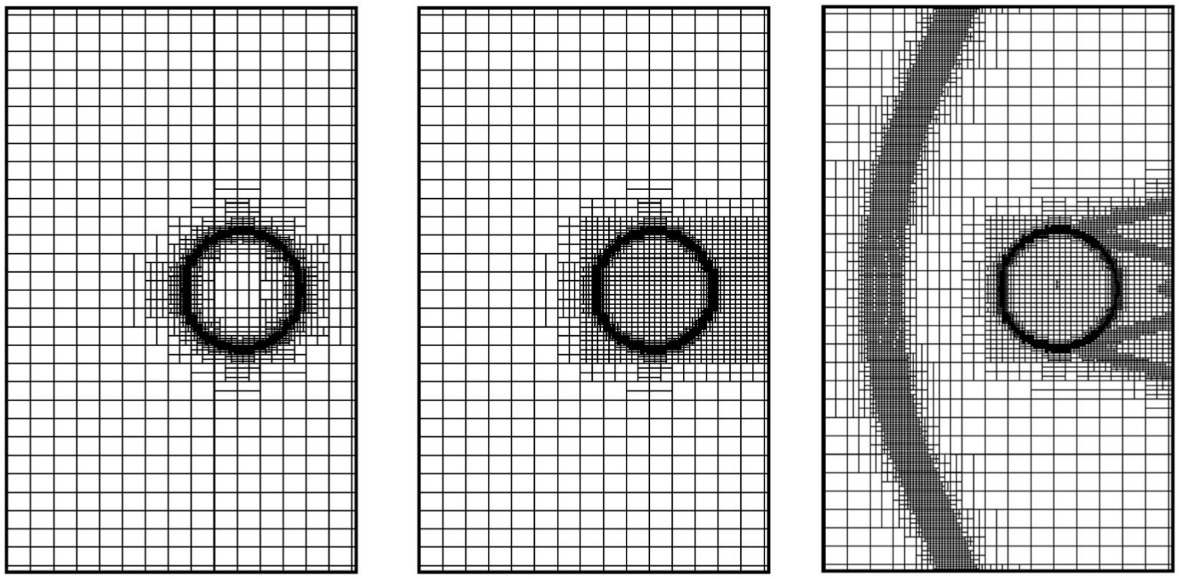


Figure 2.3 Illustration of quadtree levels representation [10].

Another similar approach to the hierarchical local grid adaptation was introduced by [88] and employed by [61, 64, 73], called the *iblanking* approach. This approach is the reverted version of the conventional hierarchical refinement procedure. In other words, it is a hierarchical coarsening procedure, the initial grid is the finer grid, and instead of applying refinement criteria, a coarsening criteria is applied to coarsen the computational domain moving away from the boundary.

The approaches mentioned above (hierarchical refinement/coarsening) generate at each level of refinement/coarsening the same number of *siblings* with the same aspect ratio, i.e., all subdivisions are equal in all directions (isotropic subdivision) of quadrants or octants. Wang *et al.* [59] introduced an *anisotropic* hierarchical grid refinement approach to simulate the flow field over F-16 aircraft. This approach is a 2^N tree-based approach that supports directional grid adaptation, where N is the dimension of directional coordinate. This approach can attain the same mesh resolution with fewer cell counts compared to the isotropic hierarchical scheme. As an example, a comparison between the number of cells generated by an isotropic versus an anisotropic grid generator for the same resolution parameters [59] results in $559,938$ versus $87,046$ cells.

The work of Wang [59] is a body-fitted application for the anisotropic hierarchical local refinement, but this methodology was extended and implemented for the IBM local refinement. de Tullio *et al.* [11] implemented an anisotropic grid to simulate viscous compressible flow over a sphere, shown in Fig. 2.4. This implementation was extended by De Marinis *et al.* by applying the anisotropic local refinement to the IBM to solve a conjugate heat transfer problem. Moreover, Tran [73] developed a ghost-cell-based IBM with anisotropic local grid refinement that holds both refinement and coarsening features. This was achieved by patching the refined siblings over the parents so that the parents' information was always stored and linked to the siblings.



(a) The immersed boundary. (b) The immersed boundary and the wake. (c) The immersed boundary, the wake, and the shock.

Figure 2.4 Local anisotropic grid refinement [11].

2.2.2 Non-uniform Cartesian grid refinement

Many researchers adopted the non-uniform structured/Cartesian grids to localize the grid resolution near the immersed boundary [12, 45, 56, 89–92]. It uses a concentration function that redistributes the density of grid nodes in a given direction. This is illustrated in Fig. 2.5, a non-uniform Cartesian grid used by De Palma *et al.* [12] for the solution of Navier Stokes equations to solve both steady and unsteady flows past cylinders and NACA0012 profile in a compressible flow. Unfortunately, no valuable details about the non-uniform grid refinement were reported by the IBM researchers.

2.2.3 Unstructured grid refinement

An additional approach to control the grid resolution locally is the use of an unstructured grid. Despite its capability to fit complex geometry, few publications reported using an unstructured grid with IBM [13, 93–95], Fig. 2.6. The use of unstructured grids for very complex cases impacts the cost of the computations, which is remarkably reported by [13, 95]. Their computational cost was comparably similar to their corresponding body-fitted

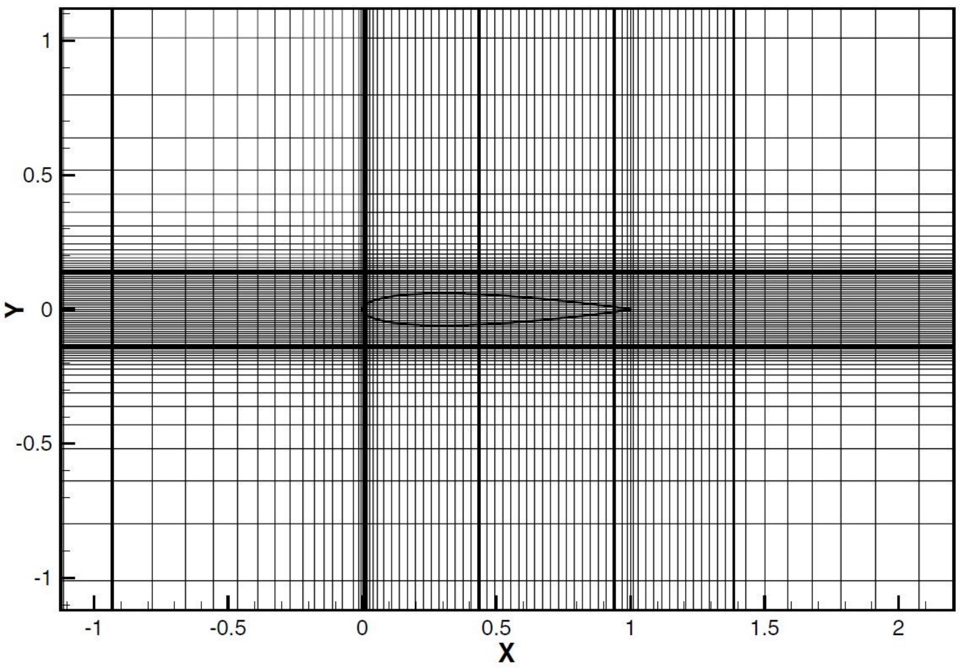


Figure 2.5 Local non-uniform grid refinement [12].

unstructured cases.

2.3 Tagging

Since the sharp interface is suitable for simulating rigid bodies, the Eulerian grid cells classification becomes a necessary process called "*Tagging.*" This process links the grid to the immersed boundary, often not aligned with the grid lines.

Different tagging approaches have been developed: the Ray Tracing approach, the Level Set (signed-distance), and the explicit minimum distance approach.

Regardless of the type of tagging approach, the outcome of the tagging process is the grid cell type, which is generally classified into fluid cells or solid cells. The solid cells adjacent to the boundary are classified as solid interface cells or "Ghost cells." Those adjacent to the fluid domain are then classified as fluid interface cells. This process is carried out in conjunction with the representation of the geometric boundary by a set of Lagrangian markers generated by the discretization of the boundary. In addition, the tagging process can deliver additional information along with the classification of the cells, such as the normal vector at the boundary segments, the intersection points between the boundary and the Eulerian grid, the intersection of the normal vector connecting the ghost cells with the boundary, as well

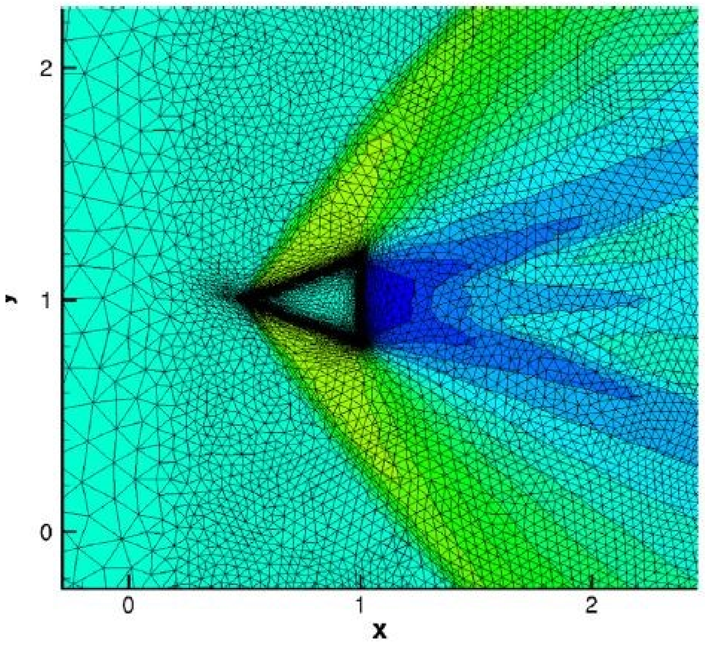


Figure 2.6 Unstructured grid refinement for IBM with the contours of density distribution over a wedge [13].

as the distance from the nearest fluid cell to the boundary.

2.3.1 The Ray Tracing approach

Several researchers adopted the Ray Tracing approach [11, 12, 14, 15, 62, 74, 78, 90, 91, 96–100], which is fundamentally based on the work of reference [101]. This approach casts a half-infinite ray from a given point (e.g., the cell center) in a specific direction (e.g., x-direction). The status of the cell, whether it is a fluid or solid, is determined by the number of intersections (even or odd respectively) encountered by this ray with the immersed boundary.

The number of intersections indicates how many times the ray penetrates different phases, as shown in Fig. 2.7. Starting with the ray launched from cell (A), which lies inside the fluid boundary, the ray encounters an even number of intersections, which means that the cell belongs to the fluid zone. Conversely, the ray launched from cell (B) encounters an odd number of intersections, which implies that the ray remains in the same phase so that cell (B) is solid.

The extension to three-dimensional applications was performed on simple geometries. Compressible flow over a sphere [11,15,62,78,90,96], compressible flows over cylinders [15,78,90], while other investigations were applied to compressible flows over wings [62, 78, 90]. The

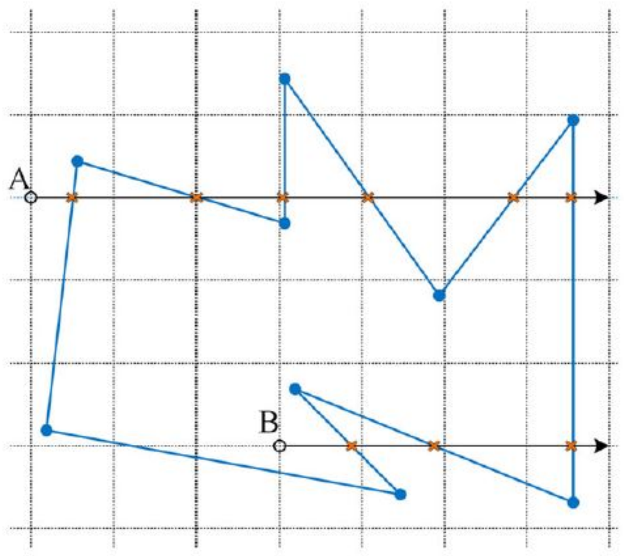


Figure 2.7 Illustration of ray tracing the concept [14].

performance of this algorithm is acceptable as long as the geometry is not complicated. The ray-tracing algorithm fails with complex geometries, and this issue was recovered by launching several random rays in different directions to assure an appropriate tagging procedure, which makes the computational cost high [96].

2.3.2 The Level Set approach

Another tagging approach is the Level Set approach. Osher and Sethian introduced this approach [102] which many subsequently adopted for the IBM application [15, 46, 71, 72, 77, 82, 98, 103–109]. This approach is basically for tracking propagating interfaces and adopted to tag the rigid and flexible immersed boundaries.

A boundary Γ is assigned as the zero level of a signed function ϕ that will always match the boundary with its evolution with time.

$$\phi(x(t), y(t), t) = 0$$

The zero level value identifies the boundary location at any time t such that:

$$\Gamma(t) = \{x, y : \phi(x, y, t) = 0\}$$

This identification procedure partitions the boundary interface's computational domain into fluid and solid zones. Each zone type corresponds to a sign of the level set function ϕ, e.g. $\phi > 0$ assigns the solid zone while $\phi < 0$ assigns the fluid zone, Fig. 2.8.

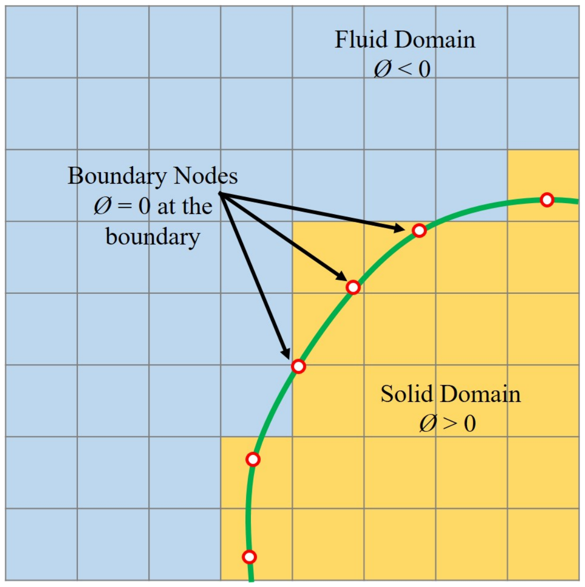


Figure 2.8 Illustration of Level Set function [15].

The usage of a Level Set function with simple geometries allows the implicit representation of the Level Set function. However, with more complex geometries, where the analytical representation is no longer achievable, the representation of the Level Set function is acquired numerically with a conjunction of other approaches, like Ray Tracing [15, 98].

2.3.3 The explicit minimum distance approach

A simple way to perform tagging presented by Luo *et al.* [80, 110] is the explicit minimum distance approach. After immersing an oriented boundary, a cross product is performed between the two vectors connecting the boundary segment to each cell center, a directional sign divides the computational domain into fluid and solid sub-domains, i.e., positive cross product means the cell center is inside, otherwise the cell center is located outside. A typical illustration of the methodology is shown in Fig. 2.9.

In order to decide whether the boundary intercepts a cell or not, the shortest distance between the cell center and the boundary points is evaluated; if it is less than the grid size, then the cell is intercepted. Further verification is made by checking the neighbors next to the intercepted cell to ensure that the cell has an incomplete stencil.

Despite the simplicity and the straightforward implementation of this approach, it is limited and valid only for regularly shaped objects, i.e., wholly convex or concave, not a mixture of

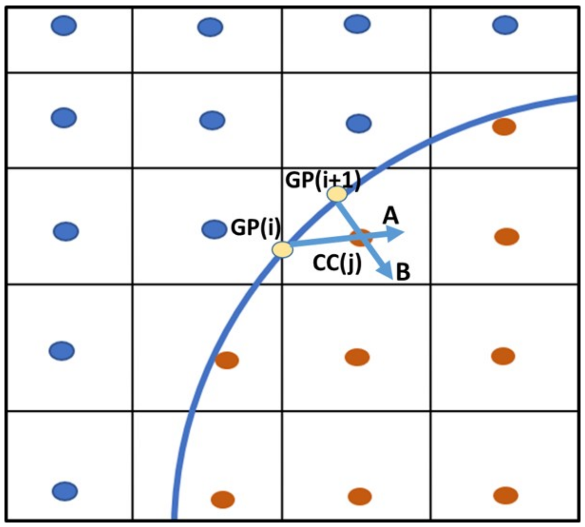


Figure 2.9 Illustration of explicit tagging.

both [80]. In addition, it necessitates looping over the entire grid cells to perform the tagging, which could be quite costly in high-resolution grids [43].

After reviewing different tagging approaches, the present work will propose a robust tagging approach to overcome the overhead cost and shortcomings of the presented methodologies. The proposed algorithm does not require to loop over the entire grid to identify the location of the immersed boundary; instead, it loops over the geometric boundary. A detailed discussion about this approach will be presented in Chapter 3.

2.4 Reconstruction schemes

The sharp interface method [53] is essentially formulated on the reconstruction of the flow variables at the Image Point (for the case of ghost cell approach) or the Interface Fluid Point (for the case of immersed interface approach) using an interpolation stencil composed from the surrounding fluid nodes and/or boundary nodes.

In that sense, various reconstruction schemes have been associated with IBM, such as the linear interpolation, implemented by [12,83,99]. Reconstruction by linear interpolation is acceptable with high grid resolution or for laminar flows. However, linear interpolation for high Reynolds number flows can lead to inaccurate prediction without adequate grid resolution, *i.e.*, the grid resolution near the boundary has to be considerably fine [111]. An extension to the linear interpolation is the Bi-Linear Interpolation (BLI)/Tri-Linear Interpolation (TLI) schemes for 2D/3D interpolation. For the application for compressible flows, the BLI has been implemented by numerous authors [14,52,71,72,77,80,89,90,92,104,106,110].

The Inverse Distance Weighting (IDW) scheme was also adopted by [11, 15, 41, 78, 96, 97, 108], and it is recommended for low Reynolds number due to its smooth behavior for that regime [112].

The moving/weighted/constrained least square is an extension to the least-square interpolation that is carried out for the IBM application by [14, 60, 92, 100, 107, 113]. The moving least-square enables the formulation of higher-order polynomials with flexible interpolation stencils; the variable order in interpolation stencil aims to prevent the system from being ill-conditioned. However, using high-order polynomials for interpolation could be a source of oscillatory behavior in the solution [15].

He *et al.* [112] invoked a Power Law interpolant to simulate 2D viscous compressible flow over NACA0012 airfoil and a wedge. Another interpolation scheme has been proposed, the Radial Basis Function (RBF), implemented by Liu *et al.* [91] to simulate viscous compressible flow problems. A reconstruction stencil of an interface cell is illustrated in Fig. 2.10, where the information is reconstructed from the fluid side and boundary conditions points.

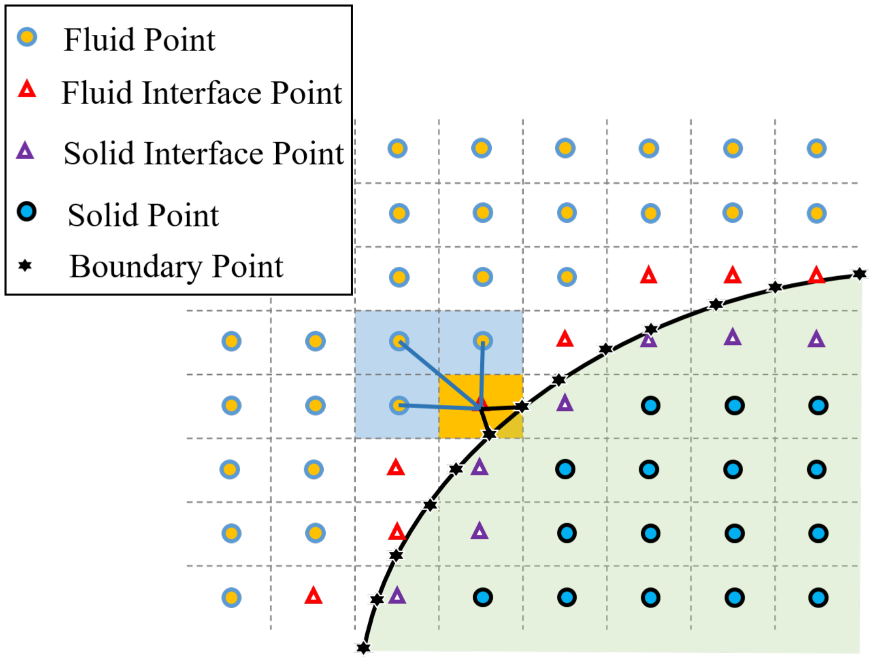


Figure 2.10 Reconstruction stencil of an interface cell based on three fluid cells and two boundary condition points.

2.5 IBM with moving boundaries

For a stationary immersed boundary, the computational domain is divided into two zones, fluid and solid, through the immersed boundary. When a boundary moves across the Eulerian grid, a given cell can lie in a different zone, *i.e.*, a fluid cell emerges in the solid zone, or vice-versa, a solid cell emerges in the fluid zone. These fresh newborn cells, called Fresh Cleared Cells (FCC) do not have a previous physical state, and it was reported [54, 114] that the FCCs are the reason for spurious oscillations that appear in the solution of moving boundary problems. These result from the spatial and temporal discontinuity caused by the appearance of such FCC.

To introduce boundary motion with the IBM, the tagging should be performed for each time step to locate the boundary position and re-identify the cell types. This re-tagging process is optimized by Mo *et al.* [108], trying to localize the re-tagging procedure by predicting the time step needed to repeat the tagging.

In order to keep a smooth solution during motion nearby the boundary interface, particular techniques are required for the FCC treatment. The field extension approach by Yang and Balaras [115] and direct reconstruction by Udaykumar *et al.* [114] are proposed to treat the spurious oscillations caused by the FCC. Another body-fitted approach was also introduced by Udaykumar *et al.* [116] but applied to the cut-cell approach, called the cell merging approach. It aims to merge the FCC with the nearest fluid cell.

2.5.1 Field Extension approach

The Field Extension was named due to the extension performed to the flowfield variables into the solid body to treat the FCC issue. In Fig. 2.11, the boundary is retracted to the left from time t^n to time t^{n+1}, and this leads to the appearance of a new cell $(i-1)$ as which was a Ghost Cell and becomes a Fluid Interface Cell. At time t^{n+1} the cell $(i-1)$ is designated as an FCC with no previous history in the fluid domain. The field extension procedure is then applied to the flow variables at time t^n to the cell $(i-1)$ in the following sequence:

1. At the time t^n tagging is performed, the cell $(i-1)$ is flagged as GC.
2. The flow variables for the cell $(i-1)$ are reconstructed using the fluid nodes at time t^n.
3. The reconstructed solution of the cell $(i-1)$ is then extrapolated to the cell $(i-1)$.
4. At time t^{n+1}, the value of the flow variables for the cell $(i-1)$ at time (t^n) are incorporated as previous "known" values.

5. Advancement in time stepping is performed and the procedure is repeated.

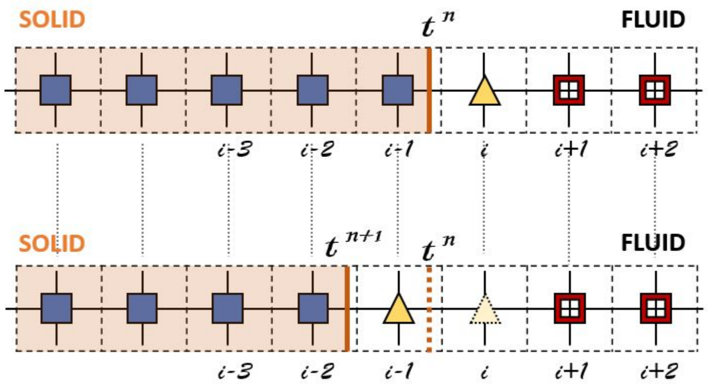


Figure 2.11 Field Extension Approach for FCC Treatment.

2.5.2 Direct reconstruction approach

The direct interpolation approach is similar to the scheme used for body-fitted moving grid methods. As shown in Fig. 2.12 for a retracted boundary to the left, the treatment of the cell $(i-1)$ at time t^{n+1} is directly achieved by the reconstruction of the flow variables at the new time t^{n+1} based on the physical state at the boundary ϕ_B^{n+1} and the computed values at cell (i) ϕ_i^{n+1}, such that:

$$\phi_{i-1}^{n+1} = \frac{\delta_\alpha \phi_i^{n+1} + \delta_\beta \phi_B^{n+1}}{\delta_\alpha + \delta_\beta}$$

Both approaches reduce the spurious oscillations [114], following non-body-fitted approaches. In the present work, the direct interpolation approach will be used to integrate the moving overset grids with the proposed IBM. This approach is the strategy to transfer the information between the overlapped grids during the grids assembly process.

2.5.3 Overset grids

Dealing with a moving boundary requires re-examining and re-visit every building element of the IB methodology, *i.e.*, meshing, tagging, and reconstruction. Further, each of these processes has to be repeated at each time step. Moreover, the time marching may constraint the usage of some of these elements. For example, the explicit tagging approach applied to

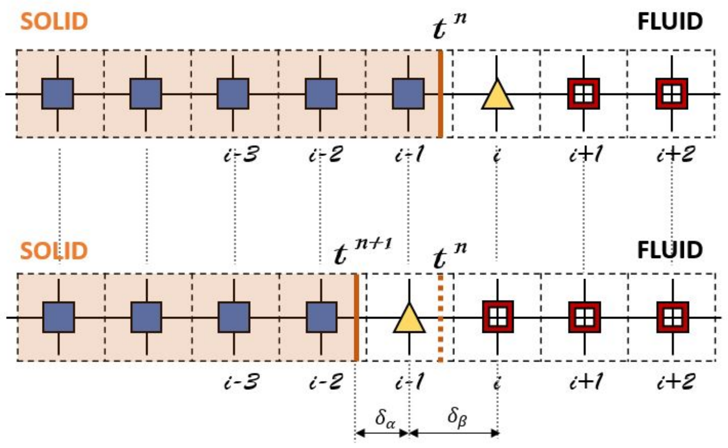


Figure 2.12 Direct Interpolation Approach for FCC Treatment.

moving boundaries is limited to a time step that does not exceed one grid per time step; otherwise, the tagging approach will fail to detect the boundary [80].

Generally, the process of re-meshing and/or re-tagging for each time step increase the computational cost, in addition to the reconstruction of the FCC. In order to reduce this, Borazjani *et al.* [16] used a hybrid IBM/body-fitted overset approach for incompressible biological application. This application was a body-fitted-like approach, where a curvilinear grid to enclose the boundary is used, as shown in Fig. 2.13. Another hybrid-IBM approach was presented that embodied the IB with the overset method [117, 118]; the integration was performed by using a body-fitted grid to describe the boundary interface; this grid is then overlapped with a Cartesian grid, where the communication between the two grids was performed via the interpolation between the two grids through a band of overlapping cells.

This type of integration permits a reduction of the computational cost of re-meshing. In addition, the interpolation is not made adjacent to the boundary, which reduces the influence of interpolation errors. This integration is accomplished by constraining the motion of the overset grid to one cell per time step.

2.6 Critical literature

The review of previous research progress in the IBM application for compressible flow opened new insights for the current project. A critical analysis has led to objective choices for some building blocks for the algorithm, and summarized as follows:

- **IB method:** The Immersed Interface Approach/Sharp Interface Method is suitable

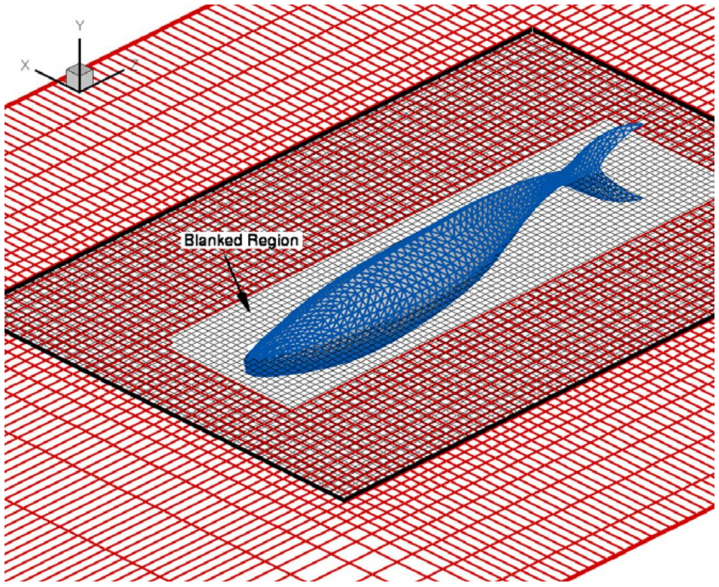


Figure 2.13 Overset grids around a mackerel, the geometry is descretized by a body-fitted tetrahedral elements [16].

for describing a physical non-diffusive geometry description. It is more convenient to use than the ghost cell approach, especially for the cases of compressible flow with discontinuities (shock waves), where the solution is not extrapolated at the ghost cell; the reconstructions are always made from the fluid side.

- **Tagging:** The reviewed tagging approaches are based on looping over the entire domain to localize the immersed boundary; this observation leaves some room to contribute with a new tagging methodology that relies on the geometry marching to reduce the computational effort.

- **Grid adaptation:** The hierarchical grid refinement is chosen to refine the Eulerian grid locally. This choice is based on its wide application with IBM, with its solid foundation and robustness.

- **Reconstruction scheme:** The bi-linear and IDW are the most used approaches for the IBM application. These explicit approaches are constrained to an ill-conditioned reconstruction system with insufficient information for the reconstruction. While, the Least-square reconstruction comes in the third place, it can be emphasized to evaluate a new robust spatial implicit version to reconstruct the BC information, where all the reconstructed cells will be coupled by the implicit system that will add more information to the system to avoid the ill-conditioned constraint.

- **Boundary motion:** To treat boundary motion efficiently, the overset approach is proposed to describe the body motion. This approach eliminates the re-tagging procedure at each time step.

To our best knowledge, no fully IBM approach was integrated with the overset method to introduce the boundary motion. This led to the decision to go through this idea to explore and exploit this strategy through the present work.

CHAPTER 3 HIERARCHICAL GRID GENERATION - OVERSET GRID AND TAGGING

Based on the literature review, the development of an IBM-based algorithm consist of integrating several building blocks: Geometry/Discrete topology, hierarchical grid definition and management, mesh tagging, overset tagging, grid refinement criteria, solution reconstruction, and the flow solver.

The structure of this chapter consists of seven sections. Section 3.1 addresses the different nature of the overall development, the fundamental functionality, and their integration into an overall code. Sections 3.2-3.7 tackle in detail the methodologies proposed to develop the first five building blocks (Geometry/Discrete topology, hierarchical grid, mesh tagging, overset tagging, and refinement criteria/grid controls). Reconstruction and flow solver building blocks are discussed in Chapter 4.

3.1 Global methodology

The proposed methodology defines each necessary building block for the development of the IBM code. It consists of seven main building blocks as listed:

- Geometry/Discrete topology
- Hierarchical grid
- Mesh tagging
- Overset grid
- Grid refinement criteria
- Solution reconstruction
- Flow solver

The main tasks performed by each building block in the focus of this project are listed in the following subsections.

3.1.1 Geometry/Discrete topology

This building block represents the immersed geometry by its discrete form, known as the discrete topology. The elements of a discrete topology are defined, formatted, and exported in a file as input to the tagging building block as the immersed boundary representation.

The list of tasks related to this building block:

- Define the basic entities of the discrete topology that represent the immersed geometry.
- Define a file format to store the discrete topology.
- Integrate the developed algorithm with the exported file format to read the discrete topology and share its information with other building blocks.

3.1.2 Hierarchical grid

This building block manages the overlapped grids and performs the necessary operations to adapt them according to their configurations. The tasks related to the overset grid building block are as follows:

- Analyse and identify the hierarchical structure for the quadtree grids.
- Identify an approach for hierarchical grid connectivity (*i.e.,* neighboring cells, cell sides, cell vertices).
- Implement a quadtree grid generator.
- Implement a grid refinement functionality to control the grid resolution.
- Define the data structure of the generated grids.

3.1.3 Mesh tagging

This building block is responsible for geometry tagging and grid topology control. It links the immersed geometry to the hierarchical grid generation through the tagging process, which localizes the immersed geometry within the grid and identifies whether the grid cells are inside or outside the geometry boundaries. The tasks related to this building block are as follows:

- Read the topology object and recognize its entities.

- Read the generated grid.
- Tag the interface cells by the geometry marching technique.
- Perform face discretization to classify the different types of cells inside the computational domain.
- Verify tagging on Cartesian and hierarchical grids.
- Conform the input/output data structure with other building blocks.

3.1.4 Overset grid

This building block is responsible for managing the overlapped grids and performing the necessary operations to adapt them according to their configurations. The tasks related to the overset grid building block are as follows:

- Identify the overlap region between grids.
- Define the list of activated/deactivated cells on each grid.
- Generate the list of interpolated cells within the overlapped grids responsible for transferring data between those grids.
- Define and integrate the output data structure of the building block with other building blocks.
- Verify and assess the coupling of the overset module with other modules.

3.1.5 Refinement criteria and grid controls

This building block provides an additional layer of grid control through the mesh tagging building block. The grid controls are proposed as follows::

- Control of grid resolution through the tagging.
- Define and implement different refinement criteria based on both geometry curvature and proximity.
- Introduce grid pre-adaptation capability.
- Export hierarchical grid elements in a vectored form.

3.1.6 Solution reconstruction

This building block is responsible for reconstructing the information in the interface cells. The reconstruction building block methodology is proposed as follows:

- Develop a robust reconstruction scheme to apply boundary conditions imposed for the flow solver.
- Generalize the reconstruction scheme to reconstruct the Overset information transferred between overlapped grids.
- Verify the reconstruction scheme for boundary conditions and overset reconstructions.

3.1.7 Flow solver

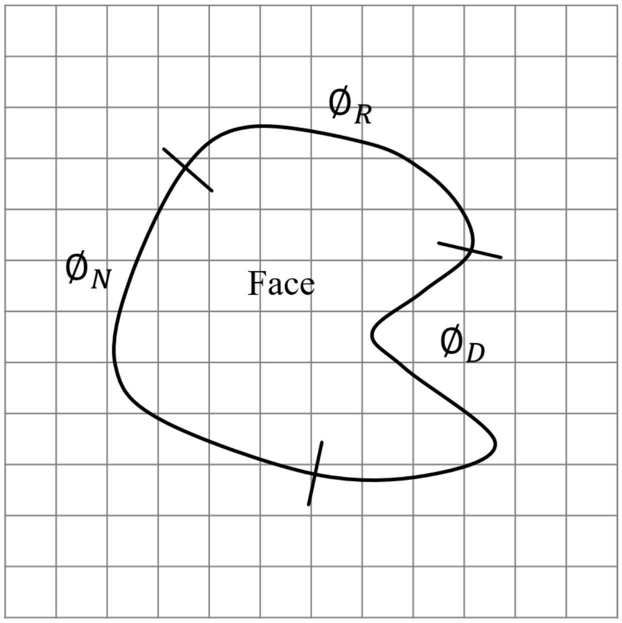
This building block is responsible for initializing the domain with a solution and advancing the solution in time. The main tasks related to this building block are:

- Define Euler's boundary conditions and assess the reconstruction scheme to reconstruct the defined boundary conditions.
- Implement Roe's scheme to solve the flowfield.
- Implement the ALE method to introduce the grid motion to Euler's equation.
- Conform the input/output data structure with all building blocks
- Verify and validate the developed algorithm.

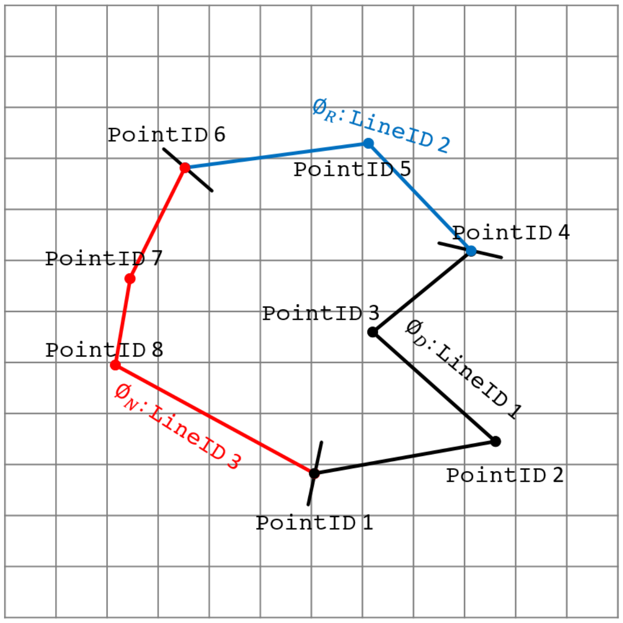
3.2 Geometry/Discrete topology

The foundation of the IBM concept is to immerse a computational domain bounded by a curve in an Eulerian grid to solve Partial Differential Equation (PDE) on the face bounded by this curve. To treat this geometry numerically, it has to be discretized to approximate its continuous description. This discrete form is called "*The Discrete Topology*," as shown in Fig. 3.1.

The discrete topology consists of faces bounded by loops (curves). These are oriented in the trigonometric sense. The enclosed surface/face corresponds to the computational domain, that is made up of the boundary and internal nodes.



(a) Geometry Curve.



(b) Discrete Topology.

Figure 3.1 Representation of the immersed geometry.

In the present work, these various entities are generated using GMSH open-source software, which uses the B-rep formalism. In addition, the software has a graphical user interface, a built-in CAD kernel, various meshing tools, and an Application Programming Interface (API) available in multiple computer languages.

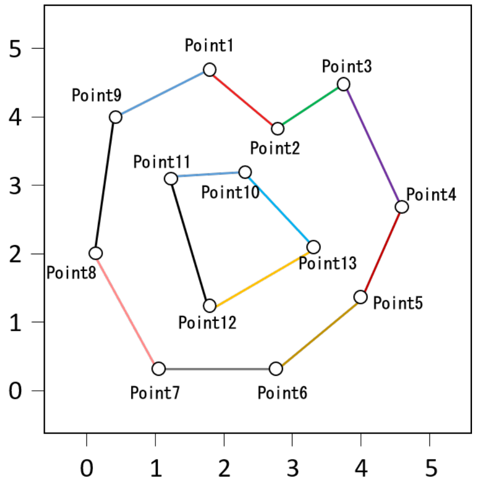
The discrete topology is represented by a set of entities to accurately represent the computational domain description and the actual geometry, as in Fig. 3.1. The employed entities for the discrete topology representation are illustrated in Fig. 3.2 and are described in the following part.

Points: The coordinate points that define the discrete curves are stored in two vectors (x) and (y). The position of each point in the list is given by `PointID`.

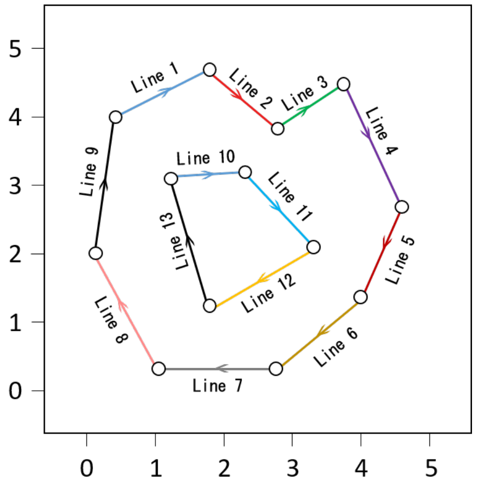
A posed hypothesis is to have a `PointID` at the location where there is a change in BC type at a given geometry line. As an example in Fig. 3.1a, three different types of BC (ϕ_R, ϕ_N, ϕ_D) are imposed at arbitrary locations along the geometry curve. Following the posed hypothesis, these locations must have three `PointID`s (`PointID1`, `PointID4`, and `PointID6`), as in Fig. 3.1b.

Lines: A list of `PointID's` defines a line or a polyline, which requires a minimum of two `PointID`. The position of a line/polyline in the list of lines corresponds to its `LineID`. The polyline can be an open or closed path (if the first and last points refer to the same `PointID`).

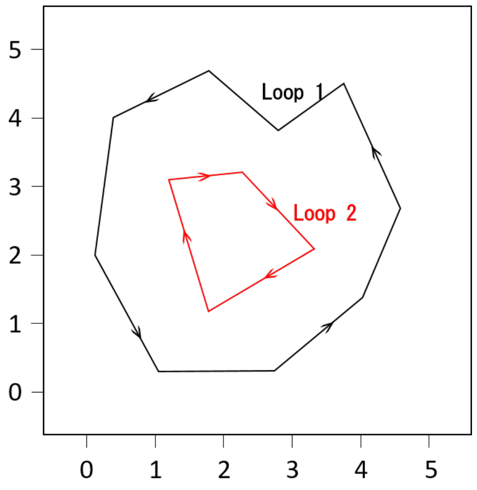
The presence of a `PointID` at the location of an alternated BC, as in Fig. 3.1, permits the imposition of the BC at its proper location. So that, BC ϕ_N is applied on the polyline `LineID3`, BC ϕ_D is applied on the polyline `LineID1`, and BC ϕ_R is applied on the polyline `LineID2`.



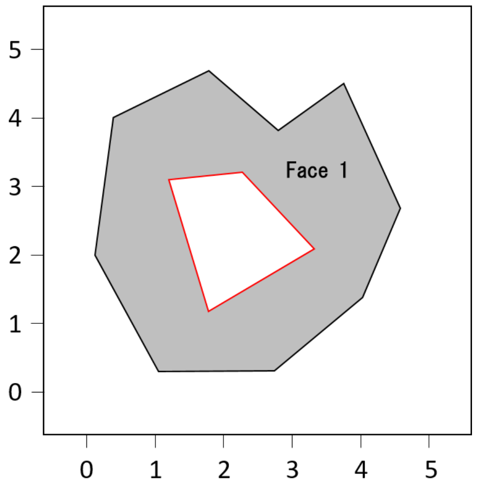
(a) Points of the discrete topology.



(b) Lines formed by the set of points.



(c) Two loops formed by the points and lines, inner loop is a hole.



(d) A face with a hole formed by the two loops.

Figure 3.2 Entities of discrete topology.

Loops: A loop is a sequence of oriented lines/polylines. The `LineID` identifies the lines defining the loop. This entity forms a closed loop, as a set of lines in a trigonometric sense. A `LineID` is positive if the direction of the loop follows the natural definition of the line and is negative if the direction of the loop follows the reverse definition of the line. Each loop has a `LoopID`, which identifies its position in the list of loops. A minimum of one `LineID` is required to define a loop.

Faces: A list of `LoopID's` defines a face. Each face is identified by a `FaceID`, which defines its position in the list of faces. A minimum of one loop per face is required to define the face. The first loop in the list is the outer loop of the face and follows the positive trigonometric sense. The following loops in the list define negative loops, or holes in the face, following the negative trigonometric sense.

A discrete topology file format is proposed to store this information. The *.tdt extension is used to store the file and stands for "Text Discrete Topology." This file is read by the $DiscreteTopology$ class, which reads and stores the entities' information in the class property and passes this information to other classes.

The same analogy could be used with other entities to represent 3D bodies. For example, for a 3D geometry represented by an elementary triangulation entity, two points represent a side or a line, three sides represent an element (triangle), a set of elements represent a surface, a set of surfaces represents a shell (loops in 2D), a set of shells represent the 3D volume.

3.3 Hierarchical grid

The Eulerian grid is represented by the hierarchical representation of quadtree grids to represent the immersed bodies accurately through hierarchical adaptation.

This approach aims to enrich an initial mesh by subdividing its elements locally into four sub-elements for 2D. The refined initial element is called the *parent cell* situated at a refinement level L_0, and the newly introduced sub-elements are called *siblings*, which are located at a refinement level L_1, as an example of one level of refinement. The four *siblings* generated by the cell refinement replace the *parent* cell in its four quadrants (SE, SW, NE, NW). Thus, a uniform subdivision into quads produces symmetrically isotropic cells with successive levels of refinement [84]. An illustration of the quadtree subdivision is shown in Fig. 3.3.

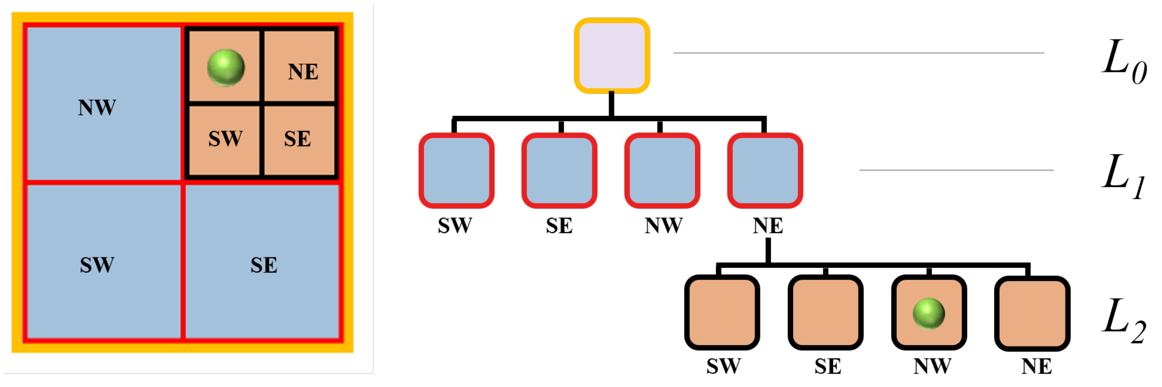


Figure 3.3 Quadtree representation with two levels of refinement. Refined siblings' location relative to the parent cell is indicated (SW, SE, NW, NE).

The depth of the tree is determined by the maximum levels of refinement in the tree, the terminal cells resulting in every refinement are called *leaves*, and any non-terminal cell is called an *internal cell*.

Based on the literature, hierarchical grids for IBM applications are widely used, as they combine the simplicity of a Cartesian structure for generating the grid, with the adaptivity of local refinement. This feature makes the hierarchical grid a suitable candidate as an adaptive grid scheme to the immersed boundary method. Several libraries were found in the literature that manage the hierarchical grids, such as `AMRex` and `py4est`. However, the proposed algorithm did not employ any of these libraries. This path is drilled to have complete control over the implemented code and help understand the details behind each implemented module better.

The linear quad/oct trees are one of the forms that represent the hierarchical data structure to describe the grid. Other representation schemes for quadtree are addressed in [84]. In linear quadtree, each leaf node of a quadtree may be represented by a unique spatial address named the location code (The node ID), and l is the node level. The way to encode or generate the node ID comprises the spatial location and the path traversed in the tree from root to leaf. Thus, we can find that the following characterizes the linear quadtree:

- Only the leaves are stored.
- The encoding used for each node incorporates adjacency properties in the four/eight principal directions.
- The node representation implicitly encodes the path from the root to the node.

The advantages offered by linear quad/oct trees representation as discussed in [84]:

- Pointers are eliminated.
- Reduction of storage demand compared to pointer-based.
- Ability to better performance with parallelization, allowing for High Performance Computing (HPC).

The linear quadtree is illustrated in the following subsection, covering the spatial addressing structure and numbering scheme.

3.3.1 Numbering scheme

The starting point is a Cartesian grid with $N_x \times N_y$ cells, where every cell is located at a level L_K, initially is said to be the unrefined level L_0, illustrated in Fig. 3.5a. The numbering pattern, as illustrated, starts from 1 to N_c, where $N_c = N_x \times N_y$. At the initial level L_0, all cells are initialized at level 0 (root level), and every cell is assigned by a local index $(i\ ,\ j)$ stored at each level. For subsequent levels of refinement, the cell numbering at the new level L_{K+1} is considered a continuation from the previous level L_K, as shown in Fig. 3.5b. As if a new layer of finer grid level starts from $N_C + 1$. So that, an `offset` at each level is calculated as:

$$\texttt{offset}(L_K) = N_c \frac{\left(2^{DL_K}\right) - 1}{2^D - 1} \quad , \quad K = 0, 1, 2, 3, ... \tag{3.1}$$

where D is the dimension (for binary tree $D = 1$, quadtree $D = 2$, and octree $D = 3$), and since the refinement advances one level at a time, so $L_{K+1} = L_K + 1$, and L_K is the current level of the refined cell before refinement.

For the Parent-Sibling relation, where for each newly generated siblings, the local index $(i_s\ ,\ j_s)$ is retrieved from the parent cell indices $(i_p\ ,\ j_p)$ as follows:

$$\begin{aligned} i_s &= 2\,i_p - 1 \quad , \quad 2\,i_p \\ j_s &= 2\,j_p - 1 \quad , \quad 2\,j_p \end{aligned} \tag{3.2}$$

The four sibling indices are formed by combining the four sibling indices in the following arrangement $(2\,i_p - 1\ ,\ 2\,j_p - 1)$, $(2\,i_p\ ,\ 2\,j_p - 1)$, $(2\,i_p - 1\ ,\ 2\,j_p)$, $(2\,i_p\ ,\ 2\,j_p)$ as shown in Fig. 3.4

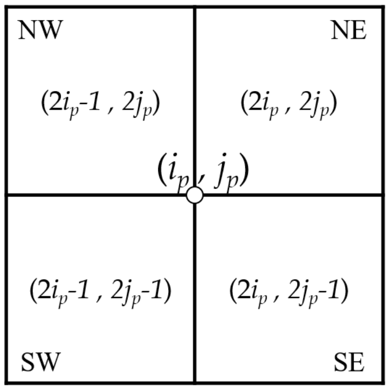


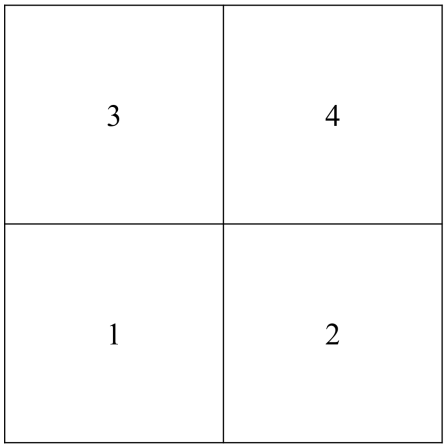
Figure 3.4 Siblings indices generated from a parent cell refinement.

Every generated cell inside the tree is defined by its own local index $(i\ ,\ j)$ and its refinement level L_K to generate a local address `LocalID`. This `LocalID` is defined as follows:

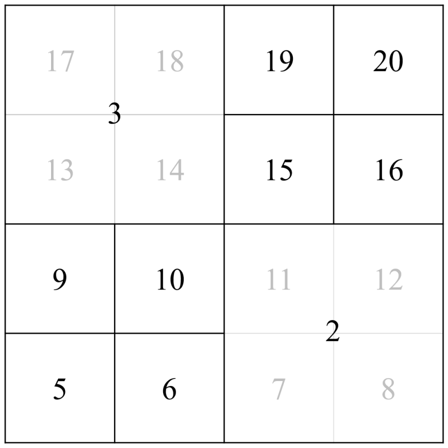
$$\texttt{LocalID} = i + (j-1) * (N_x * 2^{L_K}) \quad , \quad K = 0, 1, 2, 3, ... \tag{3.3}$$

The global address `cellID` is obtained by adding the local address `LocalID` of Eq. 3.3 to the offset presented in Eq. 3.1.The global `cellID` is a unique cell address used as the cell identifier and from which the grid generator retrieves all necessary spatial information.

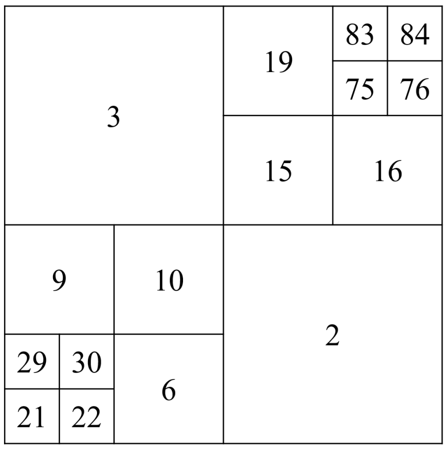
$$\texttt{cellID} = \texttt{LocalID} + \texttt{offset} \tag{3.4}$$



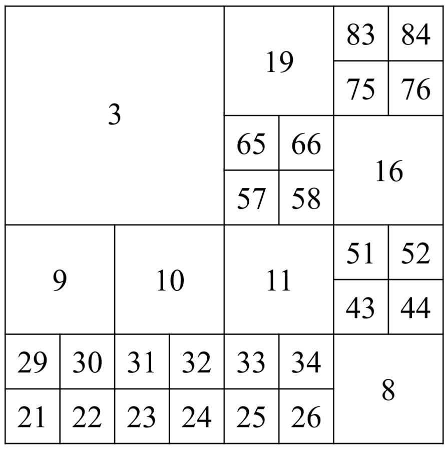
(a) Initial Cartesian grid $N_x \times N_y = 2 \times 2$.



(b) First level of refinement for cells #1 and #4.



(c) Second level of refinement for cells #5 and #20.



(d) Refinement of arbitrary cells to the second level of refinement

Figure 3.5 Numbering scheme adopted to represent a linear quadtree data structure.

3.3.2 Grid balancing

After performing all possible successive refinements in the computational grid, all the subdivided cells do not need to be at the same level. However, this consideration might result in an abrupt difference in cell levels, as illustrated in Fig. 3.6a. A grid is unbalanced if the level difference between two adjacent cells is greater than one. A *balanced grid* is a grid that conserves a maximum level difference of one, Fig. 3.6b.

The proposed procedure is inspired by [119, 120], where the information about the level difference between cardinal neighbors (WESN) is stored during the grid generation.

Initially, at the base Cartesian grid, all level differences between cardinal neighbors are set to zero because all cell levels are equal at this step, Fig. 3.7a. However, as the first cell, `cellID` $= 1$, is being refined, the level difference between the refined cell and its four cardinal neighbors will decrease by one in the direction (cell-neighbors) and increase by one in the direction (neighbors-cell).

If the NE sibling of the `cellID` $= 1$ is refined, the new siblings will be placed at two levels deeper relative to the cells #3 and #2. In this situation, to limit the level difference to ± 1, cells #2 and #3 are refined as long as the NE sibling of the `cellID` $= 1$ is refined. Thus, the refinement of the cells #2 and #3 one level deeper retains the level difference limit to ± 1 with neighbors when the recalled sibling is refined, Fig. 3.7c.

So, by tracking the evolution of the level differences between neighboring cells during the refinement, the algorithm captures the refinement that may lead to grid unbalance and perform the necessary refinements to the neighboring cells, according to the unbalanced direction (WESN), to limit the maximum level differences between neighbors to ± 1.

The procedure of a cell refinement with grid balancing is described in Algorithm 1.

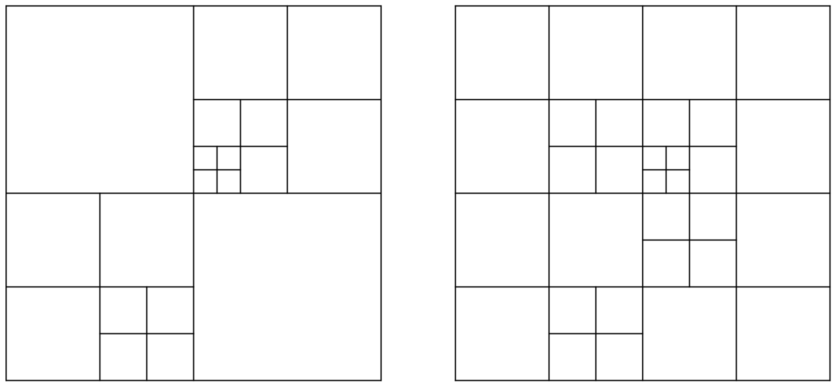
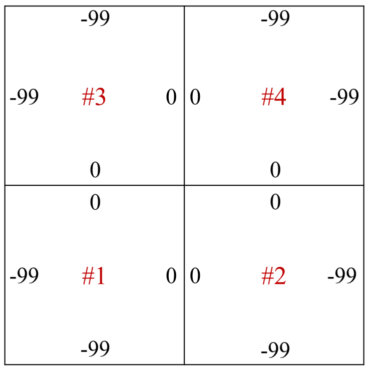

(a) Unbalanced grid. (b) Balanced grid.

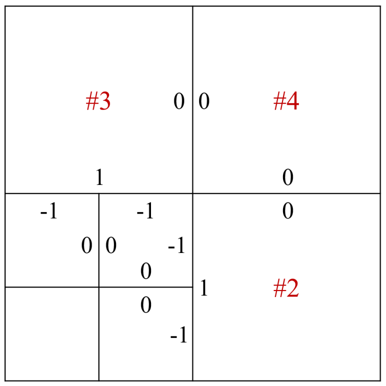
Figure 3.6 Hierarchical grid 2:1 balancing rule.

Algorithm 1 `Refine(cellID)` - Refine a given cell with grid balancing

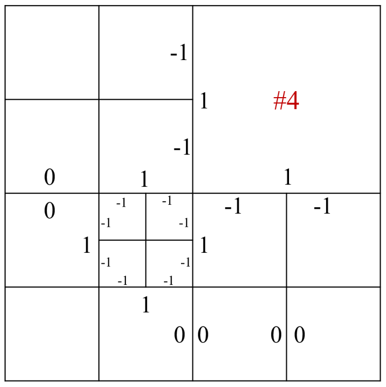
```
1: Get the identifier cellID of the given cell.
2: Get the cell level cellLvL.
3: Get the level difference in the four cardinal directions cellLvLDiff$_d$ ,   $d = (W, E, S, N)$.
4: Find the four cardinal neighbors neighborID$_d$   ,   $d = (W, E, S, N)$.
5: Get the level difference of each cardinal neighbor neighborLvLDiff$_d$,   $d = (W, E, S, N)$.
6: if neighborLvLDiff$_d = -1$ then
7:     Refine(neighborID)
8: end if
9: Create four quads (Identifiers, and vertices).
10: Update the level difference of cellID.
11: Update the level difference of neighborID.
12: Remove the parent cell cellID from the tree.
13: Update the tree.
```



(a) Initial grid with a level difference of zero with neighbors of the same level, and -99 if no neighbor is found.



(b) Refinement of `cellID` = 1 with new level differences with neighbors.



(c) Refinement of the NE sibling of `cellID` = 1. `cellID` = 2 and 3 are refined to keep the grid balanced.

Figure 3.7 Example of procedures of grid balance, using the level difference between neighbors.

3.3.3 Neighbors localization

Finding neighbors in a hierarchical structure efficiently is a demanding task. This is expected from the possibility of a cell having more than one neighbor when the neighbor has been refined. Furthermore, other cells could have coarser neighbors located at a higher level. Classical search-based algorithms could be an option, like the ones used by [121], but these algorithms are based on searching the cells that share common vertices/edges in multiple linked lists with pointers. The common feature between all these algorithms is the high computational cost of search algorithms, impacting the whole "package" performance.

In this work, an algorithm is adopted to locate cell neighbors efficiently with an order of complexity of *O(1)*, without depending on a search-based technique. Instead, it is based on simple arithmetic operations to locate the neighboring cells from their spatial address `cellID`.

Neighbors localization in hierarchical grids falls into three possibilities, as illustrated in Fig. 3.8:

- Neighbors of same levels.
- Neighbors of finer levels.
- Neighbors of coarser levels.

This approach profits from the natural numbering scheme and the adopted grid balancing approach to identify cardinal neighbors of a given cell.

Neighbors of same level

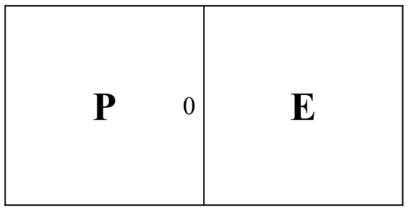
To locate cardinal neighbors of the same level for a given cell, the cell indices (i_{cell} , j_{cell}) and level L_K are sufficient information to find these neighbors. The neighbor's local address `localID` is calculated using Eq. 3.3 according to its position, *i.e.*, the neighbor's i or j index differs by ±1 to the cell (*East/West*: $i = \pm 1$, *North/South*: $j = \pm 1$), so that;

$$\begin{aligned}
\texttt{LocalID}_E &= (i_{cell} + 1) + (j_{cell} - 1) * (N_x * 2^{L_K}) \\
\texttt{LocalID}_W &= (i_{cell} - 1) + (j_{cell} - 1) * (N_x * 2^{L_K}) \\
\texttt{LocalID}_N &= i_{cell} + (j_{cell}) * (N_x * 2^{L_K}) \\
\texttt{LocalID}_S &= i_{cell} + (j_{cell} - 2) * (N_x * 2^{L_K})
\end{aligned} \tag{3.5}$$

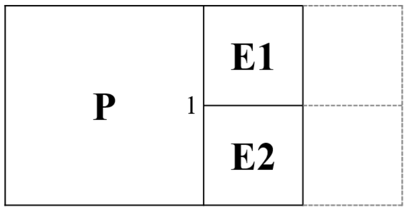
Since the neighbor level L_K is known, the offset is evaluated using Eq. 3.1. The neighbor's `cellID` is calculated using Eq. 3.4 evaluated in the corresponding cardinal direction, such that;

$$\texttt{neighborID}_d = \texttt{LocalID}_d + \texttt{offset} \quad , \quad (d = W, E, S, N) \tag{3.6}$$

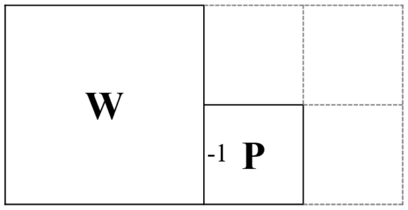
This configuration is represented in Fig. 3.8a, where cell P has a neighbor in East direction of same level. The level difference between cell P and its neighbor E is zero; in that case, the neighbor E is found using Eqs. 3.1, 3.5, and 3.6 directly.



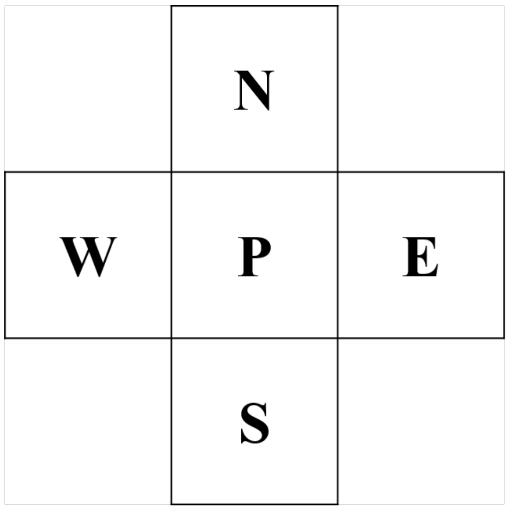
(a) Same Level Neighboring.



(b) Neighbors of cell P of finer level.



(c) Neighbors of cell P of coarser level.



(d) Cardinal Neighbors of Cell P.

Figure 3.8 Neighboring possibilities of balanced grid.

Neighbors of finer Level

The second possibility is that the cell neighbors are located at a finer level, as illustrated in Fig. 3.8b. The neighbor's level in East direction of cell P is less by one level; in that case, cell P will have two neighbors $E1$, and $E2$, representing the SW, and NW siblings of cell E.

The neighbor's localization of finer/coarser levels is always restrained to the grid balancing, resulting in a maximum number of neighbors of two in any cardinal direction.

The procedure to find neighbors of finer levels follows the same procedure of finding neighbors of same level. But instead of finding the neighbor of same level of the given cell, the neighbors are identified by finding the cell sibling's neighbors of same levels.

For a given `cellID` and a cardinal direction, the indices of the siblings $(i_s \ , \ j_s)$ of `cellID` are evaluated using Eq. 3.2. The cardinal direction determines which pair of siblings replace `cellID` to find its same-level neighbor. The neighbors are calculated using Eqs. 3.5 and 3.6. Table 3.1 and Fig. 3.4 indicate the siblings' indices $(i_{cell} \ , \ j_{cell})$ of the parent `cellID` that replace the parent cell index $(i_p \ , \ j_p)$ in Eq. 3.3 to find the corresponding neighbor according to the cardinal direction.

Table 3.1 Siblings pair of a parent cell and the corresponding indices in four cardinal directions

Cardinal Direction	Siblings pair	Sibling Index
West	SW	$i_s = 2i_p - 1 \ , \ j_s = 2j_p - 1$
	NW	$i_s = 2i_p - 1 \ , \ j_s = 2j_p$
East	SE	$i_s = 2i_p \ , \ j_s = 2j_p - 1$
	NE	$i_s = 2i_p \ , \ j_s = 2j_p$
South	SW	$i_s = 2i_p - 1 \ , \ j_s = 2j_p - 1$
	SE	$i_s = 2i_p \ , \ j_s = 2j_p - 1$
North	NW	$i_s = 2i_p - 1 \ , \ j_s = 2j_p$
	NE	$i_s = 2i_p \ , \ j_s = 2j_p$

The neighbor of finer level `neighborID` is then identified using Eqs. 3.3 and 3.6, after calculating the local index of the sibling pair and the sibling level as L_{K+1}, where L_K is the level of the parent cell.

Neighbors of coarser level

The third possibility is that the neighbor level to a cell is located at a coarser level than the cell, as illustrated in Fig. 3.8c. The neighbor's level of Cell P in the West direction is greater by one than the level of cell P, *i.e.*, the level difference of cell P in the west direction equals -1. This implies that cell P has only one neighbor on the west direction W, which is the west same level neighbor of the parent cell of P.

To find the same level neighbor of a parent cell, the parent index $(i_p\ ,\ j_p)$ is calculated from Eq. 3.2 based on the position of the siblings relative to the parent cell, or it can be directly calculated using a rounding function that rounds toward the nearest greater integer, as the function *ceil* in Matlab, so that;

$$i_p = ceil(i_{cell}/2) \qquad , \qquad j_p = ceil(j_{cell}/2) \tag{3.7}$$

These indices are then used to calculate `LocalID`$_d$ in a given direction d and `neighborID` using Eqns. 3.5 and 3.6 respectively, with a neighbor level of L_{k-1}.

This procedure is straightforward without searching inside the tree and eliminates many searches to find the neighbors in hierarchical grids. This feature enables the extension of the algorithm to 3D without any constraints, since the algorithm is of order of complexity of O(1). Algorithm 2 shows this procedure with the three possibilities.

Algorithm 2 `getNeighbor(cellID , `d`)` - Locate the neighbors to a `cellID` in a given direction d

```
1:  Get the cell level $L_K$, and index $(i_K\ ,\ j_K)$ of the given cellID.
2:  Get the level difference in the given cardinal directions cellLvLDiff$_d$, $d = (W, E, S, N)$.
3:  if cellLvLDiff$_d = 0$ then
4:      $i_{cell} = i_K \quad , \quad j_{cell} = j_K$
5:      $L_d = L_K$
6:  else if cellLvLDiff$_d = 1$ then
7:      Find the sibling couple in the given direction from Table3.1
8:      Evaluate the index $(i_s\ ,\ j_s)$ for each sibling
9:      $i_{cell} = i_s \quad , \quad j_{cell} = j_s$ (for sibling pair)
10:     $L_d = L_{K+1}$
11: else if cellLvLDiff$_d = -1$ then
12:     Calculate the index $(i_p\ ,\ j_p)$ for the parent cell of cellID from Eqn. 3.7
13:     $i_{cell} = i_p \quad , \quad j_{cell} = j_p$
14:     $L_d = L_{K-1}$
15: else
16:     The cell neighbor is a boundary
17: end if
18: Calculate LocalID$_d$ using $i_{cell}$ , $j_{cell}$ , $L_d$ from Eq. 3.5
19: Find neighborID$_d$ of same level using Eq. 3.6
```

3.3.4 Side list generation of hierarchical grid

The generation of side lists is an essential output from this class and as is required by the solver to evaluate the fluxes at the boundaries of the control volume (cell) at some stages, that requires the number of sides that contours this grid.

Since the grid generator is based on hierarchical grids, more than one side can construct the cell boundaries when the hierarchical refinement is applied to a given cell. Furthermore, in order to keep the solver vectorized, a data structure of the side list has to be established to offer the ability to describe the cell boundaries by the set of sides in vector form and pass this information directly to the solver, without the need to loop over the cells to identify the level difference between the cells to calculate the flux. All this information will be prepared and delivered directly in vector form to the solver, who will receive two linked lists that describe the grid's list hierarchy.

The sides lists consist of two main lists (Vertical sides list and Horizontal sides list). These lists are linked with other lists that describes the cell elements by the side ID (WESN) directions.

The sides numbering follows the grid `cellIDs` numbering, starting from coarser to finer cells following their address. The left vertical and the bottom horizontal sides are appended in the sides list until the grid interface for each traveled cell is reached. The four bounding sides are appended until reaching the grid interface. This description is illustrated in Fig. 3.9 for both horizontal and vertical list numbering.

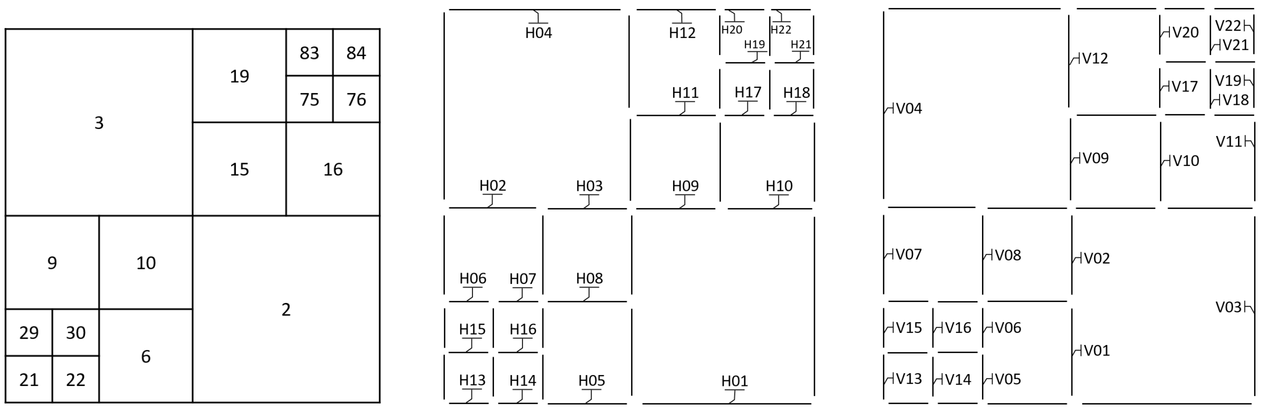
Each appended side will points to the `cellIDs` index in the grid container that shares this side. For example, for the vertical side list, each vertical side points to the left and right cells that share the vertical side. While for horizontal sides list, each side points to the bottom and top cells that share this horizontal side. An illustrated example of the vertical side list is shown in Fig. 3.9, and an example of a sample vertical side list is tabulated in Tables 3.2 and 3.3 .

Table 3.2 Example of a vertical side list

Side Index	Left Cell Index	Right Cell Index
V01	3	1
V02	5	1
V03	1	-99
V04	-99	2
V05	10	3
V06	12	3
V07	-99	4
V08	4	5
...	...	...

Table 3.3 List of cell index vs.`cellID`

Cell Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
`cellID`	2	3	6	9	10	15	16	19	21	22	29	30	75	76	83	84



(a) Hierarchical sample grid. (b) Horizontal sides numbering. (c) Vertical sides numbering.

Figure 3.9 Numbering of sides of a hierarchical grid.

3.4 Mesh tagging

The tagging is the process that generates an enriched link between the immersed boundary (discrete topology) and the hierarchical bounding grid. It aims to identify the different types of cells in the computational domain according to their relative position to the immersed boundary. Accordingly, the computational domain is divided into two regions (inside the face/ outside the face) regions where the face here is defined as the computational face containing the computational nodes.

The tagging algorithm functionality is listed as follows:

- Identify different types of cells to be treated according to their types: (i)cells inside the face (Inner Cells), (ii)cells outside the face (Outer Cells).
- Apply local grid refinement over desired targeted regions based on geometric-based/physics-based criteria specified by the user.
- Generate the necessary information about topology behavior inside the intersected cell, *i.e.*,(normal unit vector at grid boundaries, intersection points between the geometry and topology, the ID of the cut curve...*etc.*). This information is for boundary condition imposition, especially for the Neuman boundary condition, where normal fluxes are computed and will be discussed in detail in upcoming subsections.
- Retrieve the list of sides for the computational domain elements for flux calculations in the fluid solver.

This information underlines the importance of the data structure definition supplied to each building block, where tasks are mutually interconnected via the data structure.

3.4.1 Basic terminologies and definitions

This section introduces the terminologies and properties linked to the tagging that will be used often in the following sections.

This process is based on identifying the position of the cell center with respect to the immersed boundary. This cell-centered based tagging agrees with the control volume approach, where all necessary information is stored and represented by the cell center itself.

Some definitions are given to define the tagging key elements and illustrated in Fig. 3.10 as follows:

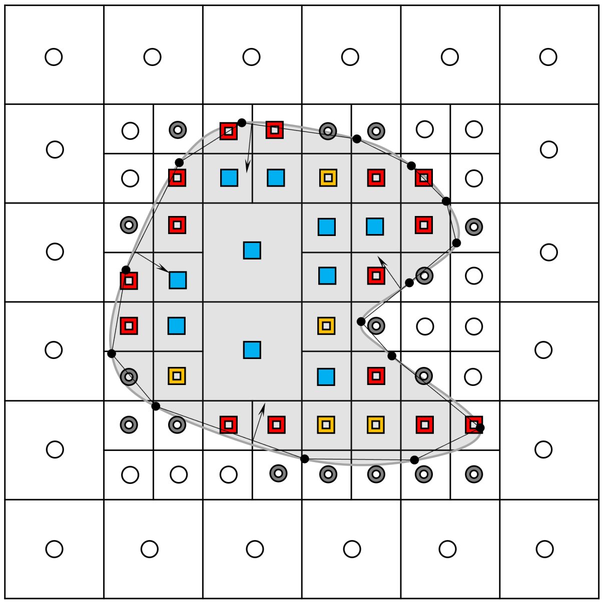


Figure 3.10 Cells classification of a tagged immersed topology.

- Inner Cell: An Inner Cell as represented by its cell center, has its cell center inside the face defined by the discrete topology. They are marked by either a solid or hollow square 🟦 , ▣.

- Face Cell (FC): A Face Cell is an Inner Cell having four Inner Cells neighbors, and marked with a solid square colored in blue 🟦.

- Interface Cell (IC): An IC is an Inner Cell where at least one of its neighbors is an Outer Cell. They are marked with a hollow square ▣. This type of cell is classified into two internal groups:

  - Intersected Interface Cell (IIC): An IICs are Interface Cells intersected by a discrete topology segment and colored by red 🟥.
  - Non-Intersected Interface Cell (NIIC): A NIICs are Interface Cells not intersected by the discrete topology segments and colored by orange 🟧.

- Interface Outside Cell (IOC): An IOC is an Outer Cell intersected by the discrete topology. They are marked by a hollow circle and colored by gray ◎.

- Outer Cell (OC): An OCs are cells where their centers are outside the face. They are marked by solid circles and colored by white ○.

3.4.2 Properties of intersected cells

Intersected cells are the cells that represent the geometric boundary, where the boundary condition will be applied. So, it is crucial to identify the set of properties and information that represent these intersected cells because, throughout the simulation, the immersed boundary is no longer used. Instead, it is represented by the layer of interface cells. Therefore, a set of information is always stored for each intersected cell (inside or outside the face). This information is the link between the cell and the part of the discrete topology that intercepts it.

Consider a cell P with dimensions Δx, and Δy, and vertices (P_1, P_2), intersected by a topology edge $\vec{L}_1$, as illustrated in Fig. 3.11. The topology edge will intersect the cell while entering, at its west side, at an intersection point ζ_{in}, and exiting the cell by intersecting its south side at the point ζ_{out}. An entering/exiting of a topology edge is considered as one cut, so the intersected cell P is cut only once by the edge $\vec{L}_1$. The valence is the normalized intersection fraction following the x-axis and y-axis, from left to right and bottom to top, and is evaluated as follows:

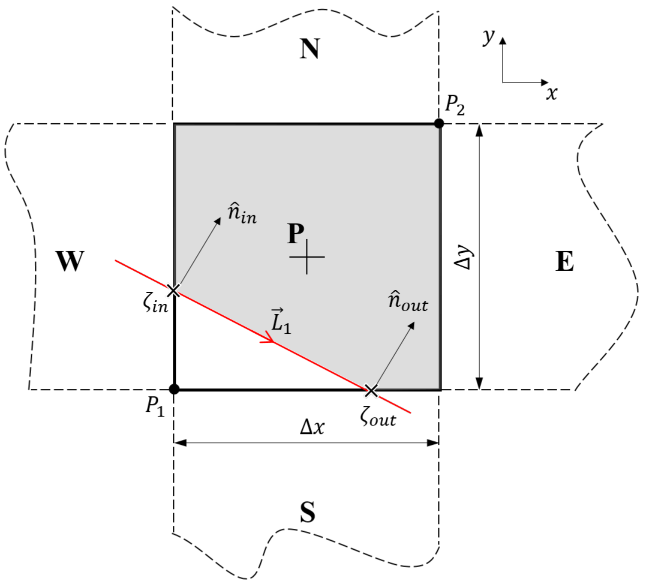


Figure 3.11 Properties of an intersected cell.

$$Valence_{x\,in/out} = \frac{\zeta_{x\,in/out}}{\Delta x} \qquad , \qquad Valence_{y\,in/out} = \frac{\zeta_{y\,in/out}}{\Delta y}$$

The role of the tagging algorithm is to establish the following properties for each intersected cell:

- Detect the number of cuts that intersect each intersected cell.
- Detect the entering and exiting sides of each cell.
- Find the intersection points and the valence of the intersection at entering/exiting locations.
- Calculate the normal vectors at the intersection points.
- Store the `LineID` and `LoopID` of each intercepting entity.

The tagging procedure begins by identifying the intersected cells, storing their properties, and using them to perform the complete domain tagging to categorize the remaining cells (FC or OC).

The algorithm starts by locating the first cell that contains a discrete topology point. Then, from this cell, the algorithm loops over all the topology sides until all loops inside the face are exhausted. During this geometry marching, each intersected cell is categorized either as an IIC 🟥 or IOC ◎. The process of the geometry marching is explained in detail in the next section.

3.4.3 Geometry marching algorithm

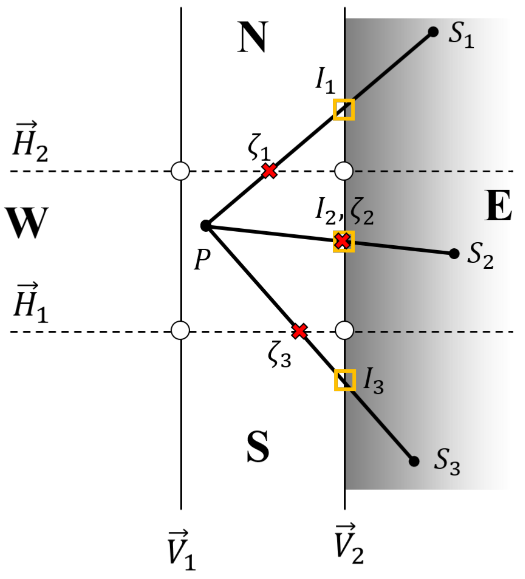
Various types of tagging approaches for IBM applications were presented in the literature. In this work, a boundary tracking approach is proposed for this step. To ensure a simple, robust, and cost effective procedure, the algorithm loops over the boundary points rather than the entire domain cells. The implemented approach aims to reduce the number of tested cells to identify the immersed boundary without relying on nested looping. A similar idea was adopted by the algorithm `x-ray` [122] that relies on reducing the numbers of cells that shed rays to identify the hole boundary on overset grids. Unlike the Ray-Tracing, a ray must travel from each cell inside the domain and be examined with every segment on the boundary.

For a given topology, the tagging is applied for each computational face. Each face is formed by a list of signed loops, and a list of oriented edges forms every loop.

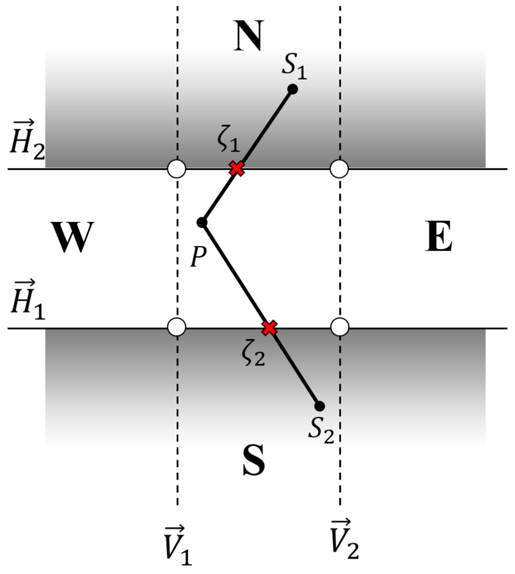
The geometry marching algorithm loops over every loop inside the computational face so that each tagging object points to its own face that corresponds to the face orientation.

Supposing that the first cell identified to begin the geometry marching is the cell that bounds the point P in Fig. 3.12, where P is the initial point of the segments that ends by the points

(S_1, S_2, and S_3) situated outside the cell. Then, to identify the cells crossed by the geometry lines, a methodology is posed to mark all the intersected cells and establish the necessary information during this marching.



(a) Step 1



(b) Step 2

Figure 3.12 Geometry marching procedure to find intersected cells.

The methodology begins by looping over the geometry segments, and during this scan, the marching algorithm allocates the cell that bound the initial segment point and verifies if the segment crosses the cell or not. If the segment crosses the cell, *i.e.*, the extent point of the segment is bounded by another cell; then a two-step test is performed.

Step1: The first step is to find whether the geometry line intersects one of the vertical rays $\vec{V}_1$ or $\vec{V}_2$ that bounds the left and right extent of the cell under test. If an intersection I is found with the vertical rays $\vec{V}$, the $y - coordinate$ of this intersection point I is then compared to the horizontal rays H_1 and $H2$, and then the exiting side of the intersected cell is determined.

As in Fig. 3.12a. The first segment $\overleftrightarrow{PS_1}$ intersects the vertical ray $\vec{V}_2$ at point I_1. By comparing the $y - coordinate$ of the intersection point with the horizontal rays $\vec{H}_1$ and $\vec{H}_2$, we can admit that the point I_1 is above $\vec{H}_2$. This implication gives one, and the only possibility is that the Northside of the cell is the exiting side, and the intersection point ζ_1 can be determined.

The same procedure can also be tested on the segment $\overleftrightarrow{PS_3}$, where it intersects the vertical ray $\vec{V_2}$ at point I_3. The $y-coordinates$ of the intersection point I_3 is below the horizontal ray $\vec{H_1}$, which implies that the Southside of the intersected cell is the exiting side, accordingly the intersection point at the exit side ζ_3 is determined.

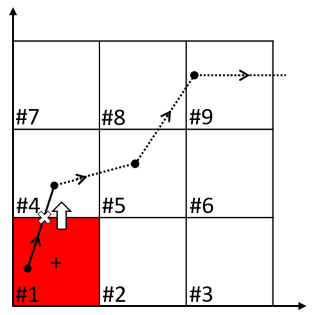
If none of the above scenarios apply, and there is an intersection between the geometry segment and the ray $\vec{V_2}$, then the only possibility is that the intersection point lies on the East side of the tested cell, represented by the segment $\overleftrightarrow{PS_2}$.

The procedure of the first step is valid for the illustrated case and its vertical mirror, where the intersections cross the vertical ray $\vec{V_1}$. In that case, the points I_1, I_2, and I_3 will intersect the North, West, South cell sides, respectively.

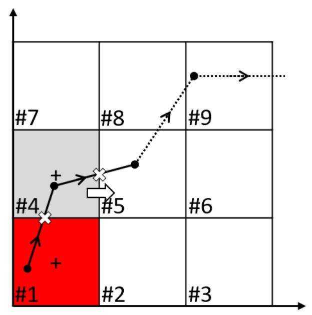
If the first step applies, *i.e.* there is an intersection between the geometry segment and the vertical sides of the cell, so the exiting sides of the tested cell are identified at this step. However, if there are no intersections captured between the geometry segment and the vertical sides of the cell, then the second step of the test is applied.

Step2: The second step detects possible intersections between geometry segment and horizontal rays $\vec{H_1}$ and $\vec{H_2}$ that bound the tested cell. For example, if the geometry segment intersects the upper ray $\vec{H_2}$, then the Northside is the exiting side of the cell, and the intersection point ζ_1 is determined. Otherwise, if the geometry segment intersects the lower ray $\vec{H_1}$, then the Southside is the exiting side of the cell, and the intersection point ζ_2 is determined.

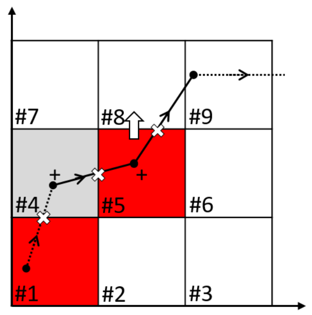
The first cell that bounds the first topology point is automatically identified as intersected and marked as a pivot cell. All pivot cells are intersected cells, but are named as pivot cell because these change with the geometry marching. The marching occurs when the algorithm identifies the exiting side of a pivot cell, which implies that the neighbor cell at the exit direction is the next pivot cell. So, the algorithm jumps to the next pivot cell and performs the two-step test to find the next pivot cell, and so on. This process is illustrated in Fig. 3.13.



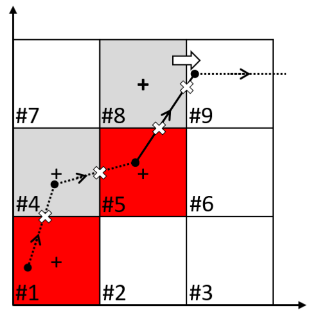
(a) Pivot cell#1.



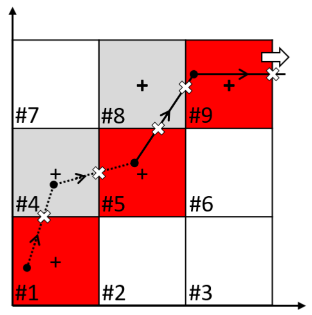
(b) Pivot cell#4.



(c) Pivot cell#5.



(d) Pivot cell#8.



(e) Pivot cell#9.

Figure 3.13 Geometry marching procedure on a Cartesian grid.

Similarly, the marching algorithm can loop over the immersed boundary inside a hierarchical grid. The only difference is that the algorithm checks the status of the level difference at the exit direction, and the next pivot cell is determined according to the local level difference and the position of the intersection point.

There are three possible configurations for the geometry marching procedure in a hierarchical grid, illustrated in Fig. 3.14:

1. The neighbor is situated at a finer level, as in North neighbor of cell#1.

2. The neighbor is of the same level as the East neighbor of cell #13.

3. The neighbor is situated at a coarser level than the East neighbor of cell #14.

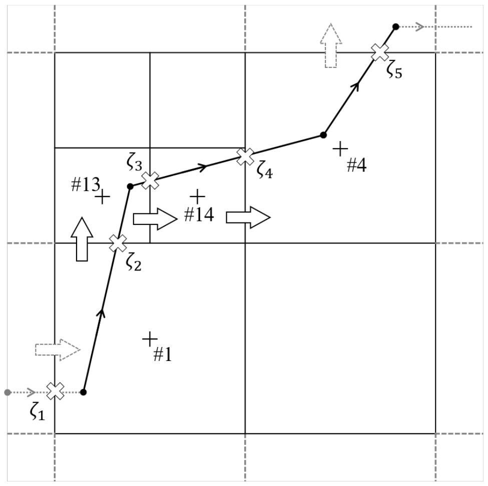


Figure 3.14 Geometry marching in hierarchical grid.

From these three configurations, the first requires special consideration. The neighbors in the exiting direction are finer than the pivot cell. In this case, the position of the intersection point is compared to the position of the pivot cell center. Accordingly, the intersection point is assigned to one of the two siblings next to the pivot cell.

In the North and South directions, the algorithm compares the x-coordinate of the intersection point with the x-coordinate of the cell center. For the East and West directions, the algorithm compares the y-coordinate of the intersection point with the y-coordinate of the cell center.

So, by applying the geometry marching algorithm to each loop of the computational face, the following information will be available:

- `CellIDs` of all intersected cells.
- The entering/exiting directions of the intercepting geometry curve.
- The location of intersection points at entering and exiting sides.
- The valence at entering and exiting sides.
- `LineID` that intersects the cell at both entering and exiting intersection points.
- `LoopID` that intersects the cell at both entering and exiting intersection points.

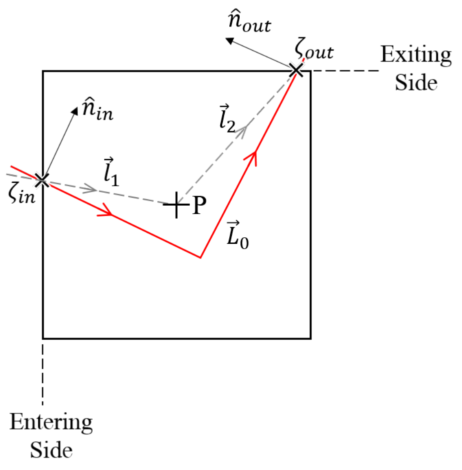
3.4.4 Types of intersected cells

During the geometry marching procedure, additional verification is carried out to determine whether the pivot cell is inside or outside the face. If the intersected cell is marked as inside the face, it will be tagged as Intersected Interface Cell 🟥, and if the intersected cell is marked as outside the face, it will be tagged as Interface Outside Cell ◎.

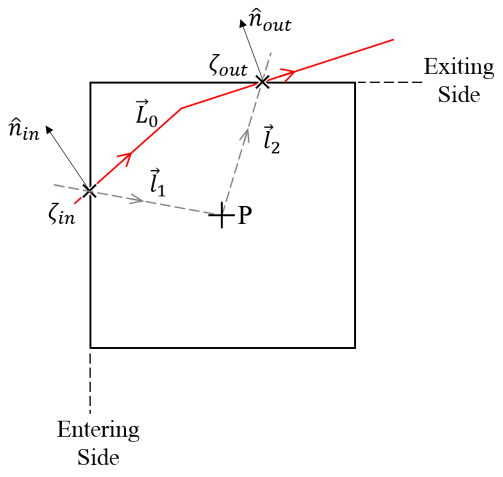
This step identifies the types of intersected cells and utilizes the cut information provided by the marching algorithm to calculate the normal vectors of the immersed boundary at the intersection points. Hence, the geometry is described inside the cell by two points of intersection and two normal vectors at these points. This information allows the algorithm to capture the variation in directions of the curve inside the cell and tags the cell without any ambiguity.

Using a vector product between the vector tangent to the geometry at the intersection points, and the vector connecting the intersection points to the cell center, allows to determine whether the cell center lies on the same side of the face. If the two cross products are positive, then the cell center is inside the face, and the cell is tagged as IIC. Otherwise, the cell center lies outside the face, and the cell is tagged as IOC. This procedure is illustrated for the two different cases in Fig. 3.15.

This completes the intersected cells properties with the normal vectors at the entering and exiting directions of each intersected cell and classifies the intersected cells into IIC 🟥 or IOC ◎.



(a) A cell inside the face IIC.



(b) Cell outside the face IOC.

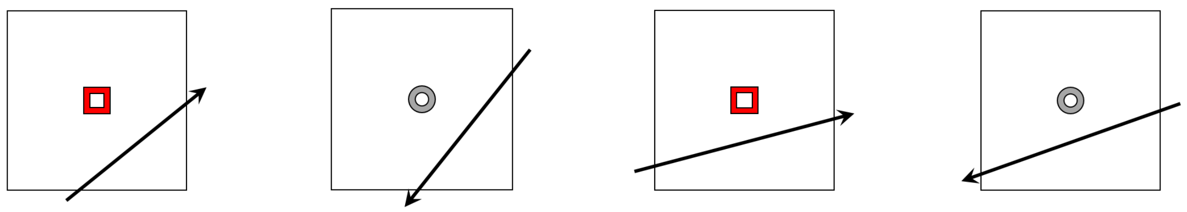
Figure 3.15 Illustration of two cases of intersected cells.

3.4.5 Types of Interface Cells (IC)

The analysis of the interface tagging is presented in this section to demonstrate different types of interface cells as a result of tagging.

The tagging of a closed topology is represented by a layer of watertight interface cells, as represented in Fig. 3.10, implying that some interface cells might be tagged as IC even though they are not intersected by the topology. The analysis of different types of IC is exposed by two configurations:

1. One-Corner cut IC: Shown in Fig. 3.16a, and Fig. 3.16b, the topology segment cuts two consecutive cell sides (WS, SW, SE, ES, EN, NE, NW, WN). The cell is identified either as IIC, or IOC according to the direction of the topology polyline.
2. Two-Corner cut IC: Shown in Fig. 3.16c, and Fig. 3.16d, two parallel cell sides are intersected (WE, EW, SN, NS).

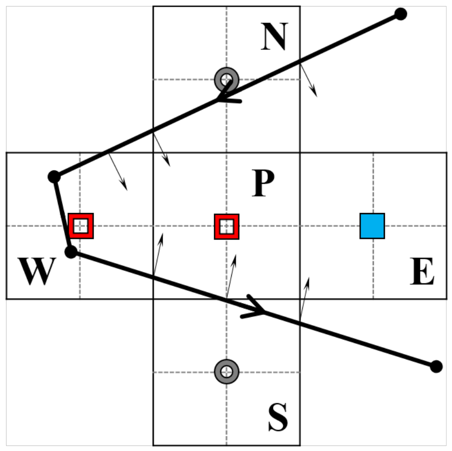


(a) One-corner cut IIC. (b) One-corner cut IOC. (c) Two-corners cut IIC. (d) Two-corners cut IOC.

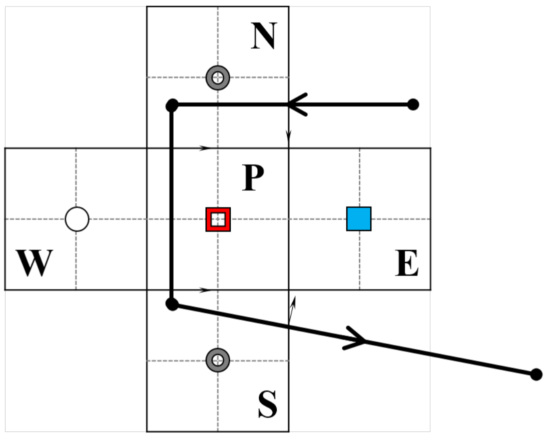
Figure 3.16 Four possibilities of intersected cells.

These configurations are illustrated on a global grid in Fig. 3.17. In Fig. 3.17a, the cells P and W are one-corner cut cells identified as IIC. Cells N and S are two-corners, and one-corner cut cells, respectively, identified as IOC. The interface is then represented by the intersection points of cells W and P.

The possibility of two-corners cuts IIC is illustrated in Fig. 3.17b, and in this case, cell W is considered an OC. This observation correlates the presence of an OC, usually perpendicular to the cut sides of a two-corner cut cell, relative to the direction of the polyline. In other words, if the polyline enters from the north side and exits from the south side, as illustrated in Fig.3.17b, this implies that the cells located in the west direction (the direction perpendicular to the cut sides) will be tagged as OC.



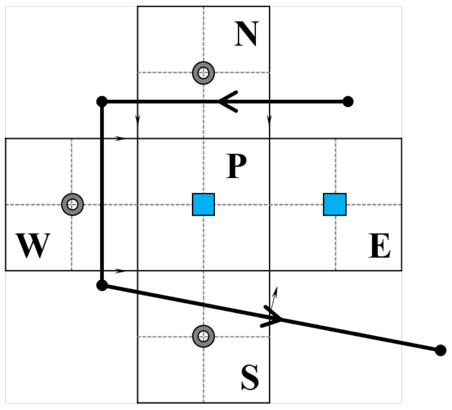
(a) One-corner cut inside P.



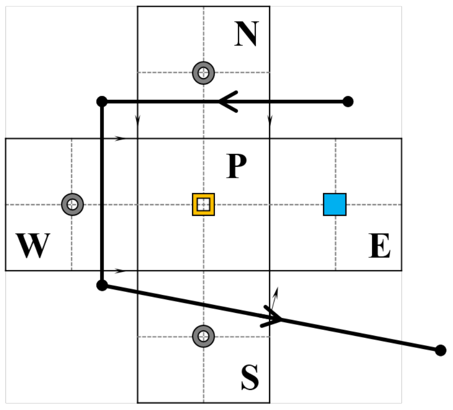
(b) Two-corners cut inside P.

Figure 3.17 one corner cut vs. two corners cut IIC.

If the position of the polyline in Fig. 3.17b extends to the left cutting cell W, as in Fig. 3.18, and cell W is tagged as IOC. Then, all the intersected cells are tagged as IOC, and there are no interface cells that can represent the current situation. From the observation of the preceding paragraph, a cell with a two-corner cut IOC must have at least one interface cell in the position perpendicular to the cut sides, relative to the direction of the polyline. So, cell P is considered a Non-Intersected Interface Cell, and this leads to the notion of the NIIC, Fig. 3.18b.



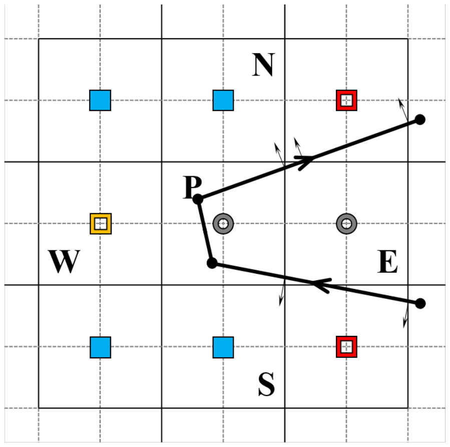
(a) Point P is not a NIIC.



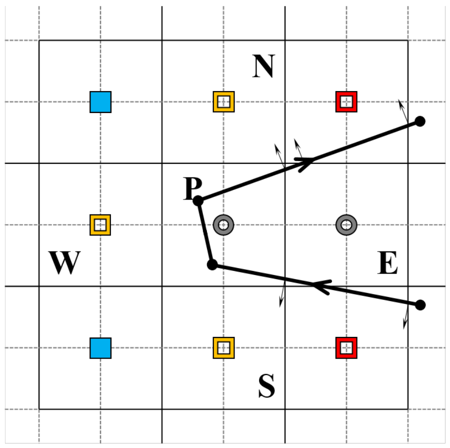
(b) Point P imposed as a NIIC.

Figure 3.18 Determination of NIIC by two-corners cut IOC of cell W.

A watertight interface layer must bind the computational face/volume to treat a closed boundary, and this requires the detection of Non-Intersected Interface Cell 🟨. Identifying the NIIC is mainly based on verifying if there is an IOC ◎ with two-corner cuts. Additional algorithm is used to verify the tagging of a watertight interface layer. It is based on verifying the neighbours of FCs, these neighbours should be Inner Cells. If one neighbor is detected as an IOC, so the FC is then tagged as NIIC. This procedure is illustrated in Fig. 3.19, where North and South neighbors of cell P are FC, by verifying their neighborhood, cell P is found as IOC, so cell N and S are then tagged as NIIC, as shown in Fig. 3.19b.



(a) Cells S and N are FC with IOC neighbor (cell P).



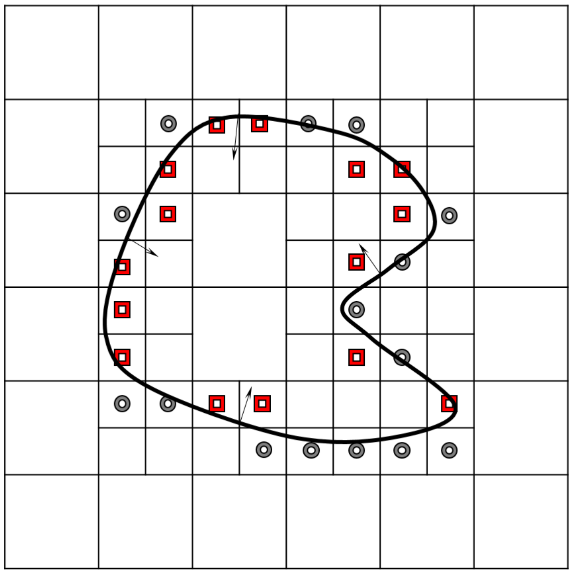
(b) Cells S and N are tagged as NIIC.

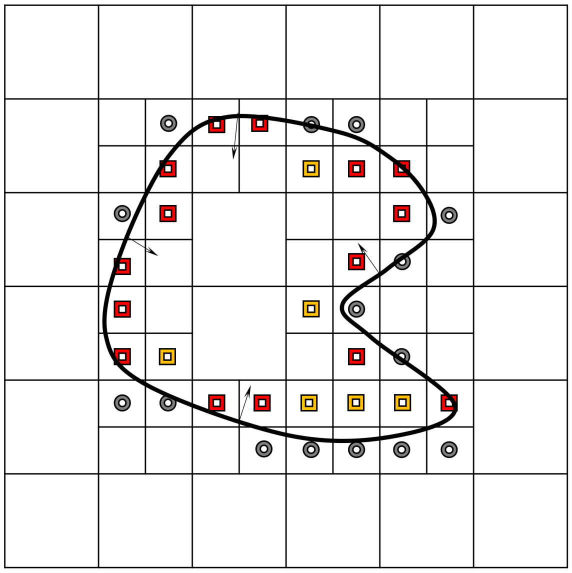
Figure 3.19 Interface tagging verification by neighborhood check.

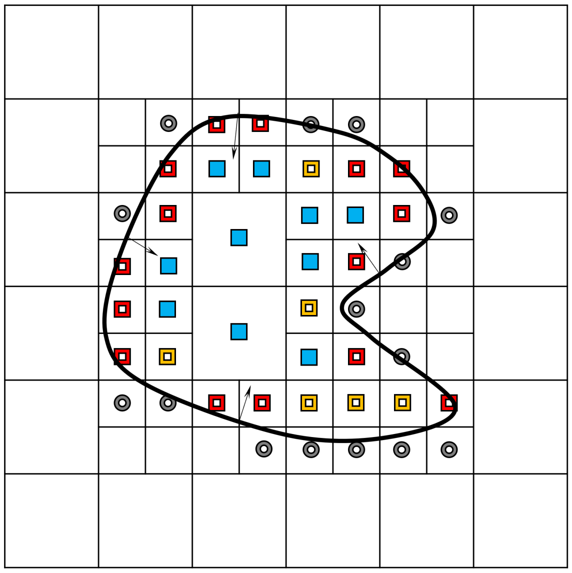
3.4.6 Face discretization

After the completion of the boundary tagging, (*i.e.* identifying the intersected cells [🟥, ◎], establishing all cut information inside each intersected cell, and identifying the presence of NIIC 🟨 needed to close the interface boundary), the next step is to identify the internal cells FC 🟦, where its four neighbors are inside the face. Then, locating the outside cells OC ○, where its cell centers are outside the face, and are identified by binary intersection algorithms.

Tagging the FC is performed using a *Flood-Fill* algorithm which starts by locating a cell inside the face and sweeps the face boundary-to-boundary until the face is filled. The remaining cells that are not tagged after this process are the OC. Fig. 3.20 illustrates the four steps of the complete tagging algorithm.


(a) Tagging of intersected cells IIC, and IOC.


(b) Tagging of NIIC to close the boundary.


(c) Tagging of FC by *Flood-Fill*.

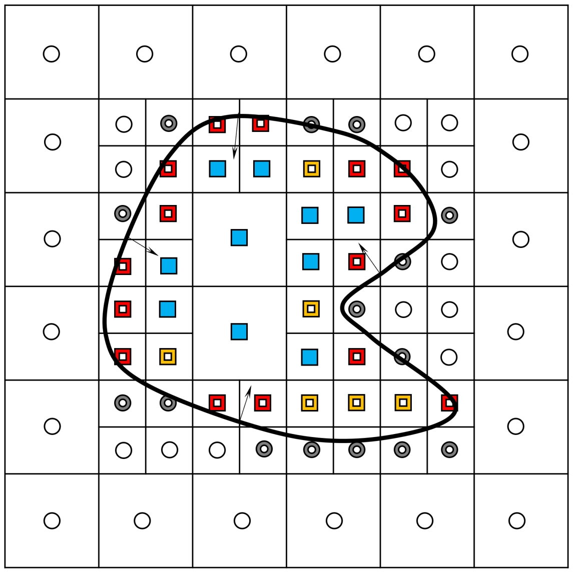

(d) Tagging of OC.

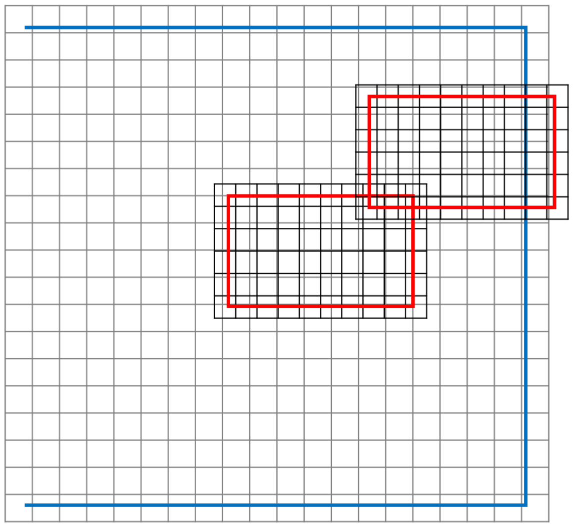
Figure 3.20 The four steps of the tagging algorithm.

3.5 Overset grid

In this section, we present the methodology adopted to integrate the overset grids into the overall strategy. The overset grid approach is adopted to deal with moving boundaries, essentially it simplifies the complex domains, principally with the treatment of moving geometries in IBM application. The moving geometry is introduced by a tagged overset grid, that will move entirely (tagged grid enclosing the immersed boundary) overlapping a reference grid. The usage of Cartesian overset grids turns the management of the grids into a simple task without resorting to libraries such as `Tioga`, which may defeat the simplicity of the implementation.

The hypothesis posed in our research to describe the overset approach is as follows:

- The computational domain consists of a background stationary grid fixed to the main global frame of reference and called the `Reference Grid`.
- All other grids that overlap the `Reference Grid` are allowed to move and called `Overlapping Grid`.
- There is no overlap between `Overlapping Grids` during the grid motion.
- All `Overlapping Grids` lie within the `Reference Grid` along the entire grid trajectory, as shown in Fig. 3.21.


(a) Invalid configuration: topology intersections `Reference-Overlapping` and `Overlapping-Overlapping`.

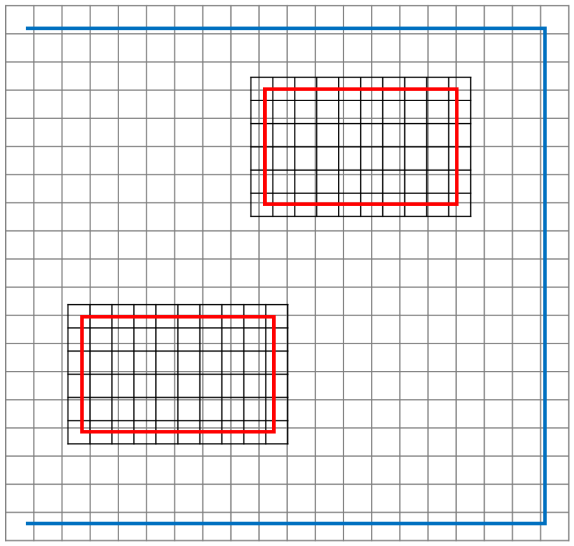

(b) Valid configuration: no Topology intersections.

Figure 3.21 Overset supported configuration.

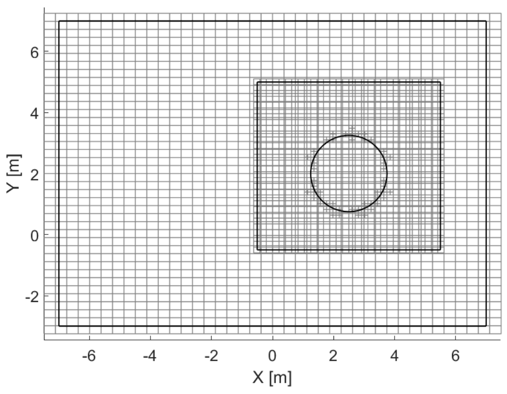
To demonstrate the data flow of the overset grid generation procedure, the *Overset* building block receives the refined and tagged grids from *Mesh Tagging* building block flagged by their type (`Reference`/`Overlapping`). Hence, the overset algorithm identifies the overlapping zones between the grids. This information identifies the enabled/disabled cells on `Reference Grid`. The disabled cells are those that lie within the overlap zone of the corresponding grid. This strategy is performed by two steps:

- **Hole Cutting:** This procedure is performed only on `Reference Grid`, Fig. 3.22b, and it aims to find the shadow of `Overset Grid` on `Reference Grid` and disable the cells that lie within this shadow, subject to some tolerance parameters.
- **Identification of Interpolated Cells:** This procedure aims to generate a layer of cells that contours the hole cut on `Reference Grid`, and the outer loop of `Overset Grid`. This type of cells is responsible for the transfer of information between the grids, known as *Interpolated Cells.* The overset tagging is shown in Figs. 3.22b, and Fig. 3.22c.

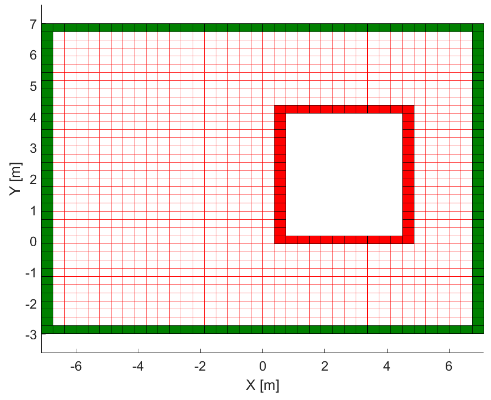
Each list of interpolated cells retrieves their flow field values from the values stored at the face cells of the opposite grid. This process illustrated in Fig. 3.23 is carried out such that the procedure of identification of disabled cells always ensures that the interpolated cells that bound the hole and the boundaries of overlapping grids have correspondent face cells in the opposite grid, the arrows in the figure indicate the transfer directions and the correspondence.

The *Overset* building process is responsible for generating the overset grids and consists of the following tasks:

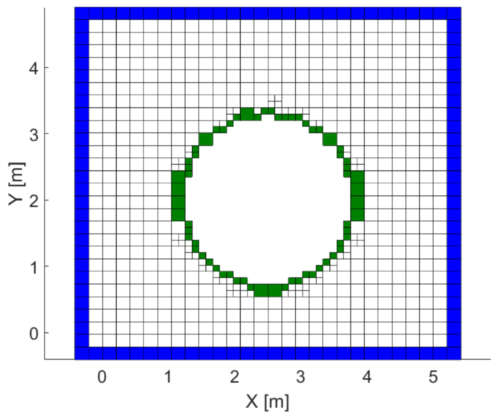
- Identify the overlap zones between `Reference` and `Overset` grids, Fig. 3.22a.
- Identify the list of disabled cells in `Reference Grid` (Hole Cutting procedures), Fig. 3.22b.
- Identify the list of interpolated cells necessary for the information transfer between grids, Figs. 3.22b and Fig. 3.22c.
- Export the updated grids into an object with the proper data structure of the main algorithm.



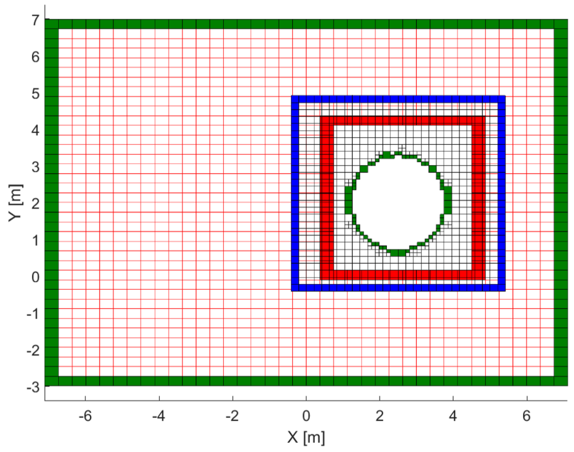
(a) Initial two hierarchical grids with topologies.



(b) Hole Cutting procedure on `Reference` grid, the shadow of the moving grid is disabled. The interpolated cells are filled in red, and interface cells are filled in green.



(c) Interpolated Cells are filled in blue on `Overlapping` grid, and interface cells are filled in green.



(d) Overlapped overset grids after treatment.

Figure 3.22 Overset treatment for overlapped grids.

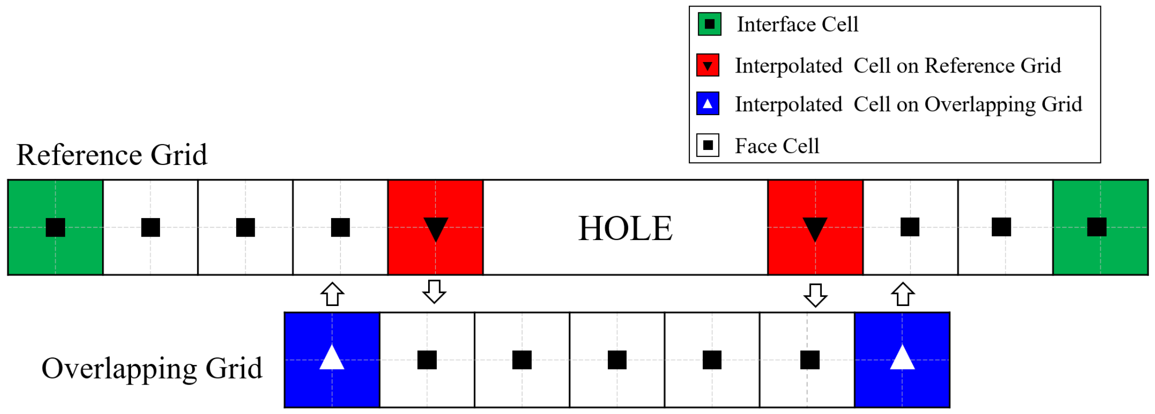

Interface Cell
Interpolated Cell on Reference Grid
Interpolated Cell on Overlapping Grid
Face Cell
Reference Grid
HOLE
Overlapping Grid

Figure 3.23 Representation of the interpolated cells conditioning for 1D overset configuration.

3.6 Refinement criteria

In addition to identifying and classifying the cell types of the computational domain, the tagging procedure also controls the generation of locally refined regions according to some input refinement criteria developed during this research. These control the local grid resolution via the tagging algorithm as the grid resolution is the foremost parameter that strongly affects the stability and precision of any numerical solution. The refinement criteria chosen as control parameters are broadly classified into two families:

Geometric-based criteria

1. **Criteria#1: Refinement based on the curvature of the geometry**

   This criterion aims to limit the number of points of the geometry contained by a given cell. The process begins by locating the set of cells containing geometric points at the Cartesian level of the grid. This process does not include any search; it is based on locating the index of the Cartesian cell that contains the geometric point.

   Given that the geometric point is $P_g(x, y)$, grid extends $X_{min}, X_{max}, Y_{min}, Y_{max}$, and the initial grid subdivisions in x and y directions respectively is N_x, N_y, with a resolution of dx, dy.

   Since the location of the geometric point is calculated in the initial Cartesian grid, so the indices I, J of the bounding cell can be found using the following formula;

$$I = floor[(P_{gx} - X_{min})/dx + 1]$$

$$J = floor[(P_{gy} - Y_{min})/dy + 1]$$

where the truncation to the lower bounds of the the indices is performed by using the built-in function $floor$.

The initial `cellIDs` that bounds the geometric points are then computed as :

$$\texttt{cellID} = I + (J - 1) * N_x$$

This approach is valid to locate any coordinate point inside the hierarchical grid. Given the cell level, L_K, and the corresponding resolution of the refined cell dx, dy. The bounding `cellID` can be found as:

$$\texttt{cellID} = I + (J - 1) * N_x * 2^{(L_K - 1)}$$

The cell is refined if the number of bounded points exceeds the given maximum number of points Nb_PTS_max. The bounded points are then transferred to the children of the refined cell. Dichotomy algorithm is used during this process to allocate the geometric points to the generated siblings, as illustrated in Algorithm 3. In Fig. 3.24, an illustrative example of the procedure to limit the number of geometric points inside a sample cell to four points.

Algorithm 3 Dichotomy algorithm with four quadrants to reallocate geometry points to the refined children bounding cells.

```
1: Get the cellID of the bounding cell
2: Get the nubmer of points $Nb\_PTS$ bounded by the cell
3: if $NB\_Pts > NB\_PTS\_max$ then
4:     Get the cell center position $x_c, y_c$ of cellID
5:     Refine(cellID)
6:     Get the four siblings
7:     if $(P_{gx} < x_c)$ && $(P_{gy} < y_c)$ then
8:         The point is allocated in the SW Quadrant
9:         $I = 2 * I_{\texttt{cellID}} - 1$
10:        $J = 2 * J_{\texttt{cellID}} - 1$
11:    else if $(P_{gx} >= x_c)$ && $(P_{gy} < y_c)$ then
12:        The point is allocated in the SE Quadrant
13:        $I = 2 * I_{\texttt{cellID}}$
14:        $J = 2 * J_{\texttt{cellID}} - 1$
15:    else if $(P_{gx} < x_c)$ && $(P_{gy} >= y_c)$ then
16:        The point is allocated in the NW Quadrant
17:        $I = 2 * I_{\texttt{cellID}} - 1$
18:        $J = 2 * J_{\texttt{cellID}}$
19:    else
20:        The point is allocated in the NE Quadrant
21:        $I = 2 * I_{\texttt{cellID}}$
22:        $J = 2 * J_{\texttt{cellID}}$
23:    end if
24: end if
```

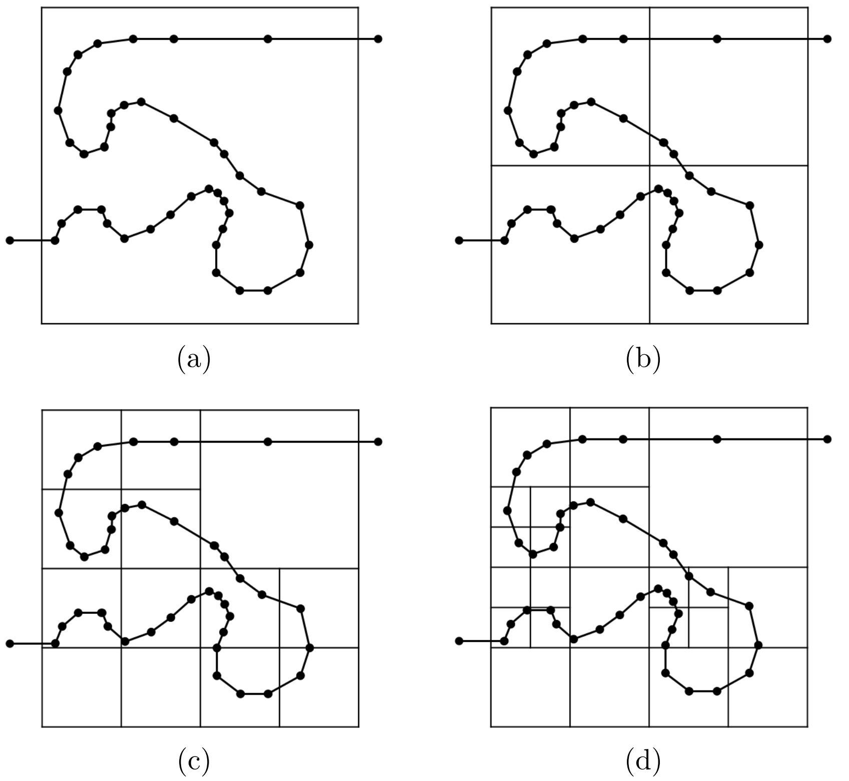


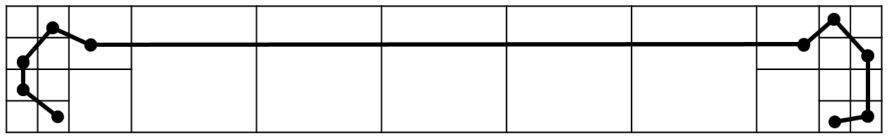
Figure 3.24 Application of the first criteria of refinement to limit to 4 the number of geometric points inside a cell.

2. **Criteria#2: Minimum Curve Level**

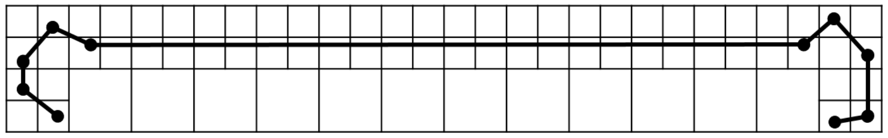
   This criterion controls the level of refinement of the cells traversed by the geometric curve. The minimum level of refinement for the cut cells *Min_Curv_Lvl* is given as input by the user. The algorithm applies this criterion during the geometry marching process presented in Section 3.4.3 to control the resolution of interface cells representing the discrete topology.

   This criterion provides an additional level of mesh resolution control to the first criterion. This feature is illustrated in Fig. 3.25, which considers the straight horizontal line for the given topology curve to be tagged. Based on the first criteria, if the user limits the number of points *Nb_PTS_Max* bounded by a cell to one, the criteria will refine only the cells that contain the geometric points to the level that satisfies that every cell bounds only one point. Nevertheless, this boundary may need more resolution for the physical interpretation of the solver's numerical solution. So, the level of refinement of the remaining interface cells needs to be controlled to achieve the required grid resolution that accurately represents the boundary. Fig. 3.25 shows the difference between applying the first criteria and a minimum curve level of 3 to refine all intersected cells to the third level.

This complementary mesh control is given by the *Min_Curv_Lvl* that is applied to each loop of the discrete topology.



(a) Application of the first Criteria to limit the number of geometric points to one inside the cells.



(b) Application of both Criteria#1, and #2 to apply a level of refinement of three to the intersected cells.

Figure 3.25 Application of refinement criteria #2.

Proximity-based criteria

1. **Criteria#3: One Cell - One Loop (OCOL)**

   The OCOL concept limits the number of loops that intercept a cell to one loop so that each intersected cell is intersected by only one loop. If the cell is cut by more than one loop, this cell is refined until it is traversed by only one loop.

   This criterion is used to refine the region of proximity to isolate the interface cells from each other by considering that each interface layer will represent its own intercepting loop.

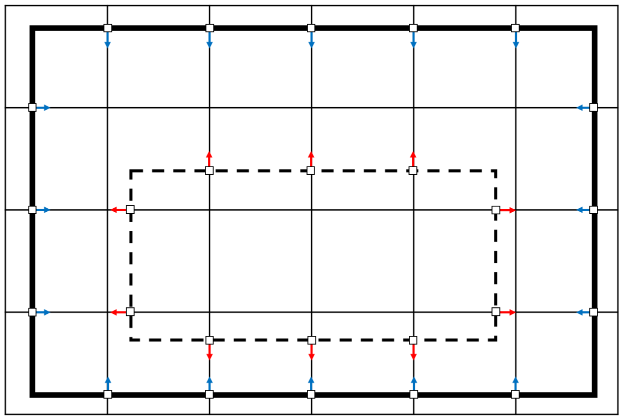
   The cut information established during the geometry marching process counts the number of cuts at each intersected cell. One counted cut has Entering/Exiting information. If the number of cuts is greater than one, the cell is cut by more than one loop, and consequently the cell is refined.

   An illustration of this criterion is shown in Fig. 3.26a. The bottom line of this grid is cut by two topology curves (an outer and inner loop); the refinement criteria #3 will apply a refinement to these cells to isolate each cell to be cut by one loop, Fig. 3.26b.

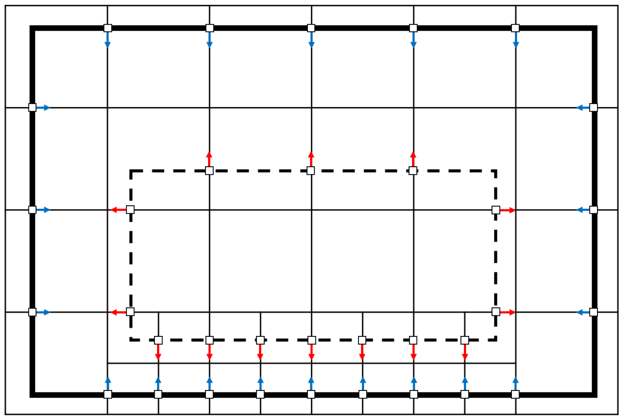
2. **Criteria#4: Refinement of curves in proximity**

   This criterion adds another complementary tool for grid management. It controls the number of layers that will be added between two neighbors' intersected cells. Moreover, this criteria allows the insertion of computational nodes between interface cells in proximity, which solvers will use to solve problems in the vicinity of two curves.

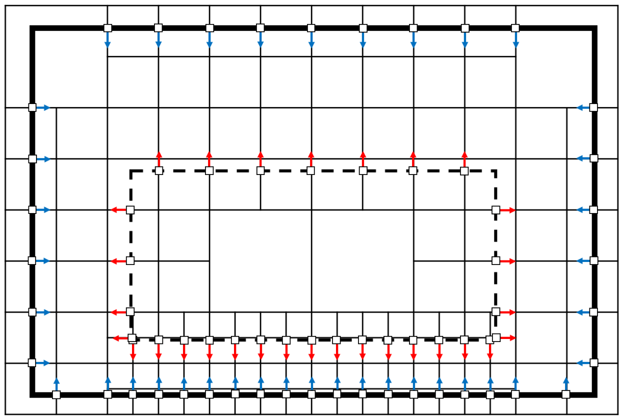
   Criteria#4 is controlled by the number of refinement cycles processed to the captured cells in proximity zones. Fig. 3.26c illustrates the application of Criteria#4 for the curves being isolated from the third criteria. The criteria applies recursive refinement to the cells in neighborhoods cut by different curves. The refinement cycles are the number of recursive refinements applied to the cells identified in proximity regions.



(a) Initial grid with cells detected as cut by two different loops.



(b) Application of Criteria#3 to limit each cell to be cut by one loop.



(c) Application of Criteria#4 to add computational points in the proximity zone (After one cycle).

Figure 3.26 Application of the proximity refinement criteria.

3.7 Grid management and control Add-ons

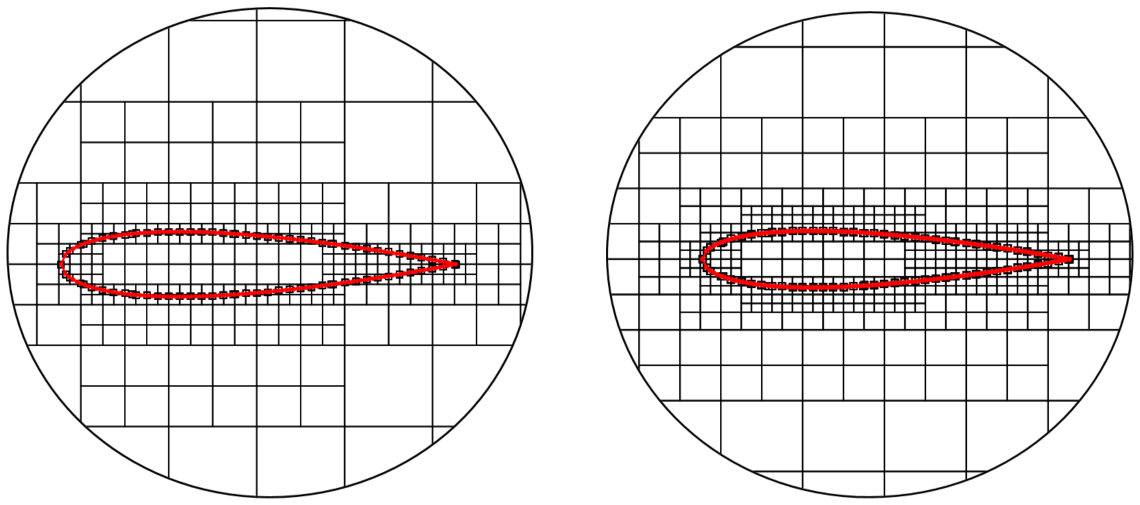
In order to have the maximum functionality possible for the meshing and tagging algorithms, additional '$Add - ons$' are implemented to provide more control on the grid generation and refinement process. Furthermore, these '$Add - ons$' are integrated with the tagging class to leverage the output from the integration of other classes (Reconstruction, Solvers).

Uniform interface level control

Since the second criterion is applied to the intersected cells, the input refinement level will be applied only to the intersected cells (The IIC 🟥 and the IOC ◎). However, the NIIC 🟨 are not subjected to those constraints so that they may have a level different from their intersected cells (usually coarser than the intersected cell).

This functionality is introduced to adjust the level of refinement of the interface-non-intersected cells so that all the interface cells at the end of the tagging process are at the same level of refinement.

This feature is presented in Fig. 3.27 for a NACA 0012 profile immersed in an hierarchical grid with five levels of refinement applied to the profile. The figure presents the results of the refined grid with and without the application of this feature. In Fig. 3.27b the levels around the profile boundaries are equal, including the NIIC.



(a) A boundary without uniform refinement.

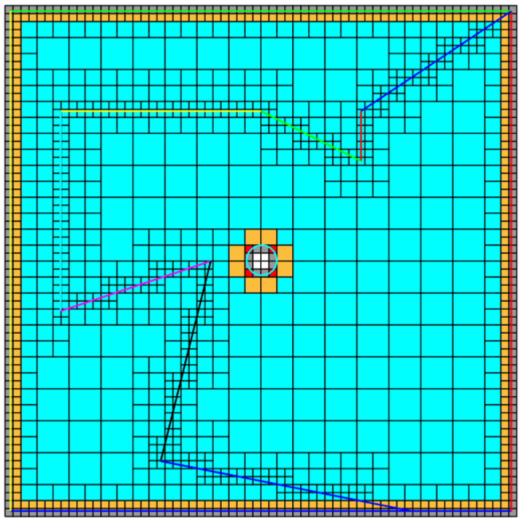
(b) The boundary with uniform refinement.

Figure 3.27 Five levels of refinement with vs. without uniform refinement around a NACA 0012 profile.

Local grid pre-adaptation

A pre-adaptation step is introduced by processing a topology file that represents the curve/volume of the target region, with the corresponding desired refinement levels. The algorithm applies the same tagging procedure; it marches the input adaptation topology and applies the required refinement according to the given adaptation topology.

An arbitrary adaptation topology is illustrated in Fig. 3.28, the generated grid results from applying both refinement criteria and pre-adaptation parameters.


(a) Pre-adaptation topology.

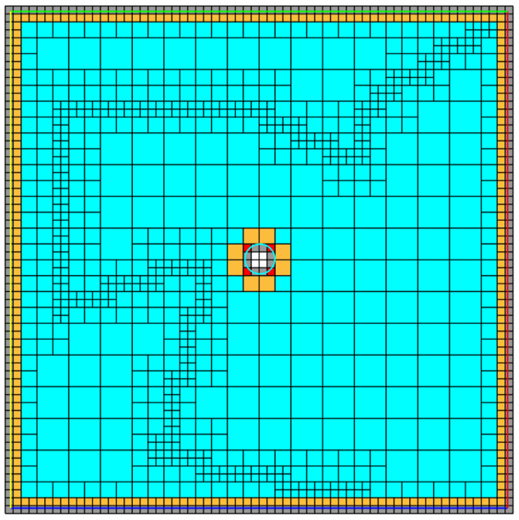
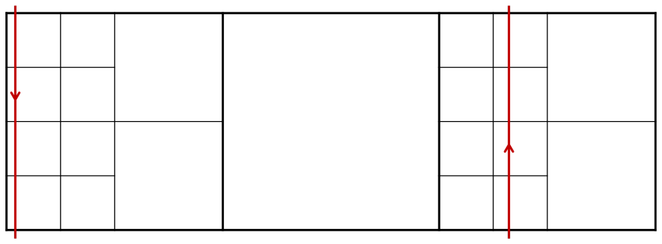

(b) Grid pre-adaptation is realized following the adaptation topology.

Figure 3.28 Grid pre-adaptation with three levels of refinement following the adaptation topology.

Regeneration of hierarchical grid topology to uniformly finer levels

This feature is motivated for grid convergence studies. Since the grid is hierarchical, a systematic refinement must be offered in all directions to calculate a given metric's convergence.

Consider a simple initial grid $N_x = 3, N_y = 1$ with a $Min\_Curv\_Lvl = 3$. In order to have a systematic refinement, doubling the initial grid resolution to $N_x = 6, N_y = 2$ will not lead to halving the whole grid elements, as illustrated in Fig. 3.29. For that case, the $RefineUniformGrid$ capability is in place to refine the whole computational domain one level deeper so that the factor between all grid elements of the two grids is always a factor of 1/2, Fig. 3.30.


(a) Initial Grid $N_x = 3$, $N_y = 1$, $Min\_Curv\_Lvl = 3$.

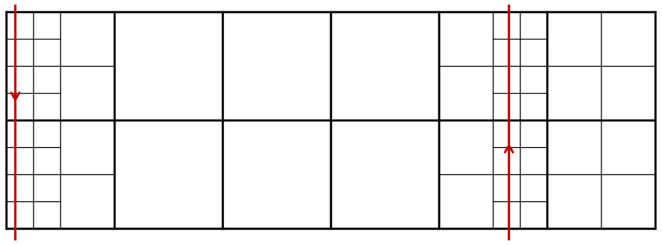
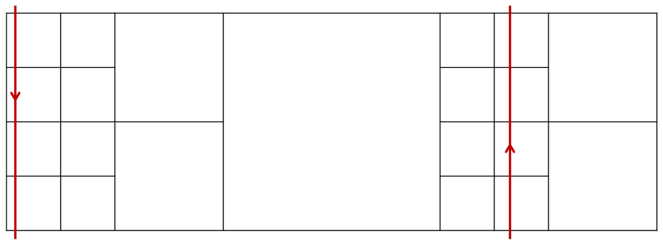

(b) Initial grid resolution doubled $N_x = 6$, $N_y = 2$, $Min\_Curv\_Lvl = 3$.

Figure 3.29 Representation of doubling the grid resolution will not lead to a grid topology finer by a factor 1/2 for all grid elements.


(a) Initial Grid $N_x = 3$, $N_y = 1$, $Min\_Curv\_Lvl = 3$.

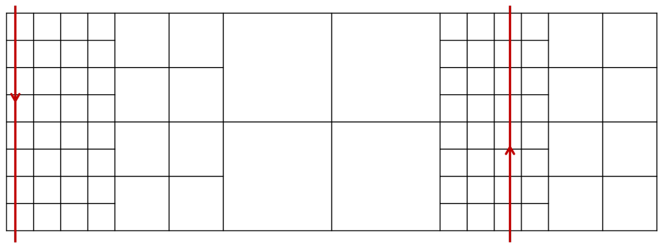

(b) $RefineUniformGrid$ reduces the initial grid elements size by a factor of 1/2.

Figure 3.30 The application of $RefineUniformGrid$ function to generate the same grid topology with $1/2^N$ factors of reduction.

CHAPTER 4 FLOW SOLVER

This chapter presents the development procedure of a flow solver for an inviscid compressible flows model for Euler's equations on overset hierarchical grids in the context of the IBM application. First, it presents the governing equations discretized with the flux difference splitting scheme (Roe's Scheme). Next, it introduces the development of the reconstruction scheme used to apply the boundary conditions on the interface cells. Then it presents different types of boundary conditions for Euler's equations, adopted in the context of this research. Finally, the integration of the ALE with the flow solver is presented to introduce the grid motion for overset grids.

4.1 Governing equations

The proposed IBM methodology is applied to the solution of $2D$ inviscid compressible flow. Four equations govern the flowfield: the mass conservation law, the momentum conservation, and the energy equation. This system of equations is represented in the conservative compact form as follows:

$$W_t + F(W)_x + G(W)_y = 0 \tag{4.1}$$

$$W = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix} \qquad F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (E+p)u \end{bmatrix} \qquad G = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (E+p)v \end{bmatrix} \tag{4.2}$$

where the flow density is noted by ρ, the static pressure by p, thermodynamic temperature T, flow velocity by $\mathbf{V}(x,y) = (u,v)$. The total energy per unit mass E, described as a function of the internal energy per unit mass e, is defined as follows:

$$E = \frac{\mathbf{V}^2}{2} + e \tag{4.3}$$

Two additional equations are required, namely, the equation of state,

$$p = \rho RT \tag{4.4}$$

which for an ideal gas gives,

$$e = c_v T = \frac{p}{\rho(\gamma - 1)} \tag{4.5}$$

where R and γ are the gas constant and the specific heat ratio of the fluid, respectively. Specific heat at constant volume c_v is given by,

$$c_v = \frac{R}{\gamma - 1}$$

4.2 Numerical scheme

Euler's equations are discretized using a cell-centered finite-volume scheme, and each control volume is represented by the cell face which in Cartesian coordinate is presented in Fig. 4.1.

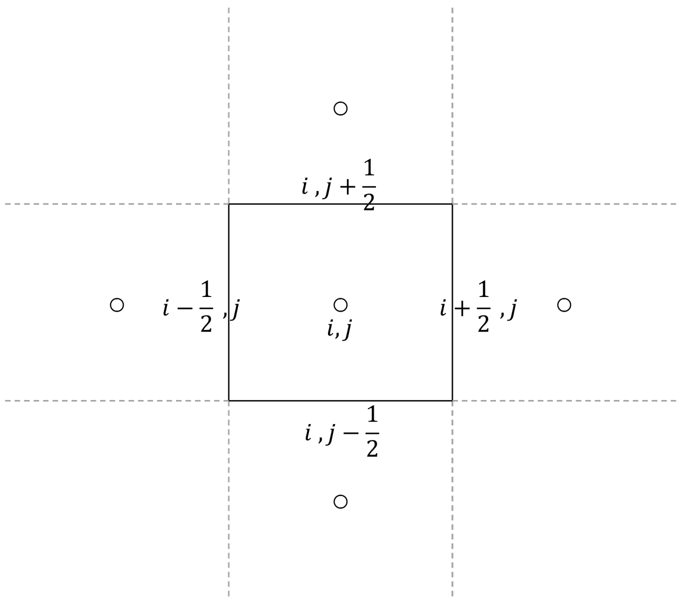


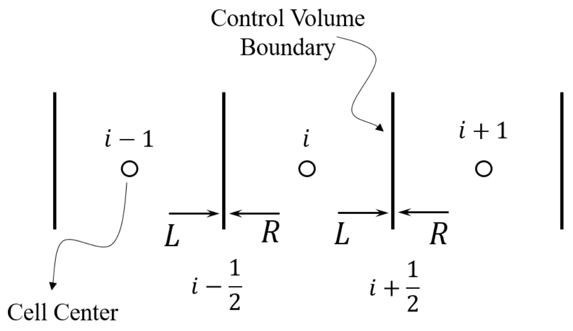
Figure 4.1 Cell-centered control volume definition in Cartesian grids.

The Euler equations are numerically cast as an Initial Boundary Value Problem (IBVP) with the explicit conservation form in the physical Cartesian frame of reference as:

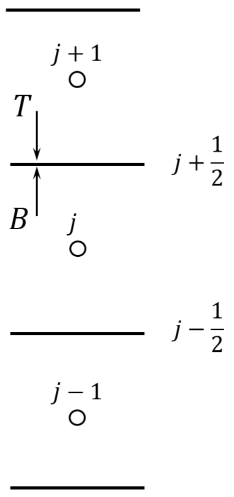
$$W_{i,j}^{n+1} = W_{i,j}^{n} + \frac{\Delta t}{\Delta x \Delta y}\left[\left(F_{i-\frac{1}{2},j}^{n} - F_{i+\frac{1}{2},j}^{n}\right)\Delta y + \left(G_{i,j-\frac{1}{2}}^{n} - G_{i,j+\frac{1}{2}}^{n}\right)\Delta x\right] \tag{4.6}$$

This hyperbolic system of equations takes the form of a Riemann problem, and the adopted resolution scheme is the approximate Roe solver. This scheme is a commonly used for solving Euler's equations for compressible flows for its capability to resolve flow discontinuities accurately [123].

Roe's scheme is based on evaluating convective fluxes F and G at the boundaries of the control volume $\left(i \pm \frac{1}{2}, j \pm \frac{1}{2}\right)$ by averaging the quantities stored in the neighboring cells, where the properties are computed. For 2-D notation, the neighbors are denoted in the left, right, top and bottom directions as L, R, T, and B, respectively, as illustrated in Fig. 4.2. For Cartesian coordinates, Roe's average properties are evaluated in x and y directions by averaging the quantities at (L, R) and (B, T) neighbors respectively. For hierarchical Roe's average properties are expressed in Eq. 4.7



(a) Left and Right states.



(b) Bottom and Top states.

Figure 4.2 Evaluation of average properties at the control volume boundaries.

$$
\begin{aligned}
\tilde{\rho} &= \sqrt{\rho_L \rho_R} \quad , \\
\tilde{u} &= \frac{u_L\sqrt{\rho_L} + u_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \quad , \\
\tilde{v} &= \frac{v_L\sqrt{\rho_L} + v_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \quad , \\
\tilde{H} &= \frac{H_L\sqrt{\rho_L} + H_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \quad , \\
\tilde{a} &= \sqrt{(\gamma - 1)(\tilde{H} - \tilde{q}^2/2)} \quad , \\
\tilde{\mathbf{V}} &= [\tilde{u} \quad \tilde{v}] \quad , \\
\tilde{q}^2 &= \tilde{u}^2 + \tilde{v}^2
\end{aligned}
\tag{4.7}
$$

Roe's fluxes at the control volume boundary $\tilde{F}$ and $\tilde{G}$ are decomposed into the sum of wave contributions,

$$\begin{aligned}\tilde{F}_{i+\frac{1}{2},j} &= \frac{1}{2}(F_{i,j} + F_{i+1,j}) - \frac{1}{2}\sum_{k_x=1}^{4} \alpha_{k_x} \mid \lambda_{k_x} \mid \overrightarrow{e}_{k_x} \\ \tilde{G}_{i,j+\frac{1}{2}} &= \frac{1}{2}(G_{i,j} + G_{i,j+1}) - \frac{1}{2}\sum_{k_y=1}^{4} \alpha_{k_y} \mid \lambda_{k_y} \mid \overrightarrow{e}_{k_y}\end{aligned} \tag{4.8}$$

where λ_k and $\overrightarrow{e}_k$ are the eigen values and the corresponding eigen vectors, such that:

$$\lambda_{k_x} = \begin{bmatrix} \tilde{u}-\tilde{a} \\ \tilde{u} \\ \tilde{u} \\ \tilde{u}-\tilde{a} \end{bmatrix} \quad , \quad \lambda_{k_y} = \begin{bmatrix} \tilde{v}-\tilde{a} \\ \tilde{v} \\ \tilde{v} \\ \tilde{v}-\tilde{a} \end{bmatrix} \quad , \quad k = 1,2,3,4. \tag{4.9}$$

$$\overrightarrow{e}_{1_x} = \begin{bmatrix} 1 \\ \tilde{u}-\tilde{a} \\ \tilde{v} \\ \tilde{h}-\tilde{u}\tilde{a} \end{bmatrix} \quad , \quad \overrightarrow{e}_{2_x} = \begin{bmatrix} 1 \\ \tilde{u} \\ \tilde{v} \\ \frac{1}{2}\tilde{q}^2 \end{bmatrix} \quad , \quad \overrightarrow{e}_{3_x} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \tilde{v} \end{bmatrix} \quad , \quad \overrightarrow{e}_{4_x} = \begin{bmatrix} 1 \\ \tilde{u}+\tilde{a} \\ \tilde{v} \\ \tilde{h}+\tilde{u}\tilde{a} \end{bmatrix} \tag{4.10}$$

$$\overrightarrow{e}_{1_y} = \begin{bmatrix} 1 \\ \tilde{u} \\ \tilde{v}-\tilde{a} \\ \tilde{h}-\tilde{v}\tilde{a} \end{bmatrix} \quad , \quad \overrightarrow{e}_{2_y} = \begin{bmatrix} 1 \\ \tilde{u} \\ \tilde{v} \\ \frac{1}{2}\tilde{q}^2 \end{bmatrix} \quad , \quad \overrightarrow{e}_{3_y} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ -\tilde{u} \end{bmatrix} \quad , \quad \overrightarrow{e}_{4_y} = \begin{bmatrix} 1 \\ \tilde{u} \\ \tilde{v}+\tilde{a} \\ \tilde{h}+\tilde{v}\tilde{a} \end{bmatrix} \tag{4.11}$$

$$\alpha_{k_x} = \begin{bmatrix} \frac{\Delta p_x - \tilde{\rho}\tilde{a}\Delta u_x}{2\tilde{a}^2} \\ \Delta\rho_{-}\frac{\Delta p_x}{\tilde{a}^2} \\ \tilde{\rho}\Delta v_x \\ \frac{\Delta p_x + \tilde{\rho}\tilde{a}\Delta u_x}{2\tilde{a}^2} \end{bmatrix} \quad , \quad \alpha_{k_y} = \begin{bmatrix} \frac{\Delta p_y - \tilde{\rho}\tilde{a}\Delta v_y}{2\tilde{a}^2} \\ \Delta\rho - \frac{\Delta p_y}{\tilde{a}^2} \\ -\tilde{\rho}\Delta u_x \\ \frac{\Delta p_x + \tilde{\rho}\tilde{a}\Delta v_x}{2\tilde{a}^2} \end{bmatrix} \quad , \quad k = 1,2,3,4. \tag{4.12}$$

where, the properties variations across the cells are calculated as:

$$\begin{aligned}
\Delta p_x &= p_L - p_R \quad , \quad & \Delta p_y &= p_B - p_T \\
\Delta \rho_x &= \rho_L - \rho_R \quad , \quad & \Delta \rho_y &= \rho_B - \rho_T \\
\Delta u_x &= u_L - u_R \quad , \quad & \Delta u_y &= u_B - u_T \\
\Delta v_x &= v_L - v_R \quad , \quad & \Delta v_y &= v_B - v_T
\end{aligned} \tag{4.13}$$

The finite volume discrete form of the Roe's scheme in Cartesian coordinates with averaged fluxes, given by Eqs. 4.8 is written as:

$$W_{i,j}^{n+1} = W_{i,j}^{n} - \frac{\Delta t}{\Delta x \Delta y}\left[\left(\tilde{F}_{i+\frac{1}{2},j} - \tilde{F}_{i-\frac{1}{2},j}\right)\Delta y + \left(\tilde{G}_{i,j+\frac{1}{2}} - \tilde{G}_{i,j-\frac{1}{2}}\right)\Delta x\right] \tag{4.14}$$

For hierarchical grid application, the finite volume discrete form of Roe's scheme presented in Eq. 4.14 will be written in general form as follows:

$$\begin{aligned}
W_{i,j}^{n+1} = W_{i,j}^{n} &- \frac{\Delta t}{\Delta x \Delta y}\left(\sum_{k=1}^{n_E} \tilde{F}_{i+\frac{1}{2},j_k}\Delta y_k - \sum_{k=1}^{n_W} \tilde{F}_{i-\frac{1}{2},j_k}\Delta y_k\right) \\
&+ \frac{\Delta t}{\Delta x \Delta y}\left(\sum_{k=1}^{n_N} \tilde{G}_{i_k,j+\frac{1}{2}}\Delta x_k - \sum_{k=1}^{n_S} \tilde{G}_{i_k,j-\frac{1}{2}}\Delta x_k\right)
\end{aligned} \tag{4.15}$$

where n_E, n_W, n_N, and n_S are the number of neighbors in cardinal directions $EWNS$ respectively. The discrete form represented in Eq. 4.15 relies on integrating the fluxes over the control volume, by summing the contribution of every neighbor's flux (Roe's average flux) in cardinal directions. An illustration of the hierarchical flux representation is shown in Fig. 4.3.

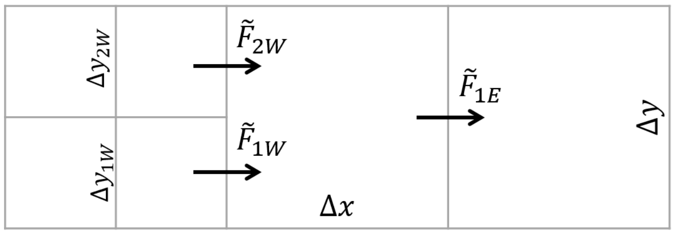


Figure 4.3 Flux representation in hierarchical grids in x-direction.

Time step calculation

The temporal integration being an explicit type, and in order to ensure the stability of the numerical scheme, the time step calculation is constrained by the CFL condition. For 1D, the CFL is presented as:

$$CFL = (a+ \mid u \mid)\frac{\Delta t}{\Delta x} \leq 1$$

To ensure a stable numerical scheme, the time step Δt is calculated by respecting the CFL condition,

$$\Delta t = min\left(\frac{CFL\Delta x}{a+ \mid u \mid}\right)$$

4.3 Reconstruction scheme

To impose boundary conditions on a physical immersed boundary, the essence of the IBM is the reconstruction of the imposed boundary to the interface cells. This consists in the calculation of flow properties at the interface cells that represent the physical boundary. In addition, this reconstruction approach is adopted for transferring the information between overset grids. Generally speaking, BCs usually falls into three types:

- Dirichlet : $\phi = D(x)$
- Neumann : $\partial\phi/\partial n = N(x)$
- Robin : $\partial\phi/\partial n = \phi R(x)$

where $D(x)$, $N(x)$ and $R(x)$ are functions that describe the boundary condition, and n is the normal to the boundary. And x is the curvilinear coordinate along the boundary.

For a given grid, the layer of interface cells is the list of targeted cells to be reconstructed because of its proximity to the boundary. These cells do not have a complete stencil, so the information of the imposed boundary condition is transmitted to these cells by the reconstruction scheme.

The reconstruction approach is applicable to two instances:

- Interface Reconstruction: This case is applied to the cells that replace the presence of a physical boundary containing boundary condition information.

- Overset Reconstruction: This case concerns the reconstruction used to transfer information between the two overset grids.

4.3.1 Reconstruction methodology

The methodology proposed to reconstruct the unknown values at the centers of the interface cells is based on the information at Face Cell (FC) centers and the intersection points associated with Interface Cell (IC), where the boundary condition is imposed. For a given cell P, the reconstruction stencil is based on finding the available information for reconstruction in the four cardinal directions (WESN) as illustrated in Fig. 4.4.

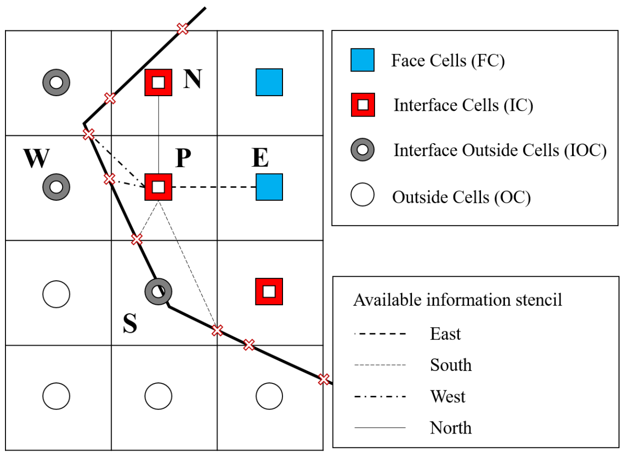


Figure 4.4 Reconstruction stencil of interface cell P.

The reconstruction polynomial ϕ is a linear function defined at the cell center of each interface cell that represents the reconstructed physical quantity.

$$\phi_P = a + bx + cy \tag{4.16}$$

This polynomial has three unknown coefficients a, b and c such that,

$$b = \frac{\partial \phi_P}{\partial x} \qquad , \qquad c = \frac{\partial \phi_P}{\partial y} \tag{4.17}$$

To describe this polynomial in terms of the reconstruction stencil, it is written as,

$$\phi_i = a + b(x_i - x_P) + c(y_i - y_P) \qquad , \qquad i = W, E, S, N \tag{4.18}$$

Where ϕ_i is the reconstructed physical quantity, and i is the index of any arbitrary point in the cardinal directions.

For a Dirichlet BC ϕ_{in} applied at point E_{in} , as illustrated in Fig. 4.5, Eq. 4.18 is given as,

$$\phi_{in} = a + b(x_{E_{in}} - x_P) + c(y_{E_{in}} - y_P) \tag{4.19}$$

Similarly, for a Neumann BC $\frac{\partial \phi_{out}}{\partial n}$ applied at point E_{out}, the BC is expressed using Eq. 4.17 as,

$$\begin{aligned}\frac{\partial \phi_{out}}{\partial n} &= \frac{\partial \phi_{out}}{\partial x} \cdot n_x + \frac{\partial \phi_{out}}{\partial y} \cdot n_y \\ \frac{\partial \phi_{out}}{\partial n} &= b \cdot n_x + c \cdot n_y\end{aligned} \tag{4.20}$$

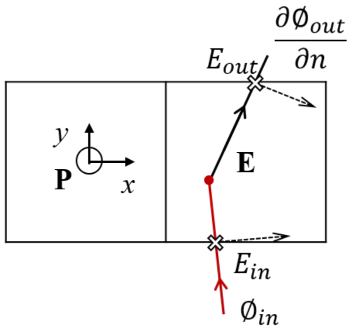


Figure 4.5 Reconstruction of Dirichlet and Neumann BC's of cell P from East side.

The reconstruction polynomial is evaluated in the four cardinal directions (WESN), expressed in Eq. 4.18 for face cells and using Eq. 4.19 and/or Eq. 4.20 according to the type of applied BC (*i.e.* Dirichlet or Neumann) as follows,

- Evaluate the BC polynomial of the applied BC $f(\phi)$ at the center of the reconstructed cell. This BC is evaluated at the intersection points of the discrete topology and the cell's boundaries. This information is delivered by the tagging procedure.
- Develop a local least-square system of equations that represents the linear polynomial. The system consists of four equations representing the cardinal directions with three unknown a, b, and c, representing the polynomial coefficients. As a result, the least-square system of reconstruction is always of fixed dimensions 4×3. A method of averaging will be presented to keep the system dimension within the dimensions mentioned above.

- Evaluate the least-square weight coefficients of each influencing neighbor.
- Solve an implicit system for all interface cells using the weighting coefficients that satisfy the imposed BC.

Fig. 4.6 illustrates a case that includes both Dirichlet and Neumann BC. The neighbors in each of the cardinal directions are identified to reconstruct the information in cell P, surrounded by FC at West and North sides, IOC at East and south directions. A Neumann BC K_ϕ is applied at segment L_1, and a Dirichlet BC is applied at segment L_2. The values of these BCs are applied at the intersection points (S_{in}, S_{out}, E_{in}, E_{out}), where the normal vectors $(n_{S_{in}}, n_{S_{out}}, n_{E_{in}}, n_{E_{out}})$ are defined.

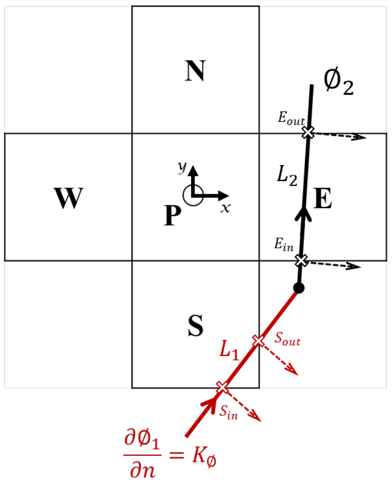


Figure 4.6 Dirichlet and Neumann BC reconstruction for cell P.

The evaluation of the BC polynomial in the four cardinal directions is written as:

$$\text{West:}\quad f(\phi)_W = a + b(x_W - x_P) + c(y_W - y_P) = \phi_W$$

$$\text{East:}\quad \begin{aligned} f(\phi)_{E_{in}} &= a + b(x_{E_{in}} - x_P) + c(y_{E_{in}} - y_P) = \phi_2(x_{E_{in}}, y_{E_{in}}) \\ f(\phi)_{E_{out}} &= a + b(x_{E_{out}} - x_P) + c(y_{E_{out}} - y_P) = \phi_2(x_{E_{out}}, y_{E_{out}}) \end{aligned}$$

$$\text{South:}\quad \begin{aligned} f(\phi)_{S_{in}} &= b \cdot n_{x,S_{in}} + c \cdot n_{y,E_{in}} = \frac{\partial \phi_{S_{in}}}{\partial n} = K_{\Phi_{in}} \\ f(\phi)_{S_{out}} &= b \cdot n_{x,S_{out}} + c \cdot n_{y,E_{out}} = \frac{\partial \phi_{S_{out}}}{\partial n} = K_{\Phi_{out}} \end{aligned}$$

$$\text{North:}\quad f(\phi)_N = a + b(x_N - x_P) + c(y_N - y_P) = \phi_N$$

In order to keep the system represented by four equations and three unknowns (a, b, c) 4×3, an averaging procedure [124] is conducted at the sides with multiple BC, as at East and South directions, such that:

$$f(\phi)_E = \frac{f(\phi_{E_{in}}) + f(\phi_{E_{out}})}{2}$$
$$f(\phi)_S = \frac{f(\phi_{S_{in}}) + f(\phi_{S_{out}})}{2}$$

So that the four cardinal equations are substituted by :

$$\text{West:}\quad f(\phi)_W = a + b(x_W - x_P) + c(y_W - y_P) = \phi_W$$

$$\text{East:}\quad f(\phi)_E = a + b\left[\tfrac{x_{E_{in}}+x_{E_{out}}}{2} - x_P\right] + c\left[\tfrac{y_{E_{in}}+y_{E_{out}}}{2} - y_P\right] = \tfrac{\phi_2(x_{E_{in}},y_{E_{in}})+\phi_2(x_{E_{out}},y_{E_{out}})}{2} = \phi_{2_{avg}}$$

$$\text{South:}\quad f(\phi)_S = b\left[\tfrac{n_{x,S_{in}}+n_{x,S_{out}}}{2}\right] + c\left[\tfrac{n_{x,S_{in}}+n_{y,S_{out}}}{2}\right] = \tfrac{K_{\Phi_{in}}+K_{\Phi_{out}}}{2} = K_{\phi_{avg}}$$

$$\text{North:}\quad f(\phi)_N = a + b(x_N - x_P) + c(y_N - y_P) = \phi_N$$

This system of equation can be represented in the form:

$$A\mathbf{x} = b$$

$$\begin{bmatrix} 1 & x_W - x_P & y_W - y_P \\ 1 & \frac{x_{E_{in}}+x_{E_{out}}}{2} - x_P & (\frac{y_{E_{in}}+y_{E_{out}}}{2} - x_P) \\ 0 & \frac{n_{x,S_{in}}+n_{x,S_{out}}}{2} & \frac{n_{y,S_{in}}+n_{y,S_{out}}}{2} \\ 0 & x_N - x_P & y_N - y_P \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \phi_W \\ \phi_{2_{avg}} \\ K_{\phi_{avg}} \\ \phi_N \end{bmatrix} \tag{4.21}$$

The general form of the local system of reconstruction is represented by :

$$\begin{bmatrix} A_{aW} & A_{bW} & A_{cW} \\ A_{aE} & A_{bE} & A_{cE} \\ A_{aS} & A_{bS} & A_{cS} \\ A_{aN} & A_{bN} & A_{cN} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} b_W \\ b_E \\ b_S \\ b_N \end{bmatrix} \tag{4.22}$$

Using the least-square formulation [78] to construct the least square reconstruction matrix

A_R, such that:

$$
\begin{gathered}
(A^T A)\mathbf{x} = A^T b \\
\mathbf{x} = (A^T A)^{-1} A^T b \\
\mathbf{x} = A_R b
\end{gathered}
$$

where A_R is evaluated as:

$$A_R = (A^T A)^{-1} A^T \tag{4.23}$$

So that the solution of the least-square system is found as:

$$\mathbf{x} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} A_{RaW} & A_{RaE} & A_{RaS} & A_{RaN} \\ A_{RbW} & A_{RbE} & A_{RbS} & A_{RbN} \\ A_{RcW} & A_{RcE} & A_{RcS} & A_{RcN} \end{bmatrix} \begin{bmatrix} b_W \\ b_E \\ b_S \\ b_N \end{bmatrix} \tag{4.24}$$

The above equation contains the local reconstruction coefficients representing the cardinal neighbors' contribution in the reconstructed value ϕ_P. Each column in the reconstruction matrix A_R denoted in Eq. 4.24 represents the neighbor's contribution corresponding to the indicated cardinal direction to the reconstructed value. Each row represents the linear combination of the four cardinal directions to impart the polynomial coefficients (a, b, c). So that;

$$
\begin{aligned}
a &= A_{RaW} b_W + A_{RaE} b_E + A_{RaS} b_S + A_{RaN} b_N \\
b &= A_{RbW} b_W + A_{RbE} b_E + A_{RbS} b_S + A_{RbN} b_N \\
c &= A_{RcW} b_W + A_{RcE} b_E + A_{RcS} b_S + A_{RcN} b_N
\end{aligned}
\tag{4.25}
$$

Since the reconstruction polynomial in Eq 4.16 is defined at the center of the reconstructed cell, the reconstructed value at the cell center is given by,

$$\phi_P = a = A_{RaW} b_W + A_{RaE} b_E + A_{RaS} b_S + A_{RaN} b_N \tag{4.26}$$

which gives the reconstruction formula at the cell center of a given arbitrary interface cell as a function of its four Cardinal neighbors values.

The last step in the presented methodology is to assemble a matrix of unknowns to reconstruct all the interface cells in one system. In such a way, the elements of the reconstruction matrix A_R represented in Eq. 4.26 will be placed in a global reconstruction system, each element being placed in its relative position, according to its order in the list of interface cells.

The following example demonstrates a simple implicit reconstruction system for clarification of the proposed methodology, without relying on a physical topology or tagging. Considering the given grid in Fig. 4.7, Cells $(7, 8, 9)$ are assumed to be interface cells with unknown values(ϕ_7, ϕ_8, ϕ_9) and should be reconstructed implicitly using the known values of the remaining cells $(1 : 5, 6, 10, 11 : 15)$. The reconstruction coefficients A_R of the interface cells are assumed to be evaluated using Eq. 4.23.

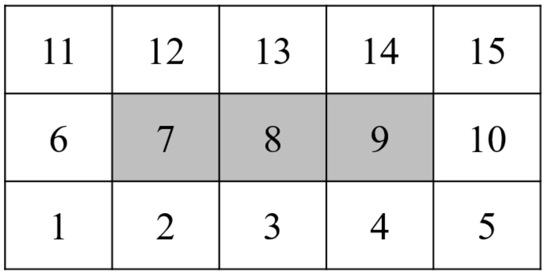


Figure 4.7 Illustration to implicitly reconstruct the gray cells (7,8 and 9).

The reconstructed values at the interface cells are formulated using Eq. 4.26;

$$\begin{aligned} \phi_7 &= A_{Ra6}\phi_6 + A_{Ra8}\phi_8 + A_{Ra2}\phi_2 + A_{Ra12}\phi_{12} \\ \phi_8 &= A_{Ra7}\phi_7 + A_{Ra9}\phi_9 + A_{Ra3}\phi_3 + A_{Ra13}\phi_{13} \\ \phi_9 &= A_{Ra8}\phi_8 + A_{Ra10}\phi_{10} + A_{Ra4}\phi_4 + A_{Ra14}\phi_{14} \end{aligned} \tag{4.27}$$

Rearranging Eq. 4.27 to move all the unknowns to the left side of the equation, such that:

$$\begin{aligned} \phi_7 - A_{Ra8}\phi_8 &= A_{Ra6}\phi_6 + A_{Ra2}\phi_2 + A_{Ra12}\phi_{12} \\ -A_{Ra7}\phi_7 + \phi_8 - A_{Ra9}\phi_9 &= A_{Ra3}\phi_3 + A_{Ra13}\phi_{13} \\ -A_{Ra8}\phi_8 + \phi_9 &= A_{Ra10}\phi_{10} + A_{Ra4}\phi_4 + A_{Ra14}\phi_{14} \end{aligned} \tag{4.28}$$

The solution of the implicit system of Eq. 4.28 leads to assemble the matrix of implicit reconstruction as follows:

$$\begin{bmatrix} 1 & -A_{Ra8} & 0 \\ -A_{Ra7} & 1 & -A_{Ra9} \\ 0 & -A_{Ra8} & 1 \end{bmatrix} \begin{bmatrix} \phi_7 \\ \phi_8 \\ \phi_9 \end{bmatrix} = \begin{bmatrix} A_{Ra6}\phi_6 + A_{Ra2}\phi_2 + A_{Ra12}\phi_{12} \\ A_{Ra3}\phi_3 + A_{Ra13}\phi_{13} \\ A_{Ra10}\phi_{10} + A_{Ra4}\phi_4 + A_{Ra14}\phi_{14} \end{bmatrix} \tag{4.29}$$

The assembled matrix is a square matrix of the size of number of the reconstructed interface cells $N_{IC} \times N_{IC}$, and is viewed generally in the following form,

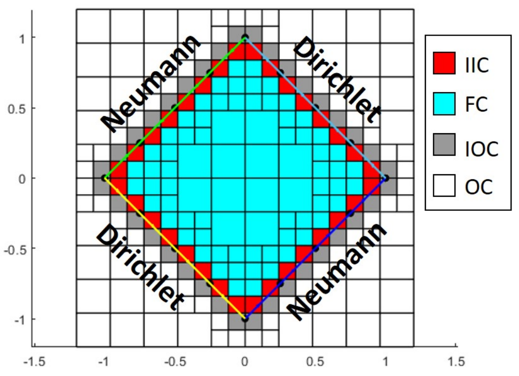
$$A_{Ri}[N_{IC} \times N_{IC}] \cdot \phi_i[N_{IC} \times 1] = \sum BC_i[N_{IC} \times 1]$$

4.3.2 Reconstruction verification

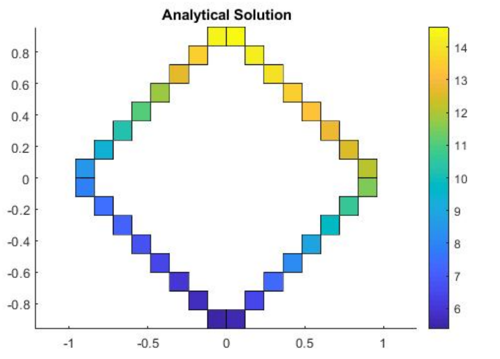
A test is performed to verify the proposed reconstruction methodology's accuracy with Neumann and Dirichlet BCs. The test consists of imposing an analytical solution to the computational domain illustrated in Fig. 4.8, following the analytical representation of Eq. 4.30.

$$\begin{aligned} \phi(x,y) &= 2x + 5y + 10 \\ \frac{\partial \phi}{\partial n} &= \frac{\partial \phi}{\partial x} n_x + \frac{\partial \phi}{\partial y} n_y \end{aligned} \tag{4.30}$$

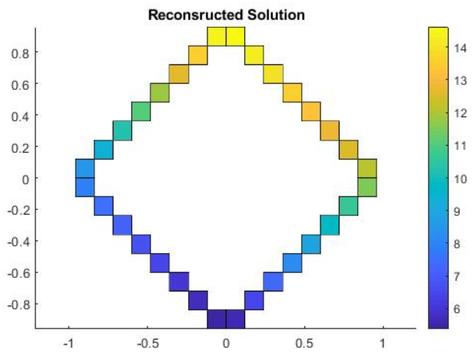
Since the reconstruction polynomial of Eq. 4.16 is linear, as the imposed analytical function, it is expected that the reconstructed values will be exact. A Dirichlet BC is imposed at two opposite boundaries of the domain, as shown in Fig. 4.8a, and a Neumann BC is imposed at the two other opposite boundaries. All internal nodes FC are imposed with the analytical values evaluated by Eq. 4.30, and all interface cells are reconstructed implicitly. The resulting reconstructed values of the interface cells are found to match the exact analytical values, with an average error (L_2 Norm) of $3.563e^{-15}$.



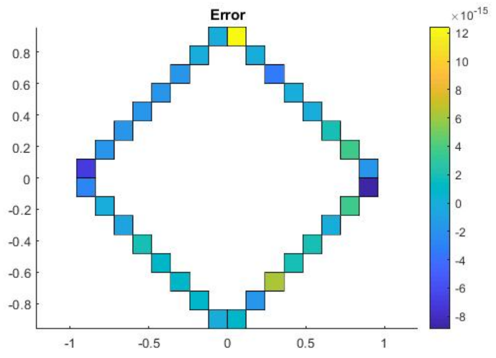
(a) Computational grid with BC.



(b) Analytical function represented on interface cells.



(c) Reconstructed solution on interface cells.



(d) Reconstruction error.

Figure 4.8 Verification of reconstruction scheme using analytical function.

4.4 Boundary conditions for Euler equations

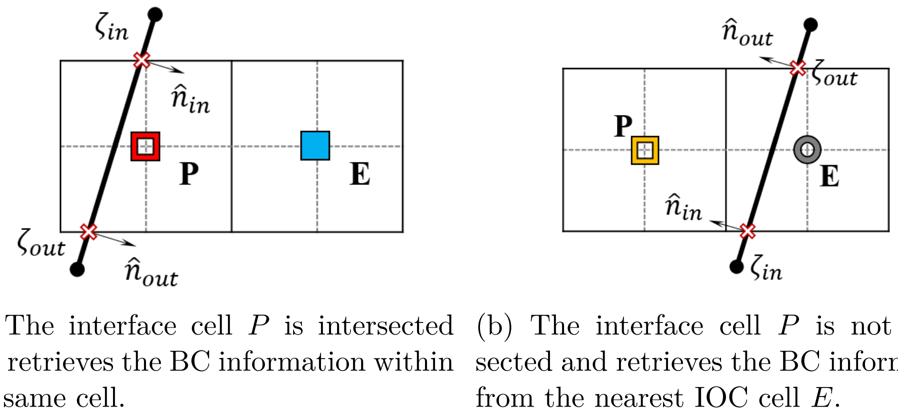
The boundary conditions for Euler equations are directly imposed at the intersection points between the discrete topology and the intersected cells. These points are identified during the tagging operation as presented in Section 3.4.3. Taking advantage of these points allows imposing the BC at a physical point that lies on the topology without approximating the position of the BC.

The imposed BC is used to reconstruct the interface cells values via the implicit approach presented in Section 4.3. The position of the stored values of the BC relative to the interface cell leads to two possible configurations:

- The interface cell is an intersected IIC: in this case, the reconstruction of the cell P

retrieves the BC information stored in the same cell, as illustrated in Fig. 4.9a

- The interface cell is not an intersected cell NIIC: in that case, the reconstruction employs the BC information stored in the nearest IOC, as shown in Fig. 4.9b, point P retrieves the BC information from the nearest IOC cell E.



(a) The interface cell P is intersected and retrieves the BC information within the same cell.

(b) The interface cell P is not intersected and retrieves the BC information from the nearest IOC cell E.

Figure 4.9 Possible configurations for BC reconstruction.

For Euler's equations, the primitive variables $U = [\rho, u, v, p]$ are reconstructed in each interface cell (IIC and NIIC) according to the BC type at the intersection points. The different types of implemented BC are as follows:

Supersonic Inlet

For a supersonic inlet BC, the free-stream values are applied at the inlet. When at least one intersection point associated with an IIC or NIIC interface cell is indicated as supersonic inlet BC, then all flow variables are imposed at the interface cell center. The pressure at the inlet p_{in}, the Mach number M_{in}, the temperature T_{in} and the flow angle θ_{in}. The imposed values are sufficient to evaluate all other flow quantities as:

$$\begin{aligned} a_{in} &= \sqrt{\gamma R T_{in}} \\ V_{in} &= M_{in} a_{in} \\ \rho_{in} &= \tfrac{p_{in}}{R T_{in}} \\ u_{in} &= V_{in} cos(\theta_{in}) \\ v_{in} &= V_{in} sin(\theta_{in}) \end{aligned} \tag{4.31}$$

Supersonic Outlet

The flow variables at a supersonic outlet are reconstructed from the fluid side using the Neumann null condition ($d\phi/dn = 0$), implying no gradients at the supersonic outlet boundary.

Subsonic Inlet

For subsonic inlet BC, the total pressure P_0 and total temperature T_0 are imposed. The pressure is evaluated at the interface cell centers of the subsonic inlet by applying a Neumann null condition on pressure ($dp_{in}/dn = 0$). The reconstructed pressure at cell centers is then used to evaluate the pressure values at the intersection points at the subsonic inlet. This evaluation is performed using the reconstruction polynomial defined in Eq. 4.16. The pressure values at the intersection points are then employed to evaluate the remaining flow properties (ρ, u, v) at the boundary using isentropic flow relations. The properties are then treated as Dirichlet BC in the reconstruction scheme. The treatment of subsonic BC is detailed in Algorithm 4.

Algorithm 4 Treatment of subsonic BC

```
1: Impose $P_0$,$T_0$, $\gamma$, $\theta_{in}$.
2: Impose $\frac{dp_{in}}{dn} = 0$ at the boundary.
3: Reconstruct the pressure at the interface cell centers using the imposed Neumann condition.
4: Evaluate the pressure $P_{in}$ at the boundary using Eq. 4.16.
5: Using isentropic flow relations, evaluate at the boundary:
6:     $T_{in}$ using $\frac{P_0}{p_{in}} = \left(\frac{T_0}{T_{in}}\right)^{\gamma/(\gamma-1)}$
7:     $\rho_{in} = p_{in}/RT_{in}$
8:     $a_{in} = \sqrt{\gamma RT_{in}}$
9:     $M_{in}$ using $\frac{P_0}{p_{in}} = (1 + 0.5(\gamma - 1)M_{in}^2)^{\gamma/(\gamma-1)}$
10:    $V_{in} = M_{in}a_{in}$
11:    $u_{in} = V_{in}cos(\theta_{in})$, $v_{in} = V_{in}sin(\theta_{in})$
12: Reconstruct $\rho$, $u$, $v$ in the interface cells using the values in the above steps 6 : 11 as Dirichlet BC.
```

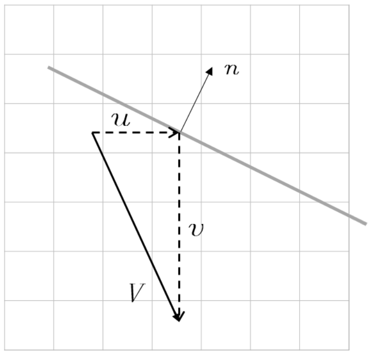
Subsonic Outlet

For the subsonic outlet BC, the pressure at the outlet boundary is imposed such that $p_{out}/P_0 < 1$ and treated as Dirichlet BC. The remaining flow properties (ρ, u,v) are reconstructed by imposing a Neumann null BC at the outlet boundary ($\frac{d\rho}{dn} = \frac{du}{dn} = \frac{dv}{dn} = 0$).

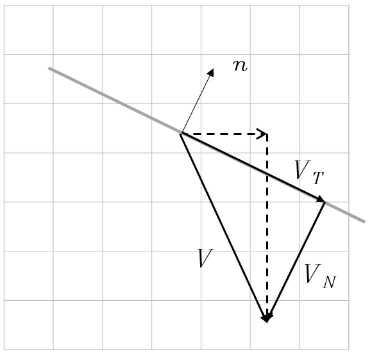
Slip Wall

A slip wall BC application is realized by subtracting the normal vector component V_N from the reconstructed velocity vector V. This reconstructed velocity V is obtained by applying a Neumann BC ($dV/dn = 0$). The result is a projection into the boundary's tangential direction, resulting in V_T. This procedure is illustrated in Fig. 4.32, where the reconstructed velocity is denoted by V, projected to the normal and tangential direction to the boundary, and denoted by V_N and V_T respectively. The projected tangential component V_T is then decomposed to the Cartesian coordinates (u_T, v_T).

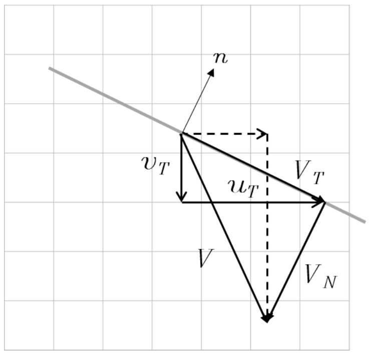
$$\begin{aligned} V_T &= V - V_N \\ V_T &= V - (V \cdot n)n \\ \begin{bmatrix} u_T \\ v_T \end{bmatrix} &= \begin{bmatrix} u \\ v \end{bmatrix} - (u \cdot n_x + v \cdot n_y) \begin{bmatrix} n_x \\ n_y \end{bmatrix} \end{aligned} \tag{4.32}$$



(a) The reconstructed velocity vector V.



(b) The reconstructed velocity V is decomposed into the tangential V_T and normal V_N components to the boundary.



(c) The reconstructed velocity V is projected to the tangential component V_T.

Figure 4.10 Slip boundary condition application by vector projection.

Overset Interface

This type of BC is imposed on the set of interpolated cells handled by the overset algorithm. These cells are responsible for transferring the information between overlapping grids. The overset algorithm always ensures well-conditioned interpolated cells, *i.e.*, an interpolated cell is always interpolated through a set of internal nodes on the underlying/overlying grid. This conditioning is previously demonstrated in Section 3.5.

A 2-D illustration of the overset reconstruction methodology is illustrated in Fig. 4.11. The interpolated cell on grid #2 is mapped on grid #1 to the coordinate $\xi(x, y)$. This mapped coordinate is the reference point where each interpolated cell's reconstruction polynomial is defined, following Eq. 4.16. Since the hypothesis of the well-conditioned interpolated cells is taken into account, then the mapped point ξ is always surrounded by internal nodes (P, N, S, W, E).

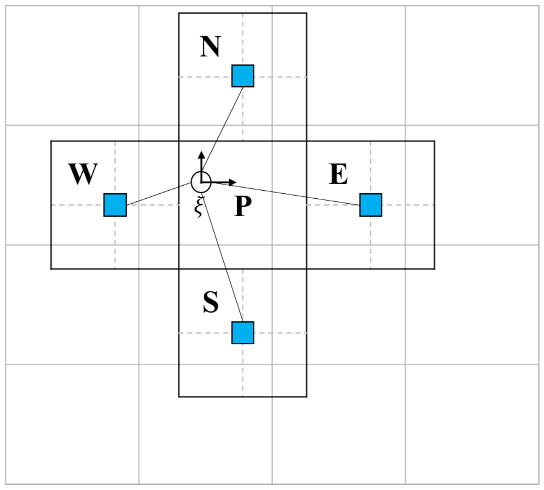


Figure 4.11 Reconstruction stencil on a 2-D overset grid.

For the illustrated case in Fig. 4.11, the bounding cell on grid #1 of the mapped point ξ is the cell P. The reconstruction stencil at the mapped point ξ employs the value stored at the bounding cell P, and the contribution of the derivatives from the neighbours (N, S, W, E), such that:

$$\phi_\xi = \phi_P + b(x_\xi - x_P) + c(y_\xi - y_P) \tag{4.33}$$

where the coefficients b and c are evaluated following the same approach of the implicit reconstruction scheme for a polynomial defined at the point ξ, and it is found using Eq. 4.25, or it can be written as:

$$\begin{array}{c} E \\ W \\ N \\ S \end{array} \begin{bmatrix} 1 & x_\xi - x_E & y_\xi - y_E \\ 1 & x_\xi - x_W & y_\xi - y_W \\ 1 & x_\xi - x_N & y_\xi - y_N \\ 1 & x_\xi - x_S & y_\xi - y_S \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \phi_E \\ \phi_W \\ \phi_N \\ \phi_S \end{bmatrix} \tag{4.34}$$

The overset reconstruction employs the values at the cell centers of the overlapped cells so that the overset reconstruction type is imposed as Dirichlet BC's on the reconstruction system.

Boundary conditions summary

Table 4.1 Reconstruction types of boundary conditions

Boundary Condition	Density	Velocity	Pressure
Supersonic Inlet	Dirichlet	Dirichlet	Dirichlet
Supersonic Outlet	Neumann=0	Neumann=0	Neumann=0
Subsonic Inlet	Dirichlet	Dirichlet	Neumann=0
Subsonic Outlet	Neumann=0	Neumann=0	Dirichlet
Slip Wall	Neumann=0	Neumann=0	Neumann=0
Overset Interface	Dirichlet	Dirichlet	Dirichlet

4.5 Treatment of overset grid motion

The ALE approach supports the concept of grid motion. This formulation, in link with the overset approach, simplifies the implementation of moving geometries, especially the case with rigid boundaries.

The motion of rigid boundary with the overset grid implies that the Geometric Conservation Laws (GCL) are always conserved, with no nodes deformation for the adopted approach.

The introduction of a grid velocity V_g modifies the governing equations of flow field in Eq. 4.2 to take into account the grid velocity for the calculation of the convective terms, such that:

$$V_g = u_g\hat{i} + v_g\hat{j} \tag{4.35}$$

$$F = \begin{bmatrix} \rho(u-u_g) \\ \rho u(u-u_g)+p \\ \rho v(u-u_g) \\ (E+p)(u-u_g)+u_g p \end{bmatrix} \qquad G = \begin{bmatrix} \rho(v-v_g) \\ \rho u(v-v_g) \\ \rho v(v-v_g)+p \\ (E+p)(v-v_g)+v_g p \end{bmatrix} \tag{4.36}$$

This grid motion modifies also the eigen-values λ_i of Roe's scheme, and can be written as below,

$$\lambda_{k_x} = \begin{bmatrix} \tilde{u} - u_g - \tilde{a} \\ \tilde{u} - u_g \\ \tilde{u} - u_g \\ \tilde{u} - u_g - \tilde{a} \end{bmatrix} \quad , \quad \lambda_{k_y} = \begin{bmatrix} \tilde{v} - v_g - \tilde{a} \\ \tilde{v} - v_g \\ \tilde{v} - v_g \\ \tilde{v} - v_g - \tilde{a} \end{bmatrix} \quad , \quad k = 1, 2, 3, 4. \tag{4.37}$$

The corresponding eigen-vectors e_i and the wave strength α_i remains unchanged. Another modification to account for the presence of the grid motion is the slip wall BC, for the relative velocity of the moving boundary. The tangential velocity component is written as:

$$\begin{aligned} V_T &= V - ((V - V_g) \cdot n)n \\ \begin{bmatrix} u_T \\ v_T \end{bmatrix} &= \begin{bmatrix} u \\ v \end{bmatrix} - ((u - u_g) \cdot n_x + (v - v_g) \cdot n_y) \begin{bmatrix} n_x \\ n_y \end{bmatrix} \end{aligned} \tag{4.38}$$

Finally, the time step calculation is also modified to account for the grid velocity.

$$\Delta t_x = min \left(\frac{CFL\Delta x}{a + \mid u - u_g \mid} \right)$$

$$\Delta t_y = min \left(\frac{CFL\Delta y}{a + \mid v - v_g \mid} \right)$$

$$\Delta t = min(\Delta t_x, \Delta t_y)$$

CHAPTER 5 VERIFICATION AND VALIDATION OF THE ALGORITHM

This chapter presents the implementation of the algorithm, and a set of conducted test cases with the goal of validate and verify the developed code.

5.1 Algorithm implementation

The IBM code is developed in an object-oriented architecture, where each building block is encapsulated into a class. The UML scheme of the developed code is presented in Appendix A, where five main classes are essentially developed, namely, (`DiscreteTopology`, `HierarchicalGrid`, `ComputationalGrid`, `OversetGrid`, `Reconstruction`, and `FlowSolver`).

The programming environment for this code is *Matlab*. Architecture classes and all visualisations employed *Matlab* for execution. The usage of this tool facilitates the programming and prototyping, as this research is the first version of the proposed algorithm. This interpreted-based environment puts a limitation on the computational performance, and as such it may not be the suitable choice for test cases with a very large number of cells.

5.2 General aspects

For most of test cases, unless otherwise stated, the upcoming generalization is applied.

- A perfect gas model is adopted with a specific heat ratio γ of 1.4, and a gas constant R of 287.05 $J\ Kg^{-1}\ K^{-1}$.
- The order of accuracy is calculated by the evaluation of L_2 norm of the error ε_{L_2} versus a mean grid resolution $\tilde{h}$ as follows:

$$\varepsilon_{L_2} = \sqrt{\frac{1}{\Omega}\sum_{i=1}^{N} \omega_i \mid \phi_{Ref} - \phi_{Num} \mid^2} \tag{5.1}$$

$$\tilde{h} = \sqrt{\frac{\sum_{i=1}^{N} \omega_i}{N}} \tag{5.2}$$

where :

  - ω_i : Cell area.
  - N : Number of cells of the computational domain.

- Ω : Total area of the domain.
- ϕ_{Ref} : Reference solution of a given physical property.
- ϕ_{Num} : Numerical solution of a given physical property.

5.3 Test cases classification

The test cases are arranged into four groups, categorized by the type of computational grid (single or overset grid) and the type of flow-field (steady or unsteady). This classification is listed in Table 5.1, and the test cases listing is illustrated are Tables 5.2-5.5

Table 5.1 Classification of test cases

Category	Tests Type
1	Steady flows on a single grid
2	Unsteady flows on a single grid
3	Steady flows on a overset grids
4	Unsteady flows on a overset grids

Table 5.2 Test cases - Category#1: Steady flows on a single grid

Test ID	Test	Objective
CAT11	Subsonic flow in a tube	Test basic solver consistency and test inlet and outlet subsonic boundary conditions
CAT12	Supersonic flow in a tube	Test basic solver consistency and test inlet and outlet supersonic boundary conditions
CAT13	Subsonic Rayleigh flow	Test source terms handling to subsonic flow
CAT14	Supersonic Rayleigh flow	Test source terms handling to supersonic flow
CAT15	Subsonic radial flow	Test basic 2D solution
CAT16	Confluence of two supersonic flows	Test shock capturing capabilities and slip line solution in 2D
CAT17	Supersonic flow over a bump	Test transonic case on a curved boundary

Table 5.3 Test cases - Category#2: Unsteady flows on a single grid

Test ID	Test	Objective
CAT21	Shock tube flow	Test the scheme accuracy for unsteady flows
CAT22	Tube in impulsive motion	Verify grid motion treatment and ALE implementation
CAT23	Unsteady supersonic flow over a symmetric wedge	Reference case to test grid motion

Table 5.4 Test cases - Category#3: Steady flows on overset grids

Test ID	Test	Objective
CAT31	Subsonic tube flow with stationary overset grid	Test the treatment of interpolated cells in two directions
CAT32	Subsonic tube flow with moving overset grid	Verify grid motion treatment and ALE implementation

Table 5.5 Test cases - Category#4: Unsteady flows on overset grids

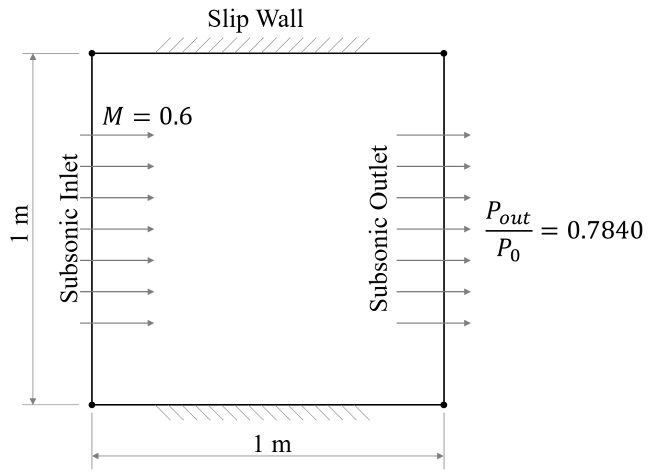
Test ID	Test	Objective
CAT41	Unsteady supersonic flow over symmetric wedge with stationary overset grid	Reference case for unsteady flow with perturbations crossing the overset interface
CAT42	Unsteady supersonic flow over symmetric wedge with moving overset grid	Reference case for testing a single grid with uniform motion
CAT43	Unsteady supersonic flow over three accelerated wedges with moving grids	Reference case for testing multiple grids with accelerated motion.

5.4 Steady flows on single grids

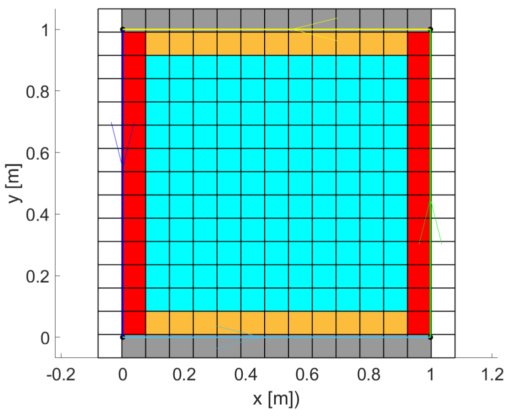
This category of tests aims to verify the developed algorithm to solve steady flows on single grids, and to demonstrate the ability to manage both single and overset grids by the same tool. These tests span subsonic/supersonic flows, shock wave interactions of multi-flows and interaction with solid boundaries.

5.4.1 Subsonic flow in tube (CAT11)

This basic test case aims to verify the implementation of the algorithm classes, by solving a constant subsonic flow inside a tube to test the implementation of subsonic inlet/outlet boundary conditions. The tube length and diameters are of one meter, the test setup is illustrated in Fig. 5.1a. The types of boundary conditions of the computational domain are Subsonic Inlet at $M = 0.6$, Subsonic Outlet and Slip Wall for the horizontal sides. This test is conducted on a coarse grid of 15x15, shown in Fig. 5.1b



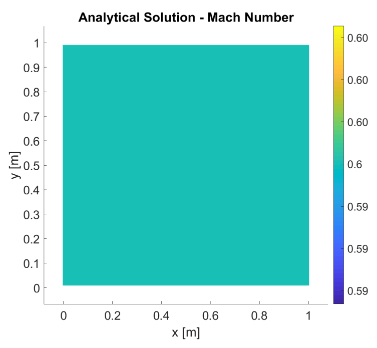
(a) Subsonic tube operating boundary conditions.



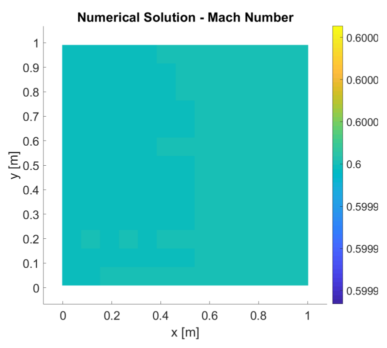
(b) Immersed boundary on a Cartesian grid of 15x15 with colored tagged cells.

Figure 5.1 Subsonic tube test setup and computational domain.

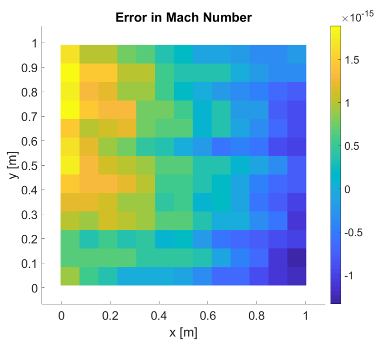
The flow enters the domain horizontally at zero incidence angle, the flow-field is initialized by the values that corresponds to the farfield conditions. Since the flow is parallel to the walls, it is expected that the solution remains unchanged with the iterations and equal to the imposed initial conditions, as shown in Figs. 5.2 and 5.3. In these figures the numerical solution of the Mach number and pressure distributions inside the tube are illustrated. The error in Mach number and pressure value are in the order of $10e^{-15}$ and $10e^{-11}$ respectively, implying that the numerical solution is almost exact.



(a) Analytical Mach number.

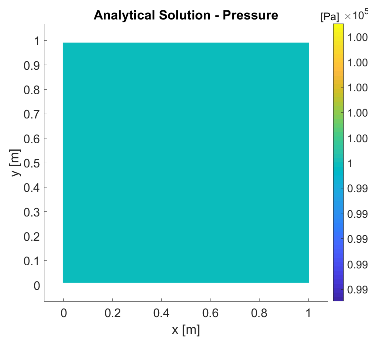


(b) Numerical Mach number.

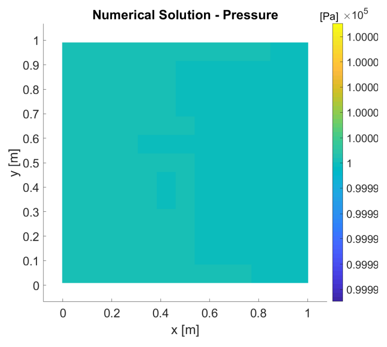


(c) Mach number error.

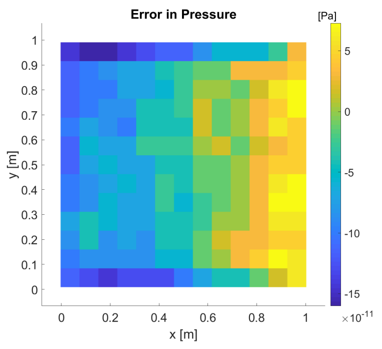
Figure 5.2 Comparison of numerical solution of Mach number for the subsonic flow in a tube.



(a) Analytical pressure distribution.



(b) Numerical pressure distribution.



(c) Pressure error.

Figure 5.3 Comparison of numerical solution of pressure for the subsonic flow in a tube.

5.4.2 Supersonic flow in tube (CAT12)

This test verifies the solution of a constant supersonic flow inside a constant diameter tube and examines the implementation of the supersonic inlet/outlet boundary conditions. The immersed geometry is the same as test (CAT11) in Section 5.4.1, a tube length and diameter of one meter. The imposed boundary conditions are given in Fig. 5.4, for a supersonic inlet Mach number of two is given, with zero incidence angle, the pressure is set to $150e3\ Pa$ and the temperature of $400\ K$ is imposed. The test is performed on a grid of 15x15, as in Fig. 5.1b to verify the capability to achieve a solution close to the exact solution of a uniform supersonic flow.

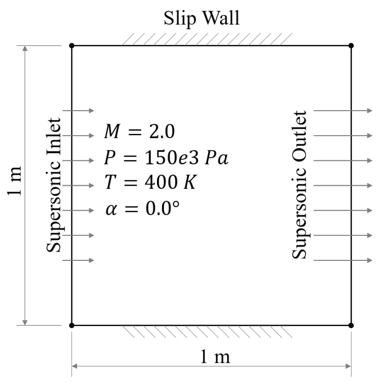
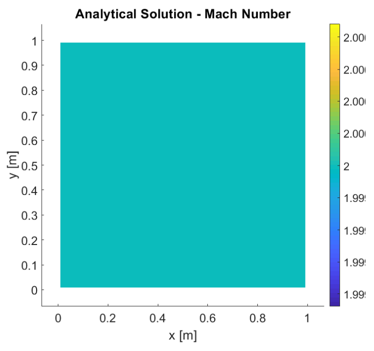
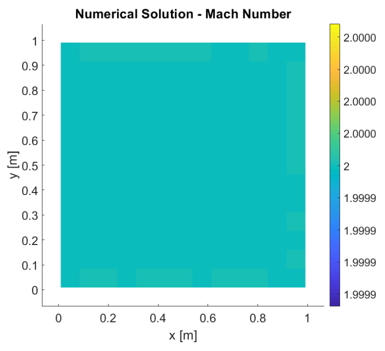


Figure 5.4 Supersonic tube test setup.

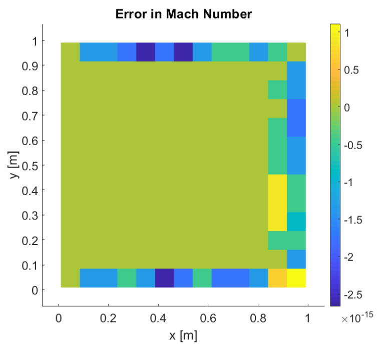
The flow-field is initialized from the inlet conditions, the solution converged from the first iteration, and achieved an error order close to the machine zero, as illustrated in Figs. 5.5 and 5.6.



(a) Analytical Mach number.

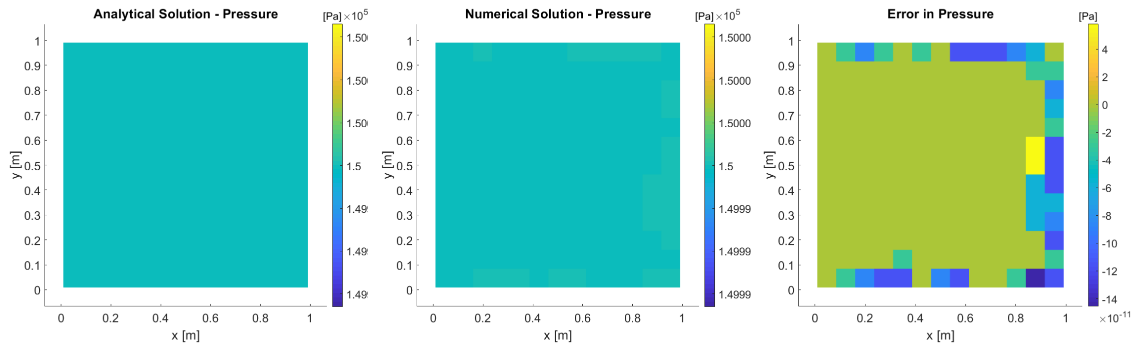


(b) Numerical Mach number.



(c) Mach number error.

Figure 5.5 Comparison of numerical solution of Mach number for the supersonic flow in a tube.



(a) Analytical Pressure. (b) Numerical Pressure. (c) Pressure error.

Figure 5.6 Comparison of numerical solution of Pressure for the supersonic flow in a tube.

5.4.3 Subsonic Rayleigh flow (CAT13)

This test is to verify the ability to solve Euler's equations with source terms in a subsonic regime. Rayleigh flow is an inviscid confined flow between two walls with heat addition $\dot{Q}$. Euler's equations presented in Eq. 4.1 becomes;

$$W_t + F(W)_x + G(W)_y = S \qquad , \qquad S = [0\ 0\ 0\ \dot{Q}] \tag{5.3}$$

where S is the source term vector. The addition of heat to a flow increases its stagnation enthalpy, which in turn increases its total temperature T_0. This drives the subsonic flow to accelerate while it is heated. The analytical solution of a Rayleigh flow and its governing equations can be found in reference [125].

Four grids were tested $(10 \times 10,\ 20 \times 20,\ 40 \times 40,\ \text{and}\ 80 \times 80)$ with the given boundary conditions in Fig. 5.7.

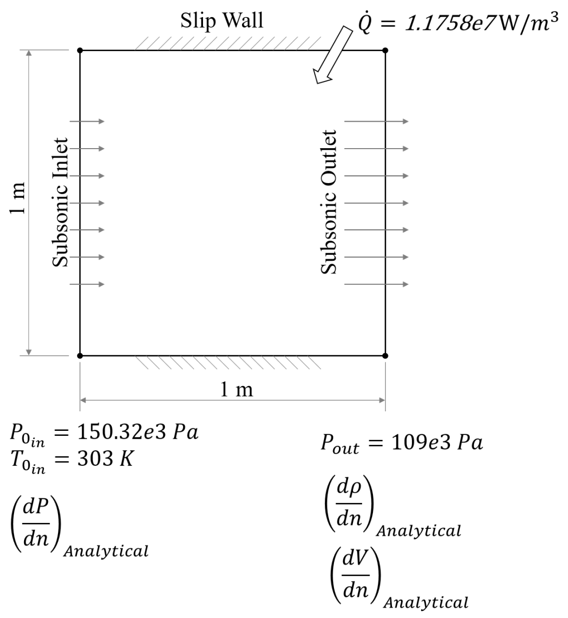

Slip Wall
$\dot{Q} = 1.1758e7 W/m^3$
Subsonic Inlet
Subsonic Outlet
1 m
1 m
$P_{0_{in}} = 150.32e3\ Pa$
$T_{0_{in}} = 303\ K$
$\left(\frac{dP}{dn}\right)_{Analytical}$
$P_{out} = 109e3\ Pa$
$\left(\frac{d\rho}{dn}\right)_{Analytical}$
$\left(\frac{dV}{dn}\right)_{Analytical}$

Figure 5.7 Subsonic Rayleigh flow test setup and operating boundary conditions.

The analytical solution of Rayleigh flow demonstrates that the flow quantities' variation along the tube is not linear, following Eq. 5.4, where the pressure varies with the Mach square. This implies that the imposition of a Neumann null condition at the inlet and the outlet will not describe the analytical solution exactly.

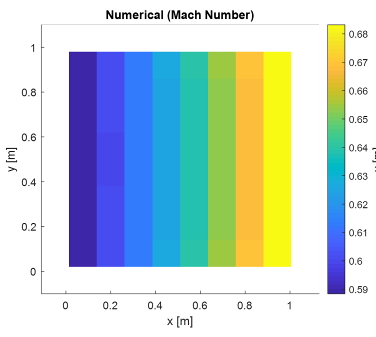
To accurately approach the analytical solution, the analytical gradient is calculated and applied as Neumann boundary condition. At the exit, the gradients of density and velocities ($\frac{d\rho}{dn}$ and $\frac{dV}{dn}$) are evaluated analytically, where at the inlet the analytical pressure gradient $\frac{dP}{dn}$ is applied. The analytical gradients are obtained by evaluating the total temperature gradient $\frac{dT_0}{dn}$ from the expression for the energy equation within a unit control volume, with boundaries 1 and 2, as expressed in Eqs. 5.5.

$$\frac{p}{p^*} = \frac{\gamma + 1}{1 + \gamma M^2} \tag{5.4}$$

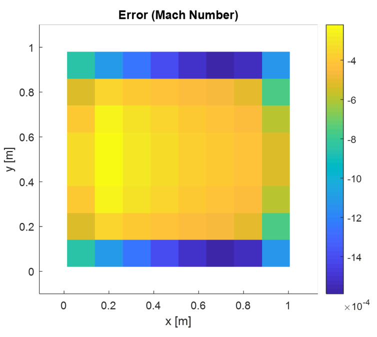
$$\begin{aligned} \dot{Q} &= \dot{m} C_p (T_{02} - T_{01}) \\ q &= C_p (T_{02} - T_{01}) \\ q &= C_p dT_0 \\ \frac{dT_0}{dn} &= \frac{q}{C_p} \end{aligned} \tag{5.5}$$

The expression of dT_0/dn in Eq. 5.5 is valid for the evaluation of all other gradients following the governing equations of Rayleigh flow.

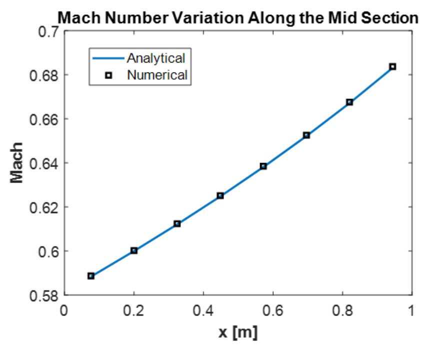
The solution shown in Figs. 5.8 and 5.9 illustrate the numerical solution of Mach number on the coarser (10×10) and finer (80×80) grids. Both grids were capable to capture the property variation along the midsection with accuracy.



(a) Numerical Mach number.

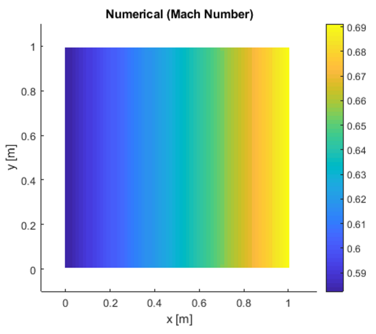


(b) Mach number error.

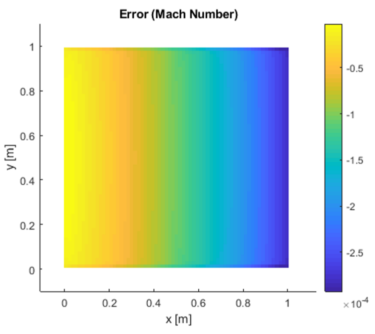


(c) Mach number variation along the mid-section.

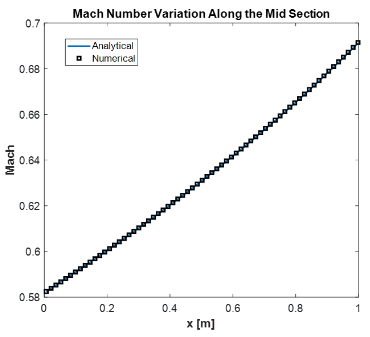
Figure 5.8 Mach number variation for subsonic Rayleigh flow on a 10x10 grid.



(a) Numerical Mach number.



(b) Mach number error.



(c) Mach number variation along the mid-section.

Figure 5.9 Mach number variation for subsonic Rayleigh flow on a 80x80 grid.

The error on the coarser grid at the mid section is bounded by (0.0368%-0.0792%). The order of convergence for the entire domain is calculated for the L_2 norm of Mach error, and is evaluated by 0.886, as shown in Fig. 5.10. The achieved order of convergence is lower than 1 due to the imposition of the analytical gradient. This may slow down the convergence of the solution, but this degradation of the order of convergence does not impact the accuracy since the order of the error is low, even with the coarser grid. A slight difference in the solution near the wall was noticed; it might be related to the implicit reconstruction; this assumption should be precised with further investigation in future work.

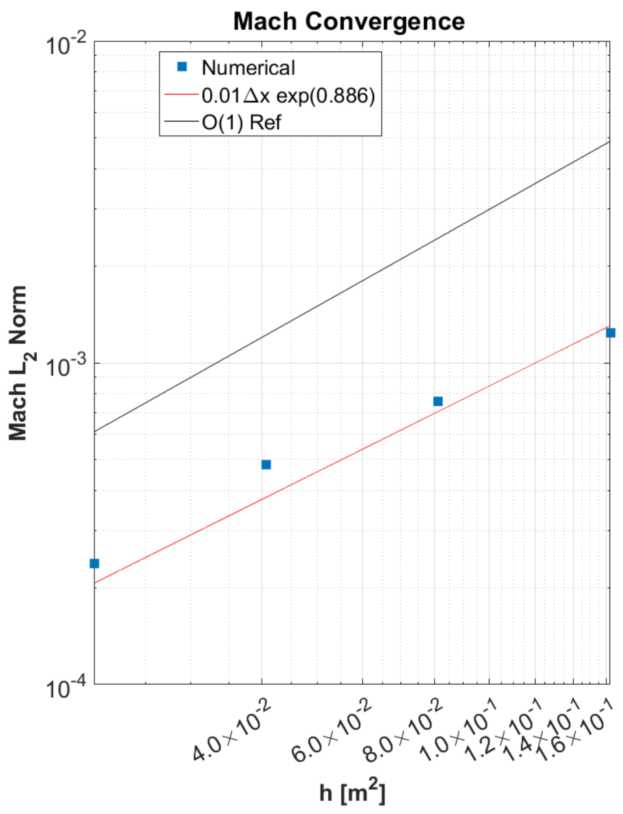


Figure 5.10 Mach number L_2 norm convergence for subsonic Rayleigh flow.

5.4.4 Supersonic Rayleigh flow (CAT14)

This test is an extension of the previous one (CAT13), and aims to evaluate the Rayleigh flow in supersonic regime. The setup is illustrated in Fig. 5.11. All flow properties are imposed at the inlet and the outlet is set as a Supersonic Outlet with the imposed values of the analytical gradients of the Rayleigh solution. As in the subsonic Rayleigh flow case, the imposition of analytical gradients at the outlet is applied because the condition of null Neumann is not sufficient to describe the analytical solution. A better choice would be the application of "Free" boundary condition, but this requires a higher order reconstruction (quadratic polynomial as example) to have enough information to reconstruct the gradients from the numerical solution.

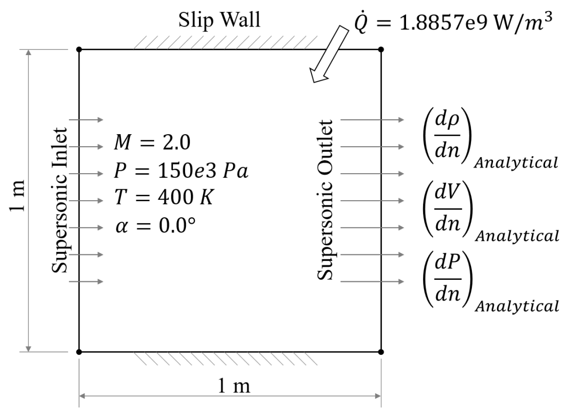
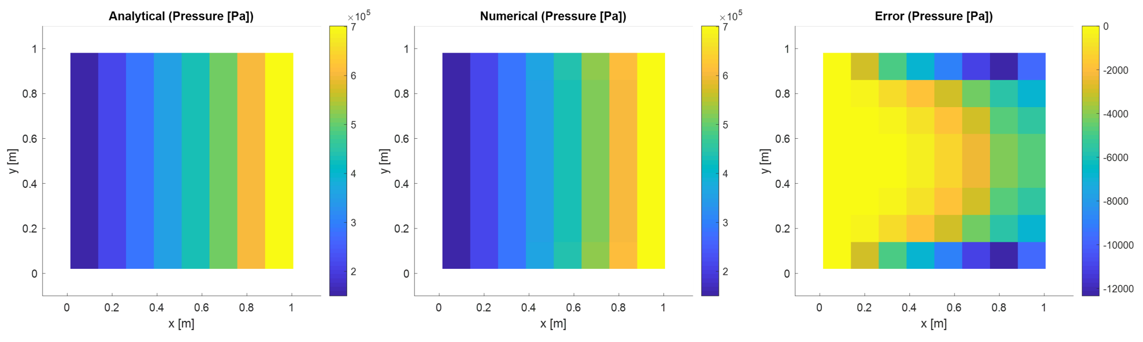

Slip Wall
$\dot{Q}$ = 1.8857e9 W/m^3
Supersonic Inlet
M = 2.0
P = 150e3 Pa
T = 400 K
α = 0.0°
Supersonic Outlet
$\left(\frac{d\rho}{dn}\right)_{Analytical}$
$\left(\frac{dV}{dn}\right)_{Analytical}$
$\left(\frac{dP}{dn}\right)_{Analytical}$
1 m
1 m

Figure 5.11 Test case setup for supersonic Rayleigh flow.

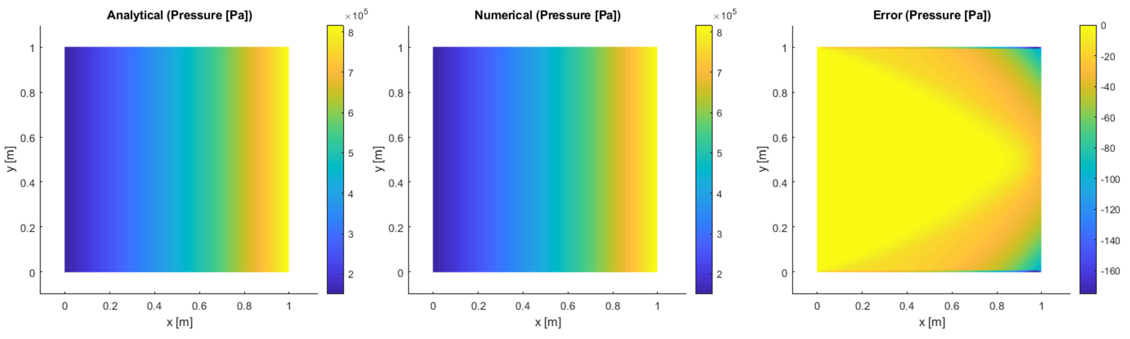
Five grids were tested ($10 \times 10,\ 20 \times 20,\ 40 \times 40,\ 80 \times 80$ and $\ 160 \times 160$) to evaluate the order of convergence and to assess the accuracy of the numerical solution.

The maximum error observed on the coarser grid in Fig. 5.12c deviates by 1.7% of the analytical solution, while on the finer grid, Fig. 5.13c, this error is 0.02%.



(a) Analytical Pressure. (b) Numerical Pressure. (c) Pressure error.

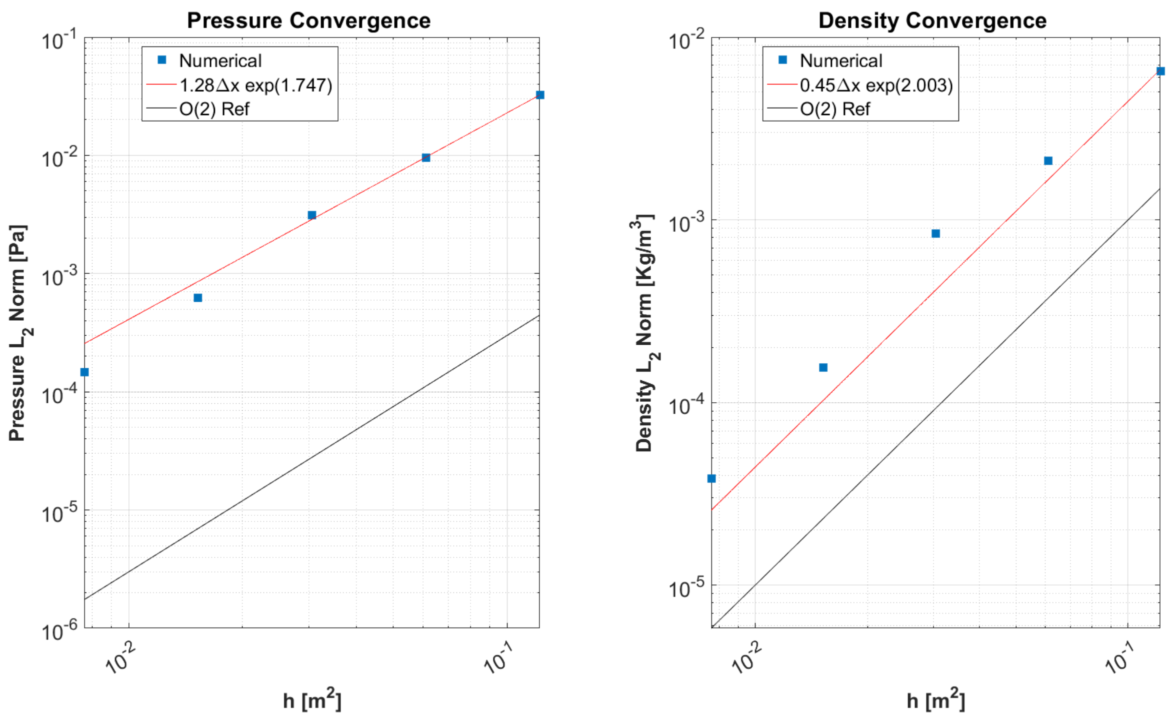
Figure 5.12 Supersonic Rayleigh flow solution on a 10x10 grid.



(a) Analytical Pressure. (b) Numerical Pressure. (c) Pressure error.

Figure 5.13 Supersonic Rayleigh flow solution on a 160x160 grid.

The imposition of analytical gradient at the outlet introduces a small upstream disturbance that converges toward the analytical solution, respecting a second order convergence as illustrated in Fig. 5.14. The shown order of accuracy is an average order that may be refined to reach an order close to 2 with finer grids.



(a) Pressure. (b) Density.

Figure 5.14 L_2 norm convergence for supersonic Rayleigh flow.

5.4.5 Subsonic radial flow (CAT15)

This test case is to verify a 2D subsonic solution, as well as the subsonic inlet/outlet reconstruction normal to a curved boundary. The geometry consists of a 2D quarter annular disk, with outer radius R_1 of 2 m and inner radius R_2 of 1 m. The flow enters from the outer radius and leaves the domain from the inner radius radially, as illustrated in Fig. 5.15. The vertical and horizontal boundaries are set as slip walls.

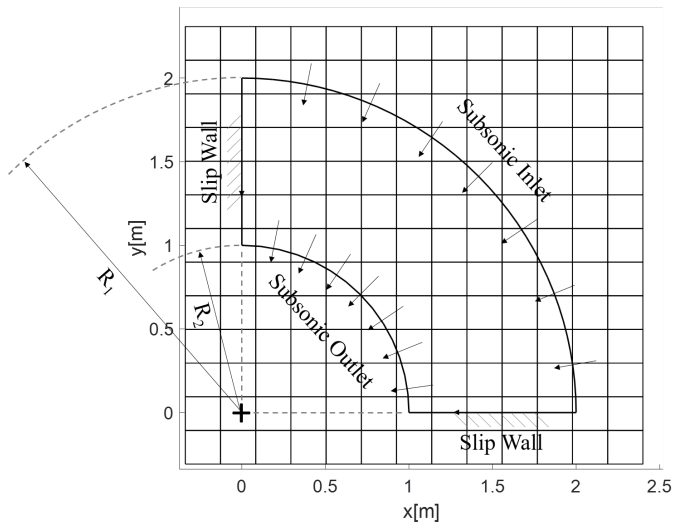


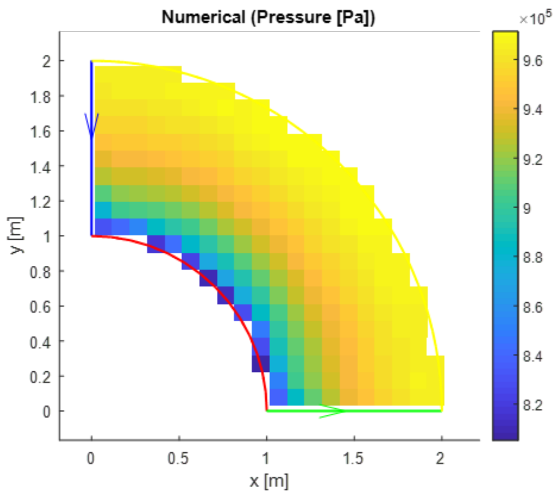
Figure 5.15 Test case setup for subsonic radial flow.

In this test case, the outer radius R_1 is set as a Subsonic inlet, where the physical quantities are reconstructed normal to the boundary to ensure the radial flow direction. The reconstruction employs the normal vectors of the immersed topology generated during the tagging process. The inner radius R_2 is set as a Subsonic Outlet where a null Neumann boundary condition is used to reconstruct the density and velocity quantities. A pressure ratio p_{out}/P_0 of 0.8 is applied at the outlet.

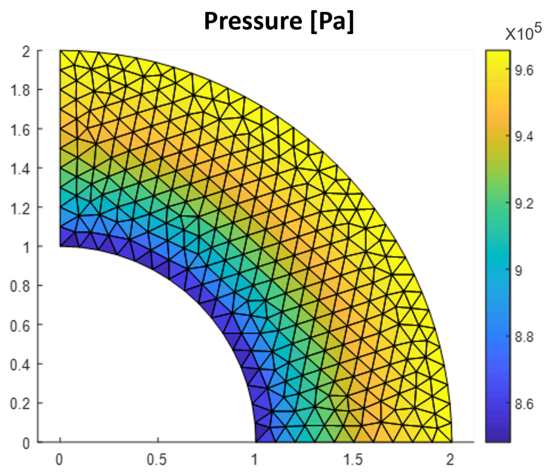
The analytical solution of the radial flow is the quasi-1D isentropic flow solution [125], where the flow converges to the annuls center, and flow properties vary along the radial direction, as in a convergent nozzle.

Four grids are tested (13×13, 24×24, 44×44, and 85×85) using the developed IBM solver. In addition to the comparison to analytical solution, the IBM solution is verified by a comparison to an unstructured grid flow solver applied on four unstructured body-fitted grids of nearly the same resolution. The grid element lengths were approximately (0.2, 0.1, 0.05, and 0.025) m, respectively. The two codes employ the same solver (Roe's solver) with the same spatial and temporal discretisation scheme.

Pressure distributions obtained by the two solvers are shown in Fig. 5.16 for an IBM grid resolution of 24×24 versus the corresponding unstructured body-fitted grid. Qualitatively, the two solvers give the same behavior, with an underestimation of the IBM solver for the pressure value at the exit section of approximately 4.5% for the given resolution.



(a) IBM solution on a Cartesian grid 24×24, average element length is 0.98 m.

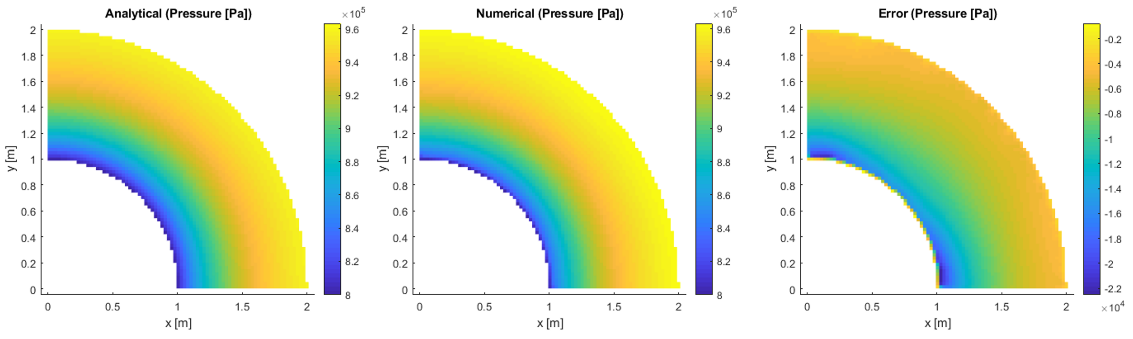


(b) Body-fitted solution on an unstructured grid, average element length is 0.1 m.

Figure 5.16 Pressure distribution of IBM vs. unstructured solvers.

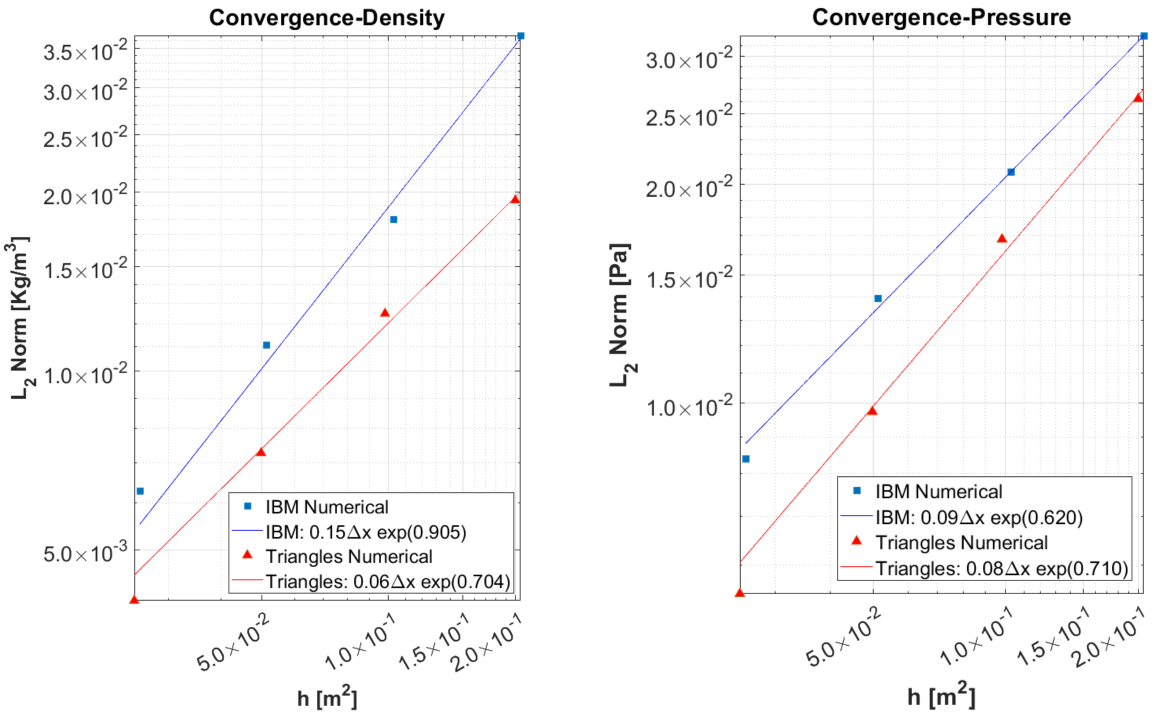
To better analyse these solutions, the order of convergence for the two solvers is evaluated. Each grid is verified with respect to the analytical solution as presented in Fig. 5.17, where the pressure distribution of the IBM solver is compared to the analytical solution. The maximum error is evaluated at the exit radius. This is because of the application of a null Neumann boundary condition at the exit to reconstruct the density and the velocity where the gradients at the exit section are not equal to zero. This limitation is imposed by the linear reconstruction scheme. A higher order reconstruction could solve this issue, by imposing a "free" boundary condition at the outlet.

The order of convergence of the two solver is examined and shown in Fig. 5.18. The two solvers show the same order of error for all grid resolutions, and it follows nearly the same order on convergence ($0.62 - 0.71$ for pressure) and ($0.704 - 0.9$ for density). These results show a consistent error trend for both solvers, with the estimation of a lower absolute value of error for the unstructured solver.



(a) Analytical solution. (b) IBM solution. (c) Pressure error.

Figure 5.17 Pressure distribution of IBM vs. analytical solution on a Cartesian grid of 85×85 (average element length of 0.0247 m).



(a) Density L_2 norm convergence. (b) Pressure L_2 norm convergence.

Figure 5.18 Density and pressure L_2 norm convergence for IBM vs. unstructured solvers.

5.4.6 Confluence of two supersonic flows (CAT16)

The confluence of two supersonic flows is the merging of two flows at different operating conditions leaving a trailing edge of a wedge. The incidence angle of the flow is denoted by θ', where it represents the wedge angle, as shown in Fig. 5.19. Each flow generates a shock wave at the trailing edge denoted by β as measured from the extension of the wedge surface. The region between the shock is characterized by the presence of a slip line, where the properties of the two flows equalize. The aim of this test is to evaluate the capability to solve flows in confluence and capture the shock waves and slip line. The verification is carried out with the analytical solution [126].

The dimensions of the computational domain is ($L_x = 3$ m, $L_y = 10$ m). The inlet is divided into two parts by the location of the trailing edge; the upper inlet, which consists of the upper half of the vertical inlet and the upper horizontal boundary of the domain. At this inlet the flow properties of flow 1 are imposed (M_1, θ'_1, p_1, and T_1). The second inlet is the mirror of the upper inlet, where the flow properties of flow 2 are imposed (M_2, θ'_2, p_2, and T_2). The outlet of the domain is set as a Supersonic outlet represented by the vertical boundary at the right of the domain. This configuration with the boundary conditions are illustrated in Fig. 5.19.

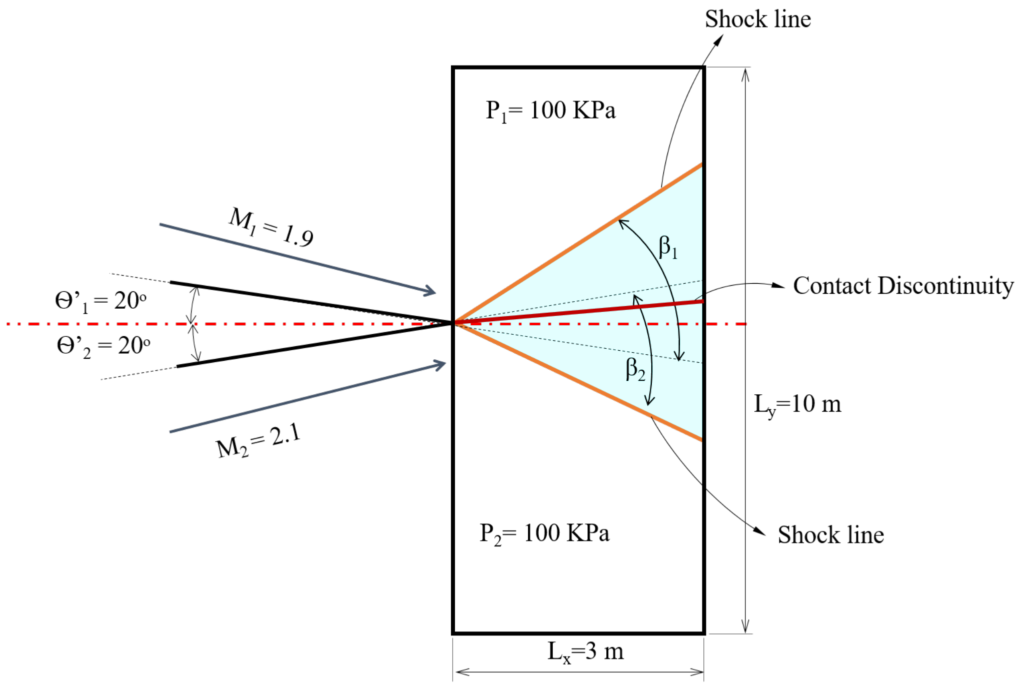


Figure 5.19 Test case configuration, flow angles, and boundary conditions for the confluence of two supersonic flows.

The location of shock wave is predicted analytically, and is used to pre-adapt the grids to follow the predicted shock waves. Four grids where used in this test, the base grid 15 × 150 with one level finer of adaptation that follows the shock line, as illustrated in Fig. 5.20a.

Each finer grid is discretized by a multiple of two with respect to the previous one, these grids are generated using the $Add - on$ feature (`RefineUniformGrid`) presented in Section 3.7.

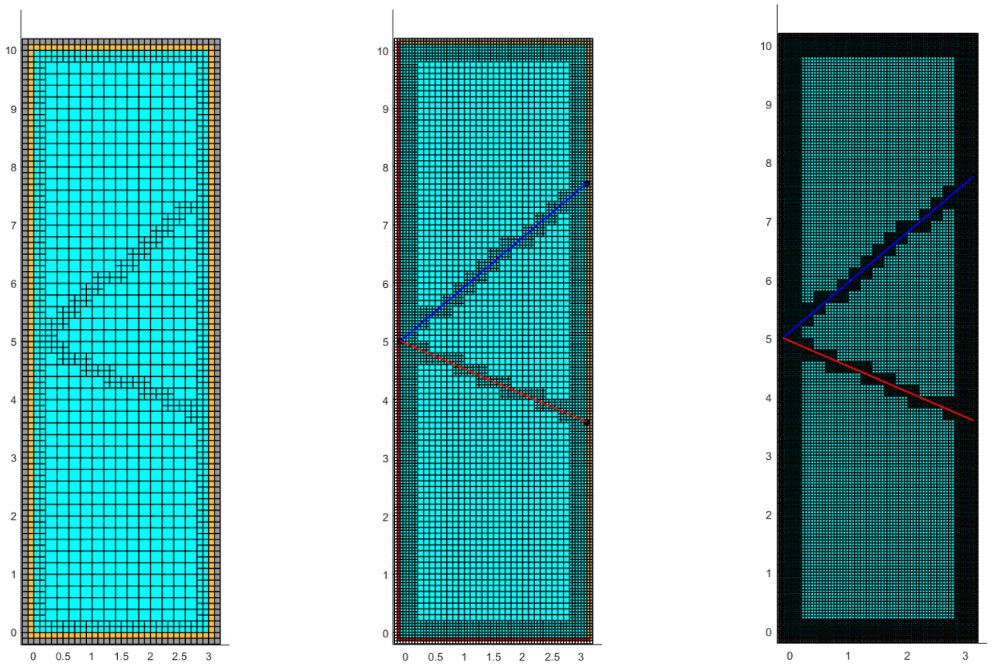
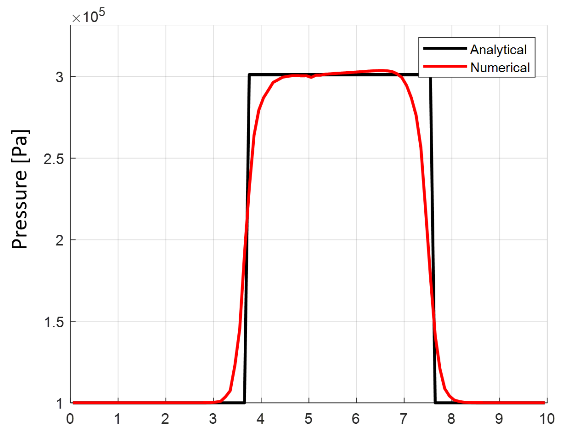

(a) Base grid 15 × 150 with one level of local refinement. (b) Level 1 grid is one level finer than the base grid. (c) Level 2 grid is one level finer than the level 1.

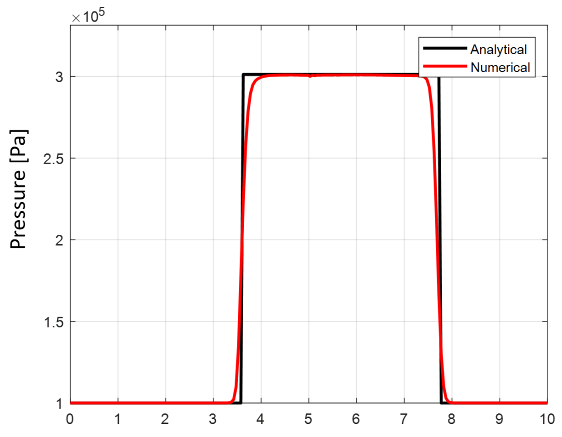
Figure 5.20 Computational domains of pre-adapted refined grids for confluence test case.

The solution at the outlet is captured and compared to the analytical solution. The numerical solutions at the exit for the four grids are shown in Fig. 5.21. The results show a convergence toward the analytical solution with grid refinement, and the error becomes more localized as shown in Fig. 5.22. The order of convergence of the pressure L_2 norm is of around 0.45, shown in Fig. 5.23, which agrees with similar test cases that give an order of around 0.5 for the cases of flow discontinuities and shock waves for a first order flow solver [127].

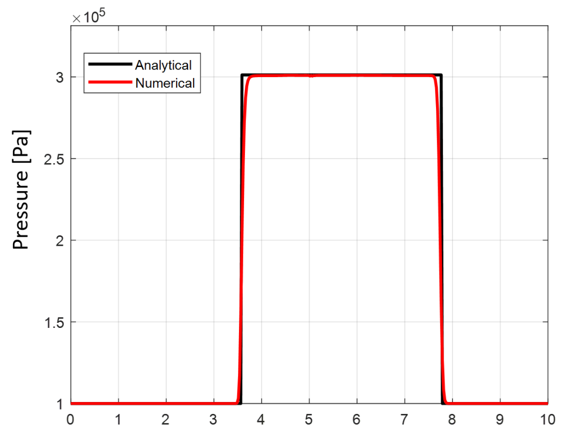
The pressure and density distribution is given in Fig. 5.24a, where the slip line is captured in Fig. 5.24b.



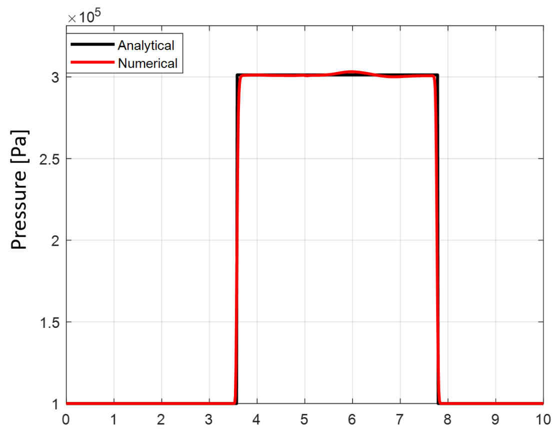
(a) Base grid.



(b) Level 1 grid.

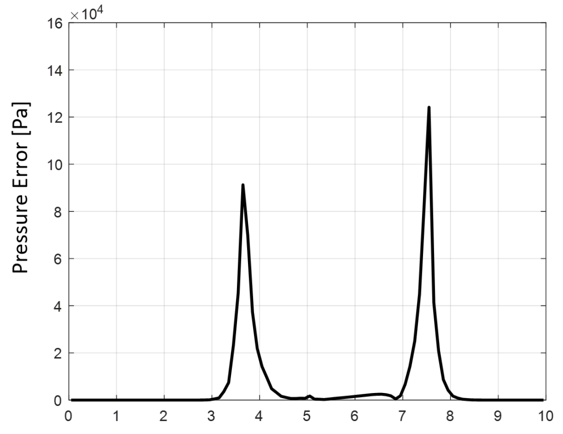


(c) Level 2 grid.

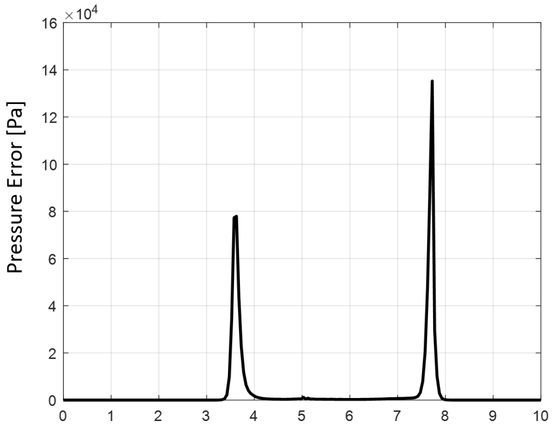


(d) Level 3 grid.

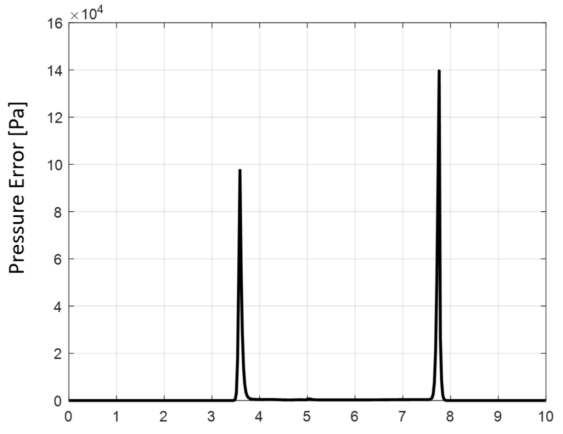
Figure 5.21 Pressure distribution along the outlet section for different grid resolutions.



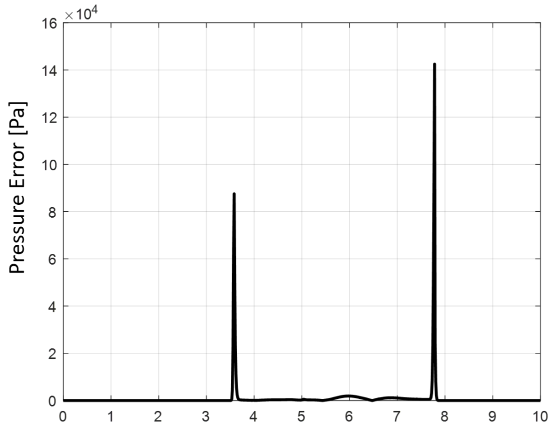
(a) Base grid.



(b) Level 1 grid.



(c) Level 2 grid.



(d) Level 3 grid.

Figure 5.22 Pressure error along the outlet section for different grid resolutions.

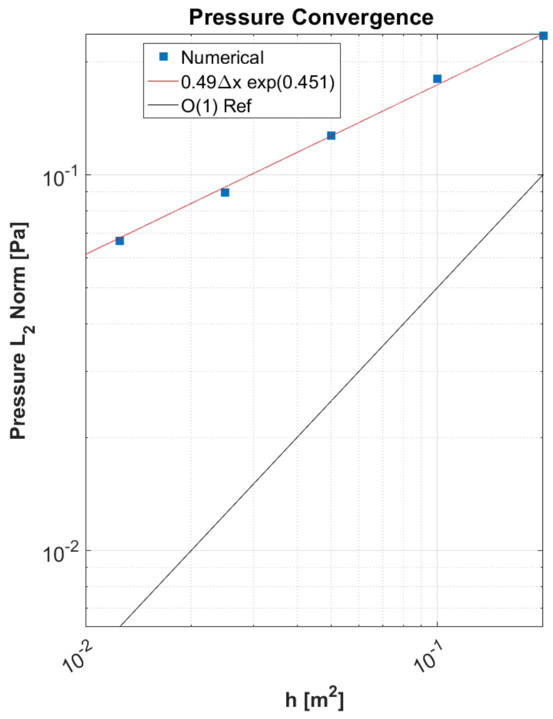
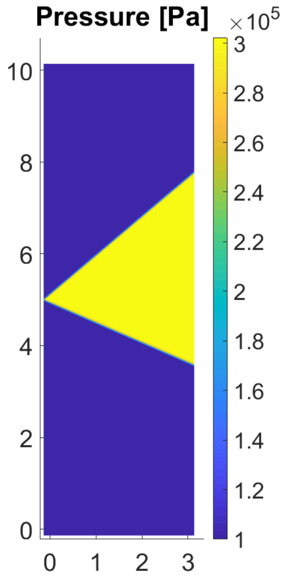
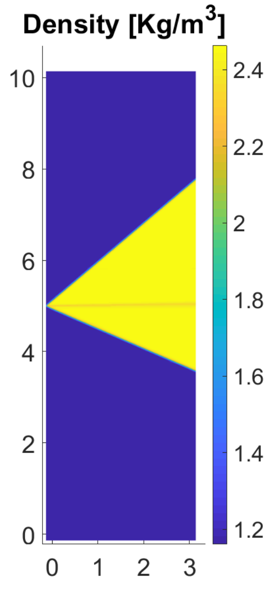


Figure 5.23 Pressure L_2 norm convergence for supersonic confluence of two flows.



(a) Pressure distribution.

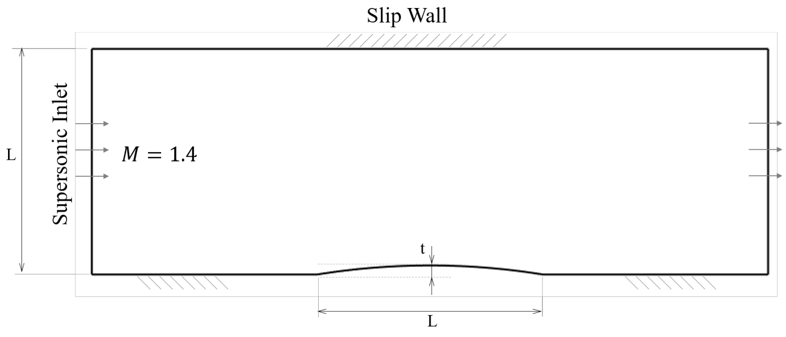


(b) Density distribution.

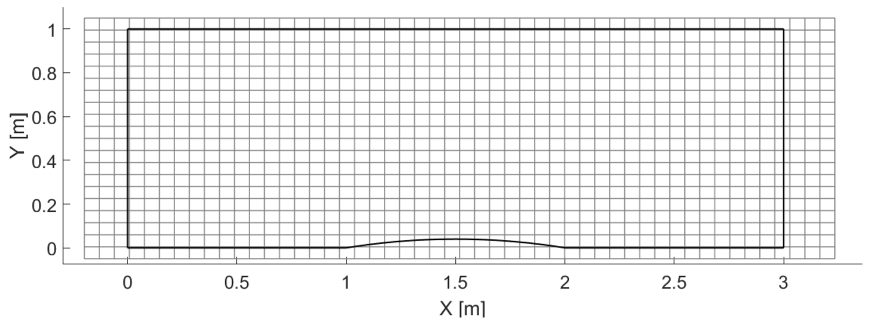
Figure 5.24 Numerical results on a Level 3 grid.

5.4.7 Supersonic flow over a bump (CAT17)

This test aims to verify supersonic flow solution over curved boundaries. The test consists of a channel 3 m long, with a 1 m bump located at half length of the channel. The ratio of bump thickness to length (t/L) is 4%. The inlet Mach number is 1.4. Four Cartesian grids are tested (50×20, 100×40, 200×80, and 400×160). The test case configuration and grids are illustrated in Fig. 5.25.



(a) Configuration and boundary conditions of the supersonic bump test case.

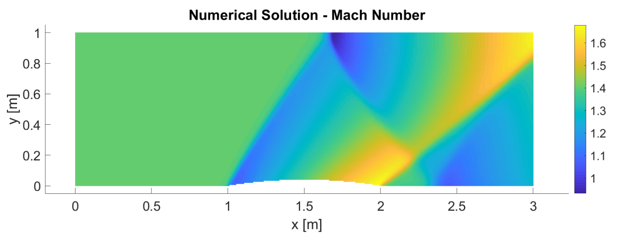


(b) Supersonic bump topology immersed in a Cartesian grid (50×20).

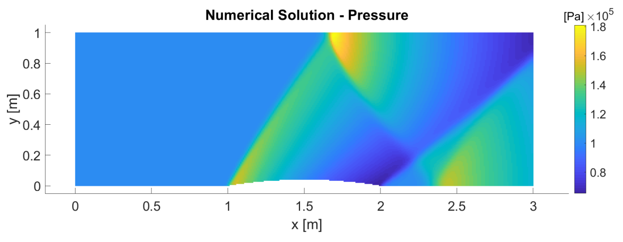
Figure 5.25 Test case setup for a supersonic bump configuration.

The flow properties contours are shown in Fig. 5.26, where the shock waves interference and reflection from walls is captured. The numerical results are compared to Ni [128], while others also conducted the same test case [129]. The Mach number distribution along the upper and lower walls is shown in Fig. 5.27, the solution converges to the reference solution with grid refinement. The first order numerical scheme shows a diffusive behaviour around flow discontinuities, as presented in Fig. 5.27a and 5.27b. This behaviour is reduced by grid refinement, where the flow features are better defined on finer grids, as shown in Fig. 5.27d.

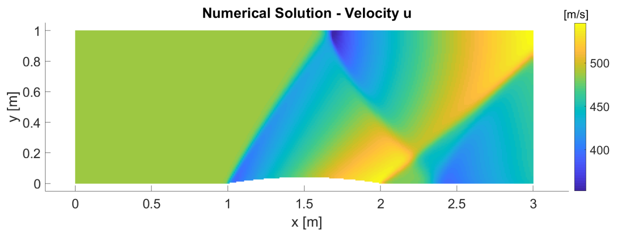
A second order discretization scheme could capture flow discontinuities in higher accuracy for the same grid resolution.



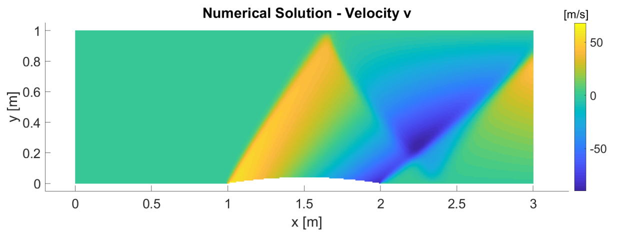
(a) Mach number.



(b) Pressure.



(c) u-velocity.



(d) v-velocity.

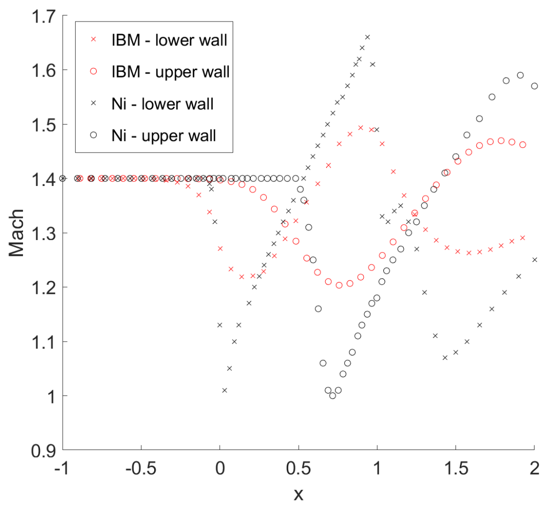
Figure 5.26 Distribution of physical properties over a supersonic bump $M = 1.4$.

Since no analytical solution for the current test is available, a grid convergence method is conducted to evaluate the apparent order of accuracy of the tested grids [130].

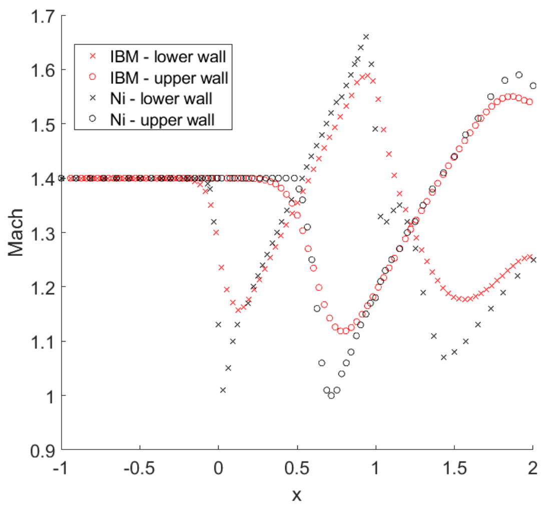
The apparent order of accuracy p is evaluated using the three tested grids as follows:

$$p = \frac{1}{\ln(r_{21})} \mid \ln \mid \varepsilon_{32}/\varepsilon_{21} \mid\mid \tag{5.6}$$

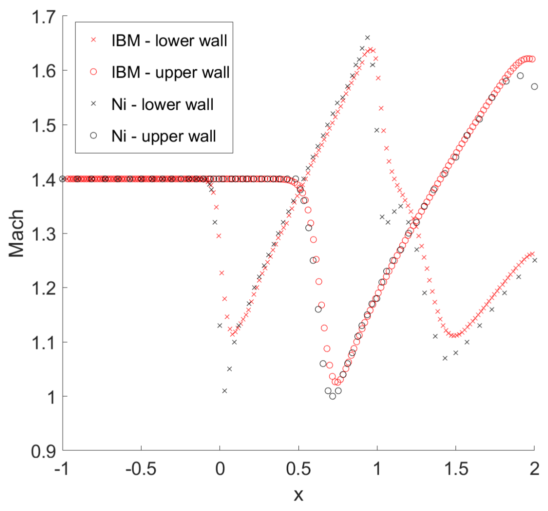
where $r = h_2/h_1$, and h is the representative mesh size, or cell area. For the given meshes; $h_1 < h_2 < h_3$, and ε_{ij} is the L_2 norm of the error between mesh i and j. Since the grids are uniformly refined, so $r = 2$. The apparent order of accuracy is evaluated as 0.551, this value is consistent with the predicted order of accuracy to be around 0.5.



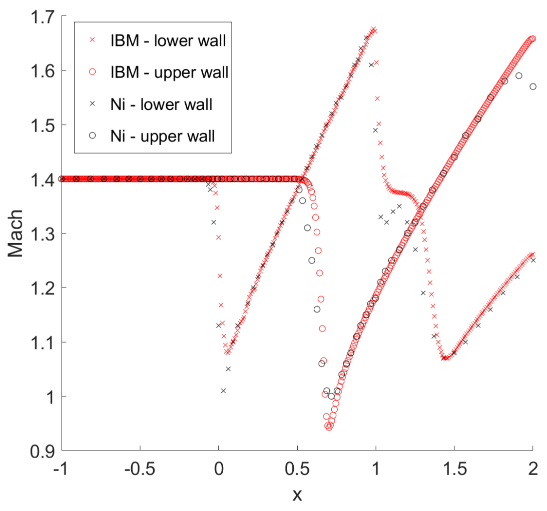
(a) Grid 50 × 20.



(b) Grid 100 × 40.



(c) Grid 200 × 80.



(d) Grid 400 × 160.

Figure 5.27 Mach number distribution on upper and lower walls.

5.5 Unsteady flows on a single grid

This category of tests is to verify the numerical solution of unsteady flows on single grids. It consists of three cases: a) Shock tube flow, b) Tube in impulsive motion, and c) Unsteady flow over a wedge.

5.5.1 Shock tube flow (CAT21)

The shock tube test case consists of a tube closed at its two ends, and divided into two regions by a membrane placed inside the tube. Each part of the tube is filled with the same gas at different conditions (pressure, Mach number, temperature) initially at rest (at $t = 0$, $u = 0$). The high pressure region is designated by the subscript $_L$ and located in the left part of the tube, and the low pressure region is designated by the subscript $_R$ and located in the right part of the tube, as illustrated in Fig. 5.28.

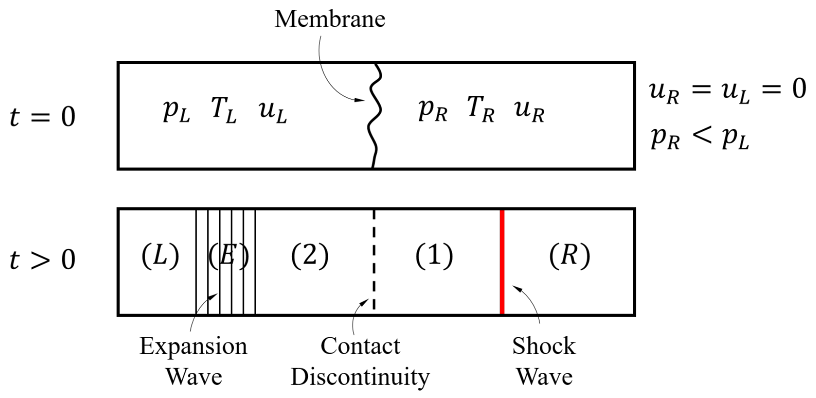
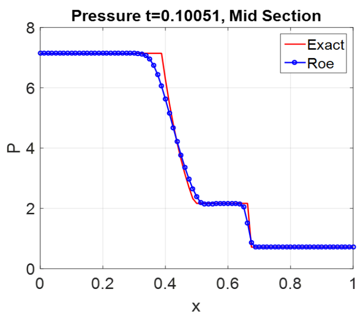


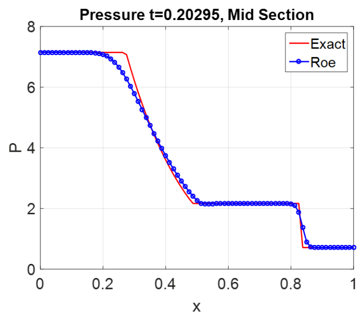
Figure 5.28 Shock tube configuration.

A sudden fracture of the membrane will cause the flow to go through the equilibrium state to equalize the pressure inside the tube, the high pressure region will expand, and generate an expansion wave that travels to the left, the region of expansion is called (expansion fan) and designated in Fig. 5.28 by Region (E). The low pressure region is compressed by the the expansion from the left side, and a shock wave is formed at the low pressure region that travels to the right. The expansion fan and the shock wave are separated by the *contact discontinuity* where entropy is discontinuous.

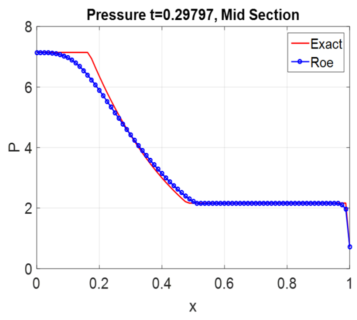
A 2D shock tube (1 m $\times$ 1 m) is constructed and the solution at midsection is compared to the exact solution [131] at different times. The developed IBM code is able to capture the flow discontinuities, as shown in Figs. 5.29-5.31. The figures show the displacement of the shock and the expansion waves with time, and agree with the exact solution at the contact discontinuity, where the pressure and velocity are continuous, while the density is discontinuous.



(a) $t = 0.10051$

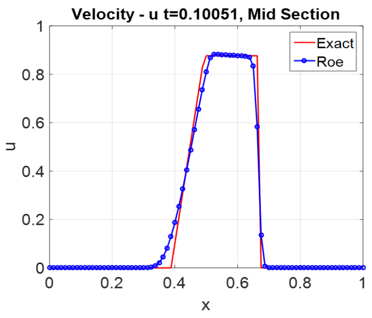


(b) $t = 0.2095$

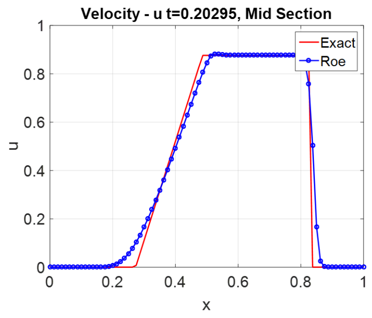


(c) $t = 0.29797$

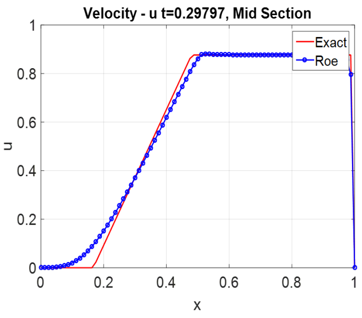
Figure 5.29 Pressure distribution at mid-section of a shock tube at different times, grid size 81×81.



(a) $t = 0.10051$

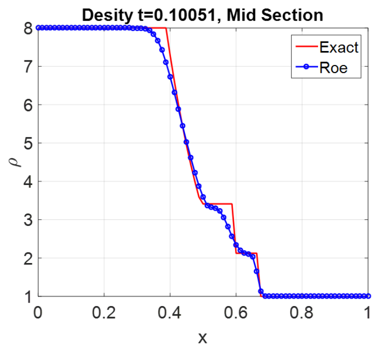


(b) $t = 0.2095$

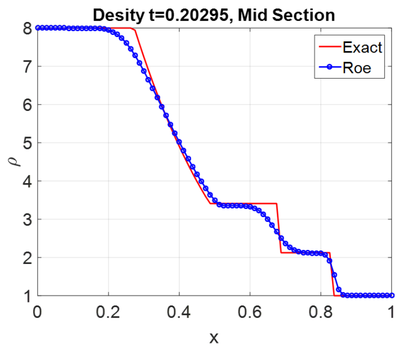


(c) $t = 0.29797$

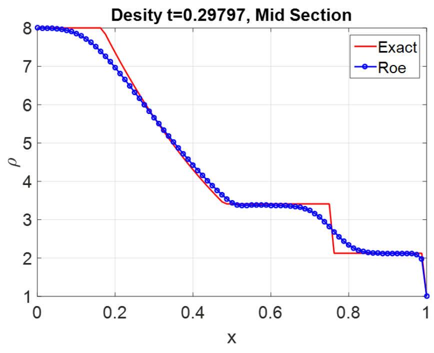
Figure 5.30 u-velocity distribution at mid-section of a shock tube at different time frame, grid size 81×81.



(a) $t = 0.10051$



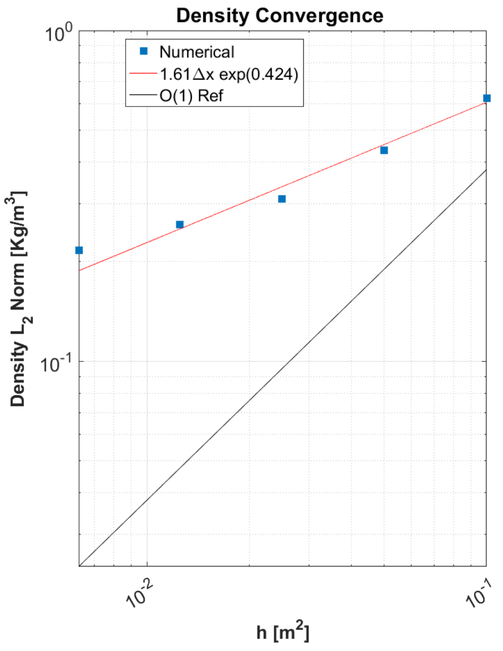
(b) $t = 0.2095$



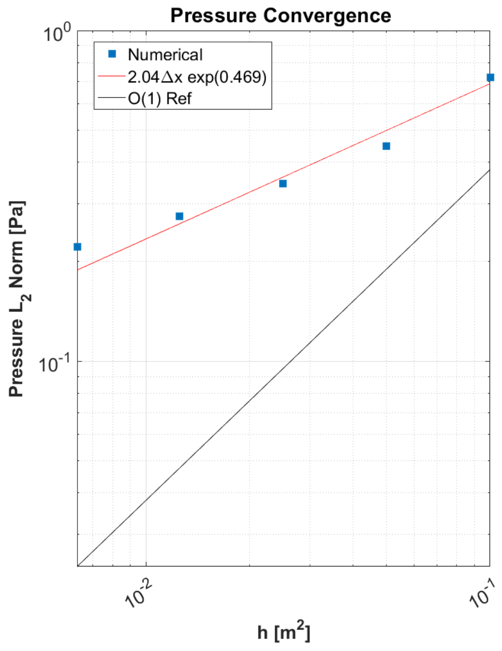
(c) $t = 0.29797$

Figure 5.31 Density distribution at mid-section of a shock tube at different time frame, grid size 81×81.

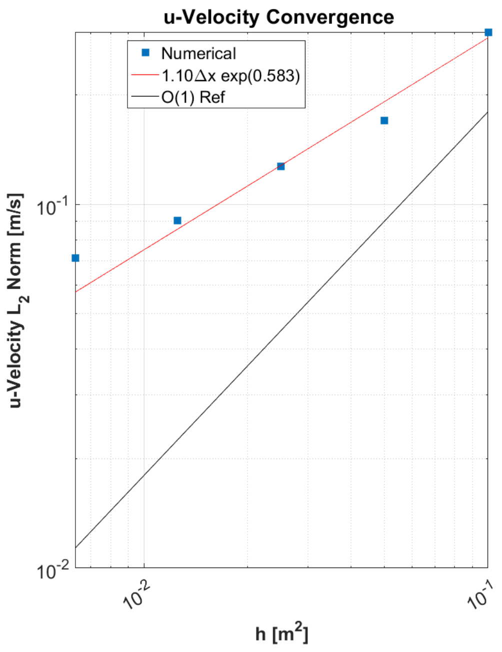
The order of convergence of density, pressure, and u-velocity is 0.42, 0.47, and 0.58 respectively, as shown in Fig. 5.32.This agrees with the predicted value where a flow discontinuity occurs [127].



(a) Density.



(b) Pressure.



(c) u-velocity.

Figure 5.32 Order of convergence of L_2 norm of physical quantities of shock tube.

5.5.2 Tube in impulsive motion (CAT22)

This test aims to verify the implementation of the ALE approach to the IBM solver presented in Section 4.5, by evaluating the solvers' ability to handle the grid motion.

The test consists of a tube closed at its two ends filled with a homogeneous gas with a given thermodynamic parameters (p, T, u). Initially the confined gas and the tube are set at rest (at $t = 0 \ \rightarrow u = 0$, $u_{tube} = 0$).

As the tube moves impulsively to the right (*i.e.*, at $t > 0 \ \rightarrow u_{tube} > 0$), the left wall of the tube will hit the stagnant flow, and a shock wave propagates to the right at a constant speed greater than the speed of sound. The motion of the tube to the right will produce an under-pressure on the right wall, and an expansion fan will propagates to the left. This propagation of waves divides the tube into four regions: a) Region 2 behind the shock wave and limited to the location of the traveling shock wave x_s. The velocity in this region is equal to the velocity of the tube u_{tube}. b) Region 3 behind the expansion fan and limited to the tail of the expansion fan located at x_t. The velocity in this region is also equal to u_{tube}. c) Region 4 as the expansion region of the fan. d) Region 1 is the region between the shock wave at x_s and the advancing front of the expansion fan at x_{adv}. This is the non affected region by the motion before reaching the steady state condition. A schematic representation of this configuration is illustrated in Fig. 5.33

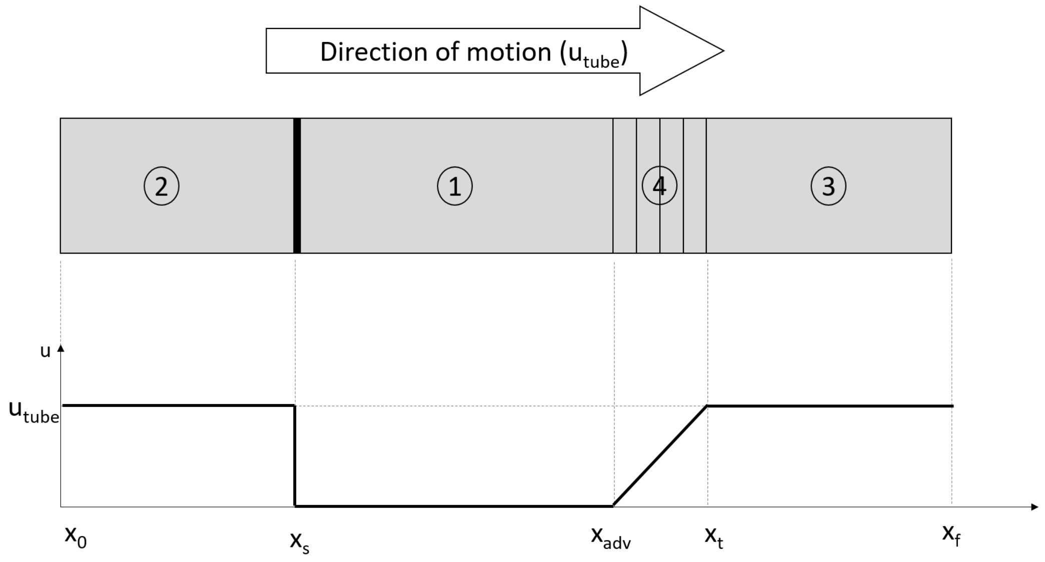


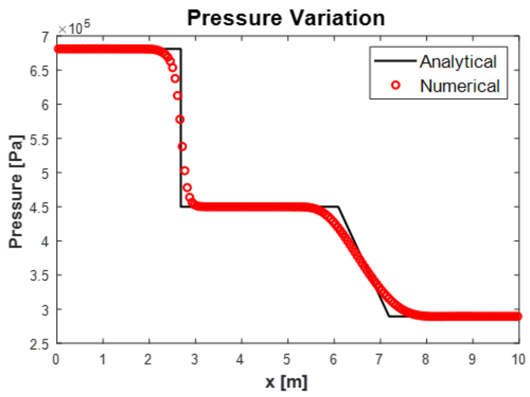
Figure 5.33 Schematic representation of a tube in impulsive motion with operating zones.

The computational domain is a tube of 10 m length and 1 m of diameter, tested on four grids (50×5, 100×10, 200×20, 400×40). The operating conditions of the test case are listed in Table 5.6

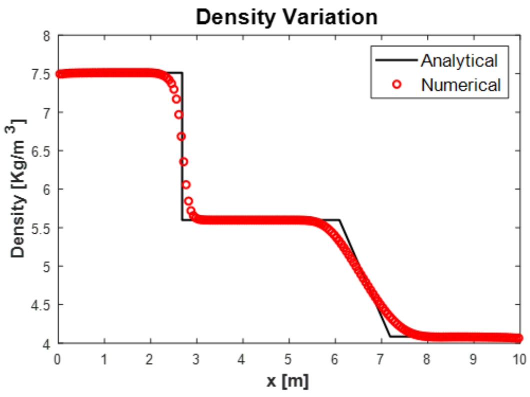
Table 5.6 Operating conditions of tube in impulsive motion

Parameter	Value
p	45 KPa
u_0	0 m/s
T	280 K
u_{tube}	102.5 m/s

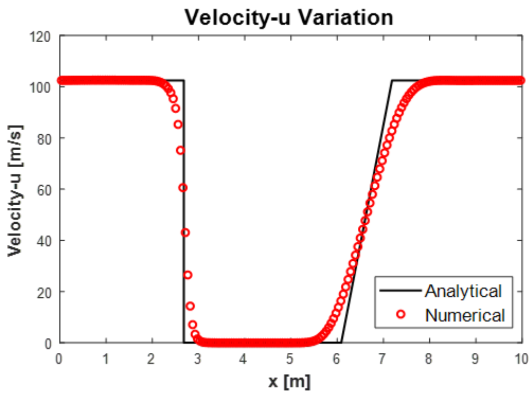
The solution at mid-section of the tube is illustrated in Fig. 5.34 compared to the analytical solution [132] at $t = 0.0089s$ for a grid resolution of 200×20. The results shows a good agreement with the analytical solution. The numerical order of convergence of the flow variables is near the 0.6 as shown in Fig. 5.35.



(a) Pressure.

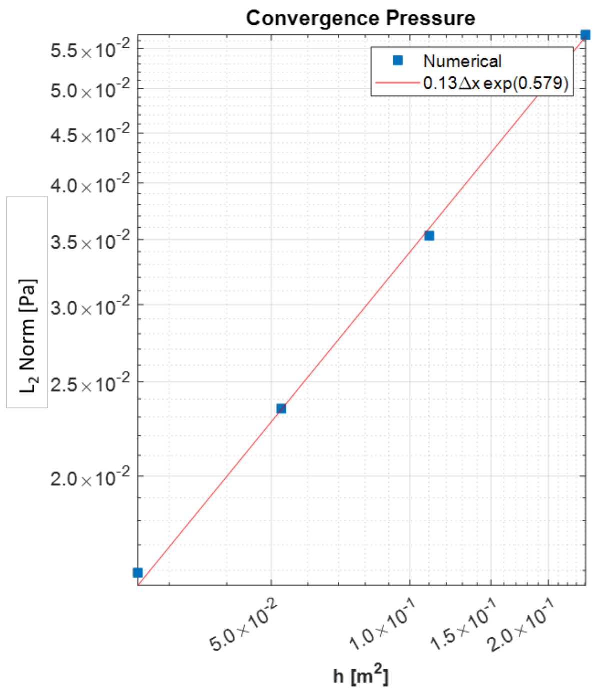


(b) Density.

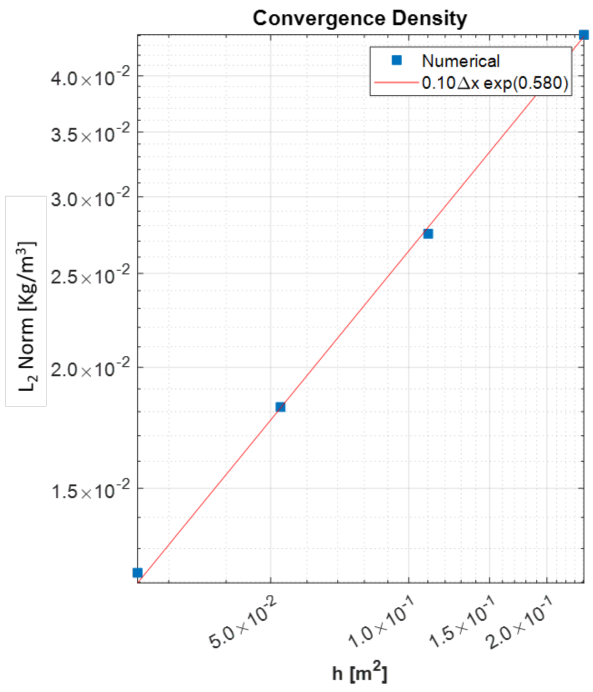


(c) u-velocity.

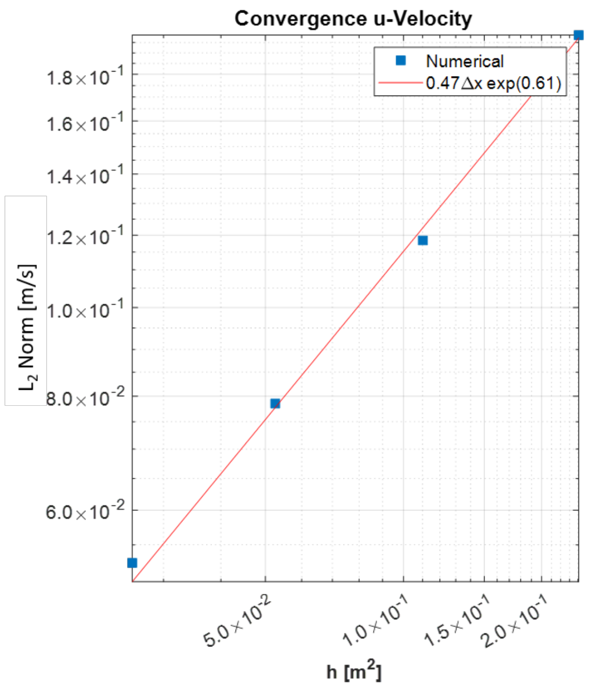
Figure 5.34 Flow properties at mid-section of a tube in impulsive motion for a grid resolution of 200×20 at $t = 0.0089s$.



(a) Pressure.



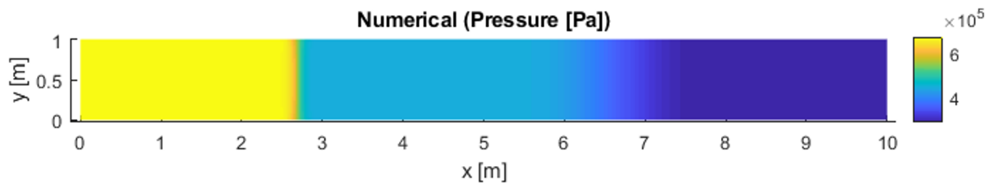
(b) Density.



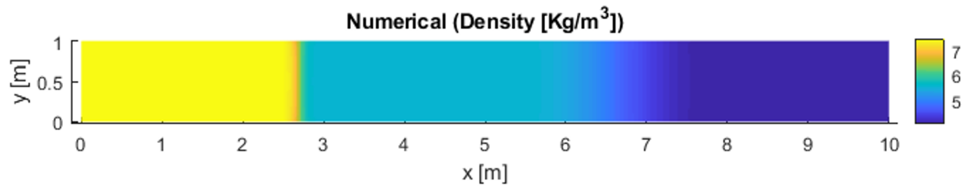
(c) u-velocity.

Figure 5.35 Flow properties L_2 norm convergence of a tube in impulsive motion.

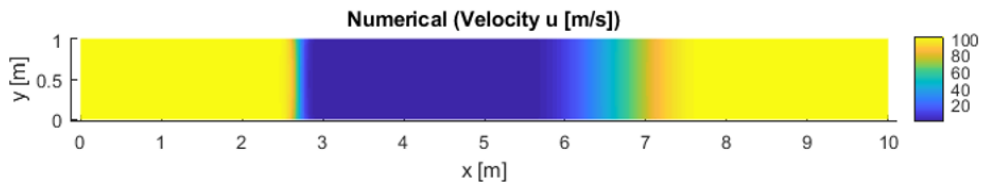
The flow properties distribution inside the moving tube is illustrated at Fig. 5.36 at $t = 0.0089s$, which shows the sharp discontinuity along the shock, at approximately 2.6 m. The 2D solution is consistent with the 1D solution, the flow inside the tube remains unidirectional for all cells.



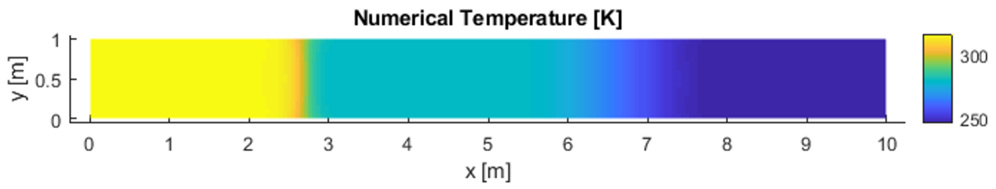
(a) Pressure



(b) Density



(c) u-velocity



(d) Temperature

Figure 5.36 2D flow properties distribution in a tube in impulsive motion, grid 400×40 at $t = 0.0089s$.

5.5.3 Unsteady supersonic flow over a Wedge (CAT23)

This test is conducted to assess the ability to capture unsteady shock development over a static symmetric wedge at Mach 2. In addition, this test case will be used as a reference case to verify the solution of a supersonic flow over the same wedge on overset grids. This comparison will be addressed later in Section 5.7.1.

The computational domain is a uniform channel of 2 m diameter, as shown in Fig. 5.37 where a symmetric wedge is placed at the center line of the channel, placed at 1 m far from the domain inlet. The wedge angle and length are 30 deg and 1 m, respectively.

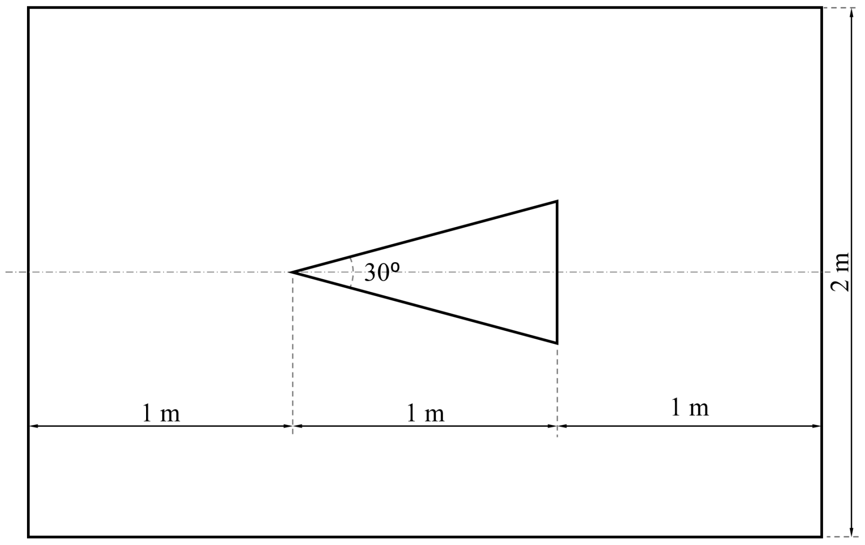
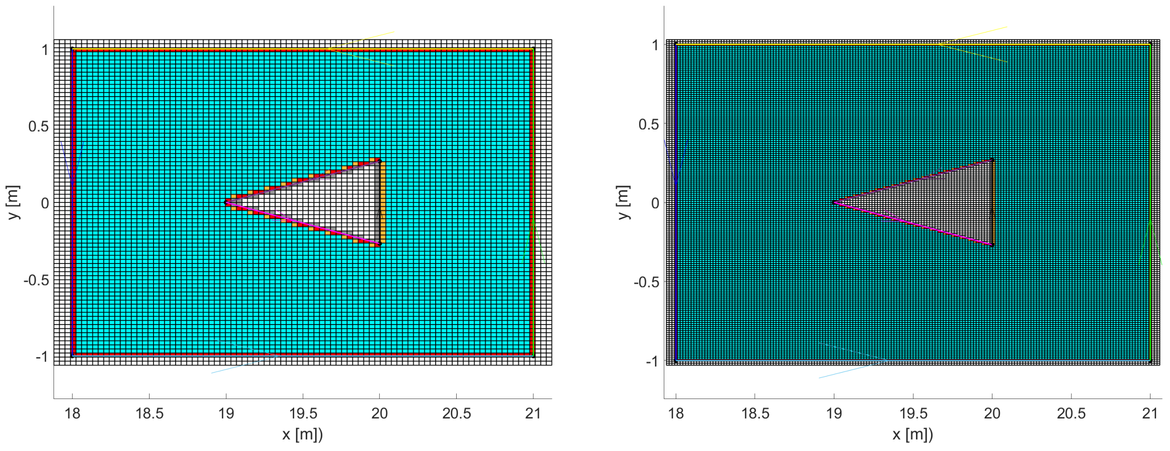


Figure 5.37 Computational domain of supersonic wedge test case.

The wedge boundaries are set as Slip Walls, the domain inlet operating conditions are listed in Table 5.7. All other boundaries are set as Supersonic Outlet. Three Cartesian meshes are examined (90×80, 180×160, and 360×320), two sample grids are illustrated in Fig. 5.38.



(a) Grid resolution 90×80.

(b) Grid resolution 180×160.

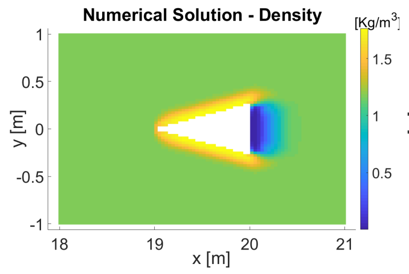
Figure 5.38 Immersed wedge topology in Cartesian grid for different grid resolutions.

Table 5.7 Operating conditions of supersonic wedge on single grid

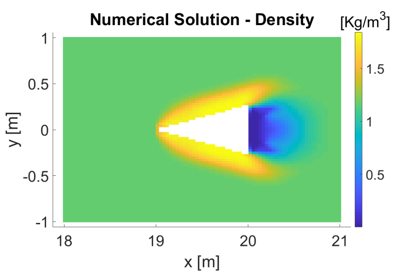
Parameter	Value
p	100 KPa
M	2
T	300 K
α	0 deg

The flow enters the domain horizontally ($\alpha = 0$) at Mach 2, hits the wedge boundary, and a shock wave is developed at the leading edge of the wedge. The development of the shock with time is illustrated in Fig. 5.39. The solution attains the steady state condition after 2000 time steps at a CFL of 0.75. At 4000 time steps, after 0.26 seconds, in Fig. 5.39k, the shock wave, the expansion wave and the wake are fully developed and stabilized.

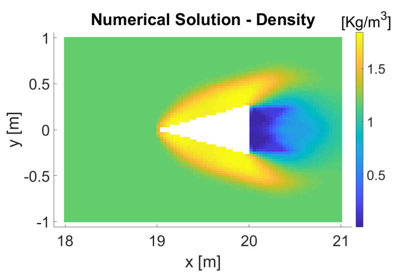
The verification of this test case with the analytical solution of shock wave theory and Prandtl Meyer expansion is not straight forward. This is due to the presence of the wake behind the body that contributes to a complex structure of the flow field.



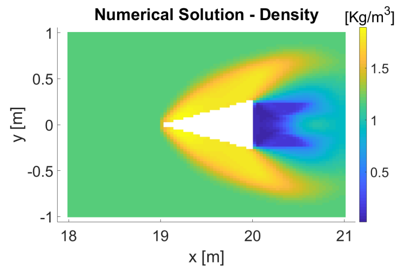
(a) $n_t = 20$.



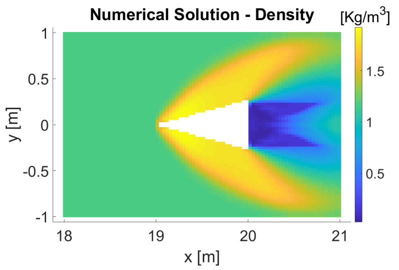
(b) $n_t = 60$.



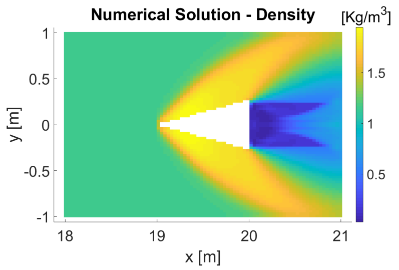
(c) $n_t = 100$.



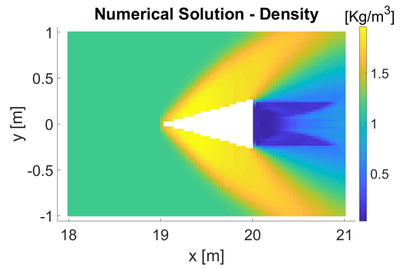
(d) $n_t = 140$.



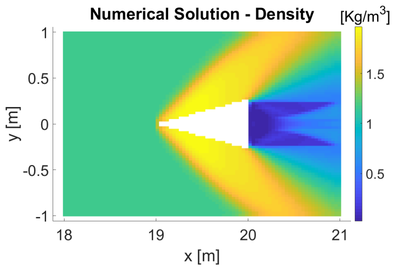
(e) $n_t = 180$.



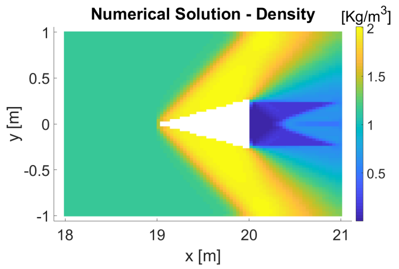
(f) $n_t = 220$.



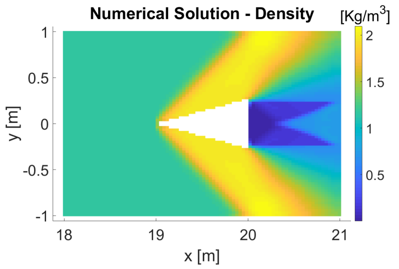
(g) $n_t = 260$.



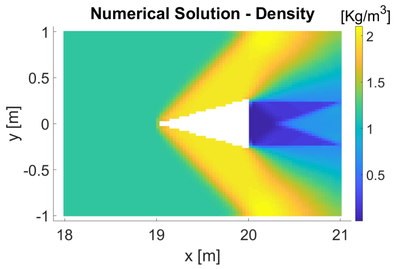
(h) $n_t = 300$.



(i) $n_t = 600$.



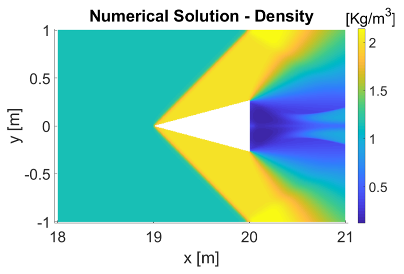
(j) $n_t = 1000$.



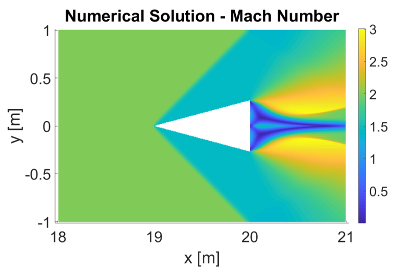
(k) $n_t = 4000$.

Figure 5.39 Density distribution of the shock development over a wedge at Mach 2, grid resolution of 90×80.

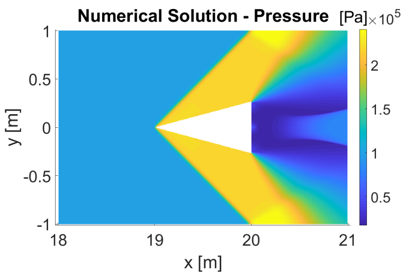
A better definition of the flow structure is shown in Fig. 5.40, where all flow variables are illustrated on the finest grid 360×320.



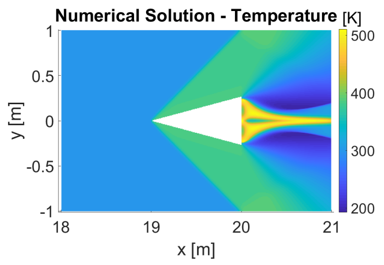
(a) Density.



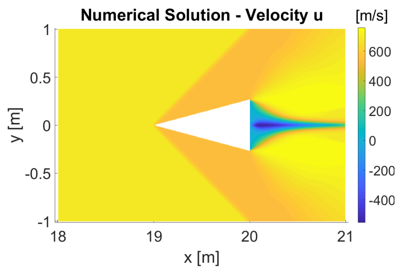
(b) Mach number.



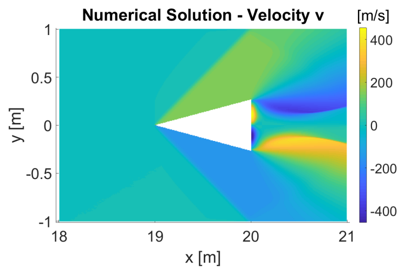
(c) Pressure.



(d) Density.



(e) u-velocity.



(f) v-velocity.

Figure 5.40 Flowfield properties distribution of the shock development over a wedge at Mach 2, grid resolution is 360×320 at steady state conditions.

5.6 Steady flows on overset grids

This category of tests aims to verify the implementation of the overset module with the IBM solver, and verify the solution of steady flows on overset grids.

5.6.1 Supersonic tube flow with stationary overset grid (CAT31)

This is the first test to verify the overset integration with stationary overlapping grids. The test aims to evaluate a constant supersonic flow inside a tube with the presence of an overlapping empty grid inside the tube. The computational domain consists of two grids, the reference grid and the overlapping grid. The reference grid contains the tube topology with dimensions of (16×1 m). The horizontal boundaries of the tube are set as Slip walls, the vertical boundaries are set as supersonic inlet/outlet. An empty overlapping gird of dimensions (1×0.5 m) patches the reference grid as illustrated in Fig. 5.41.

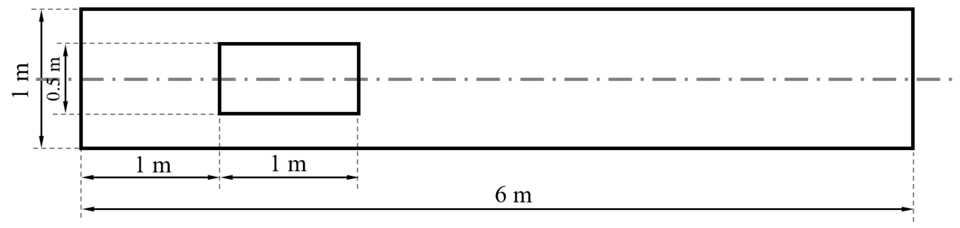
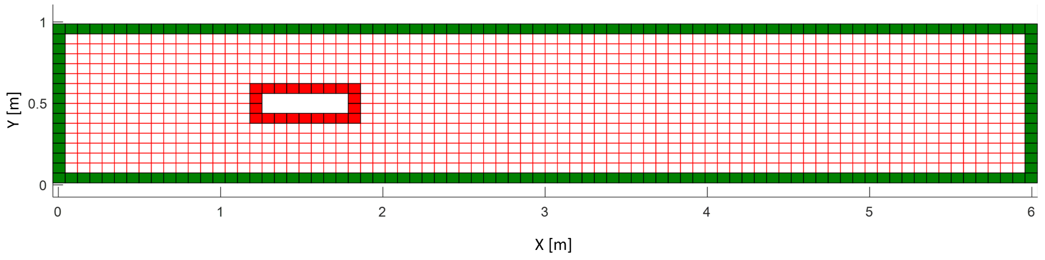


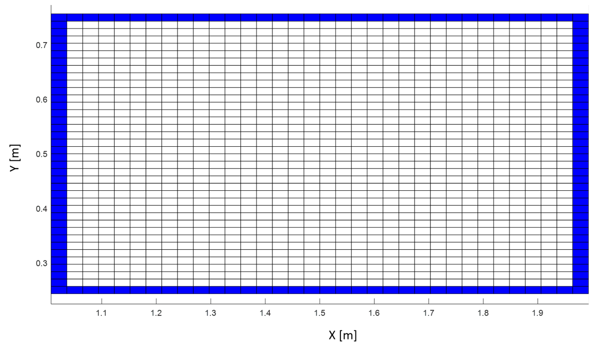
Figure 5.41 Computational domain of a stationary overset supersonic tube.

The reference grid resolution is 100×20, the mesh tagging is illustrated in Fig. 5.42a, where the green cells represents the interface cells where the boundary condition is applied. A set of cells are disabled by the Hole Cutting procedure to de-activate the cells in the shadow of the overlapping grid. Around the hole, a layer of interpolated cells are tagged by the overset module, these cells are the cells responsible for data transfer between the two grids and colored by red.

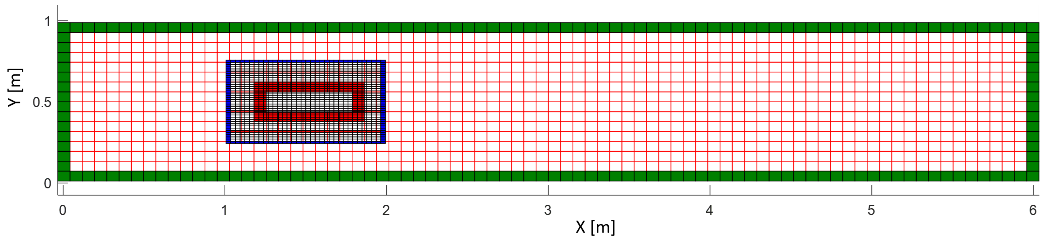
The overlapping grid resolution is 40×40, tagged by the overset module to identify the list of interpolated cells. These cells are the outer layer of the overlapping grid and colored by blue, as shown in Fig. 5.42b. The assembled computational domain is illustrated in Fig. 5.42c.



(a) Reference grid, 100 × 20, with mesh tagging and overset tagging.



(b) overlapping grid, 40 × 40, with overset tagging.

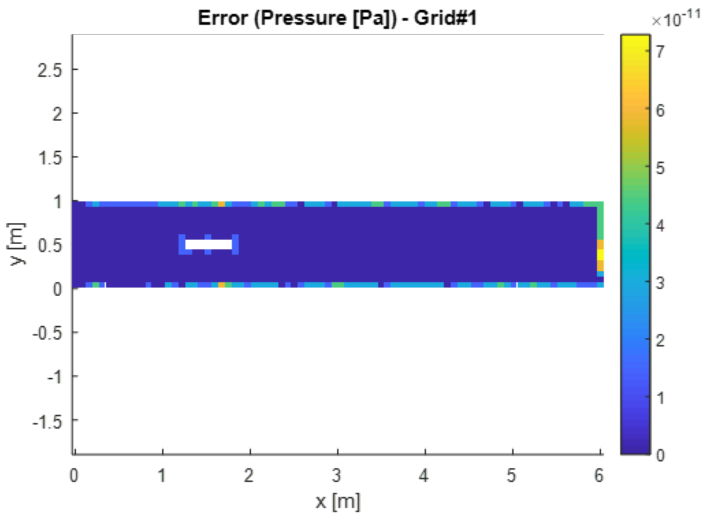


(c) Assembled grid - Reference grid and overlapping grid.

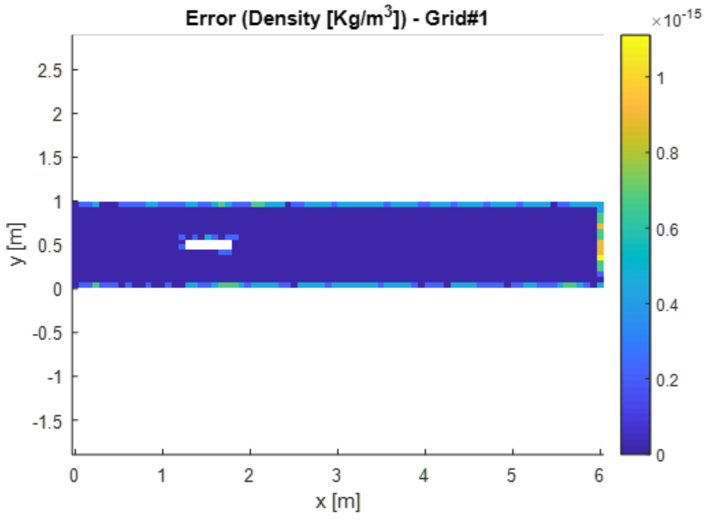
Figure 5.42 Overset computational domain for supersonic flow tube.

An inlet Mach number of 2 at a static pressure of 100 KPa and static temperature of 300 K are applied at the domain inlet. The numerical flow solution is compared to the inlet conditions to assess the accuracy of the numerical solution.

The error on each grid is illustrated in Figs. 5.43 and 5.44, the observed error on both grids is of the order of machine zero. This result agrees with what is expected from the reconstruction scheme. Since the flowfield is constant, the reconstruction is expected to be exact since it employs a linear reconstruction polynomial. The solution on the overset grid is illustrated in Fig. 5.52, where the flow variables are constant along the entire domain with an error at the overset interface close to machine zero.

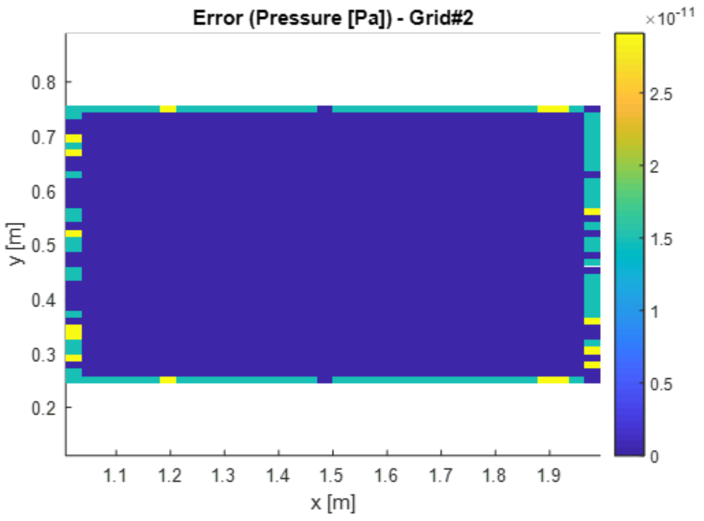


(a) Pressure error.

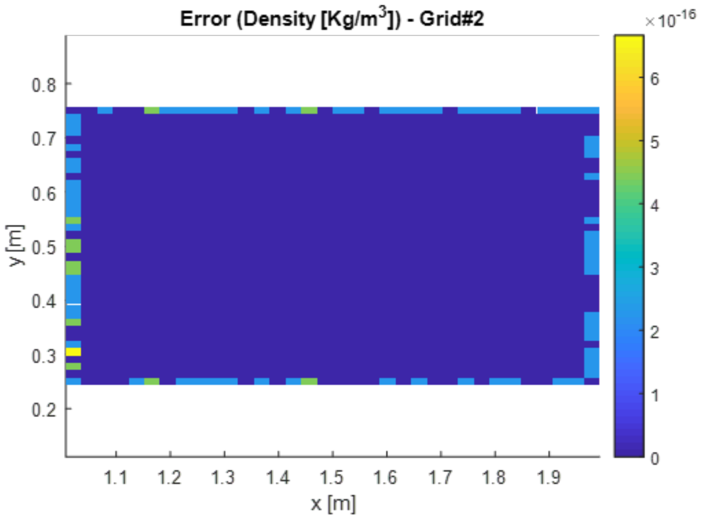


(b) Density error.

Figure 5.43 Solution error of flow properties on the reference grid.

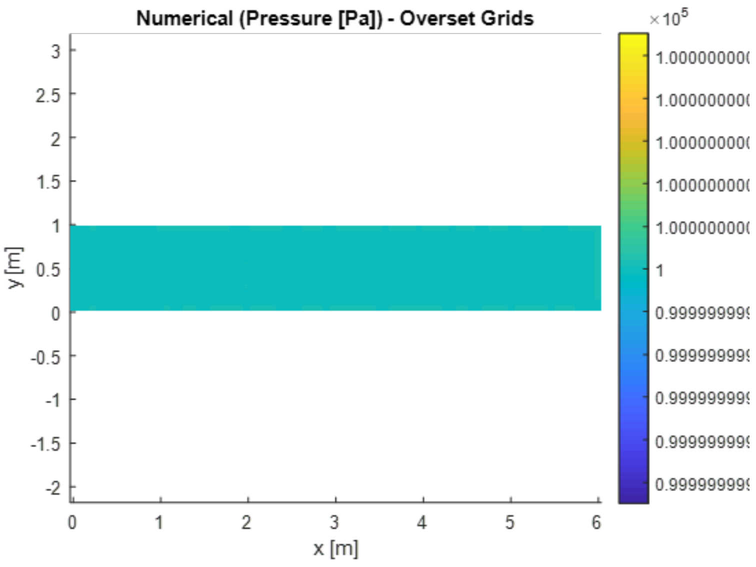


(a) Pressure error.

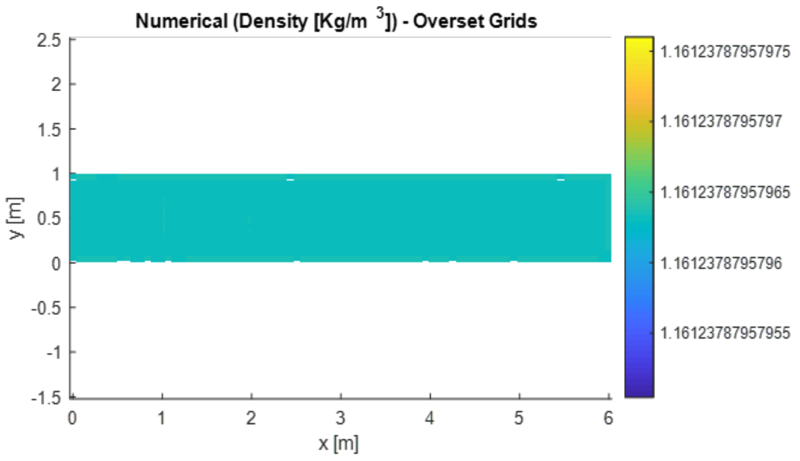


(b) Density error.

Figure 5.44 Solution error of flow properties on the overlapping grid.



(a) Pressure.



(b) Density.

Figure 5.45 Solution of flow properties on the overset grids.

5.6.2 Supersonic tube flow with moving overset grid (CAT32)

This test evaluates the flow solution and the overset reconstruction of a moving overset grid through a constant supersonic flow. The same computational domain of test CAT31, presented in Section 5.6.1, is used with the same boundary conditions. In this test the overset grid moves with a travel speed $u_{Grid} = 100$ m/s, as illustrated in Fig. 5.46.

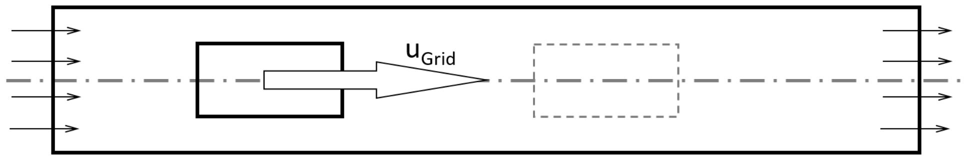


Figure 5.46 Computational domain of moving overset supersonic tube.

The error of the numerical solution is evaluated along the trajectory, and reported in Figs. 5.47 and 5.48 at three positions; the initial position, at 500 time steps (nearly at mid-trajectory), and at 999 time steps (before reaching the end of tube). The pressure error along the grid motion is bounded by 5×10^{-10} on both, reference and overlapping grids. The consistent results show the ability of the reconstruction scheme to reconstruct the solution for stationary and moving grids in uniform flows without triggering an error of high order than the CPU precision of the overset reconstruction.

The pressure distributions along the trajectory remain constant on both grids, as illustrated in Figs. 5.48 and 5.50.

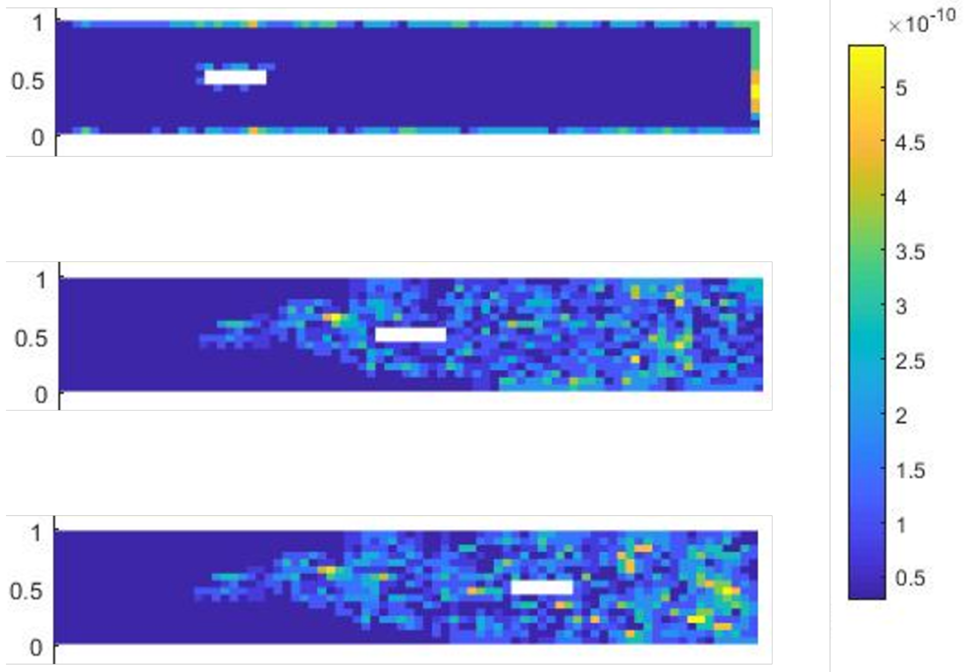


Figure 5.47 Pressure error on reference grid at initial position, mid-trajectory, and at the end of trajectory. The position of the hole changes with the mobile grid motion.

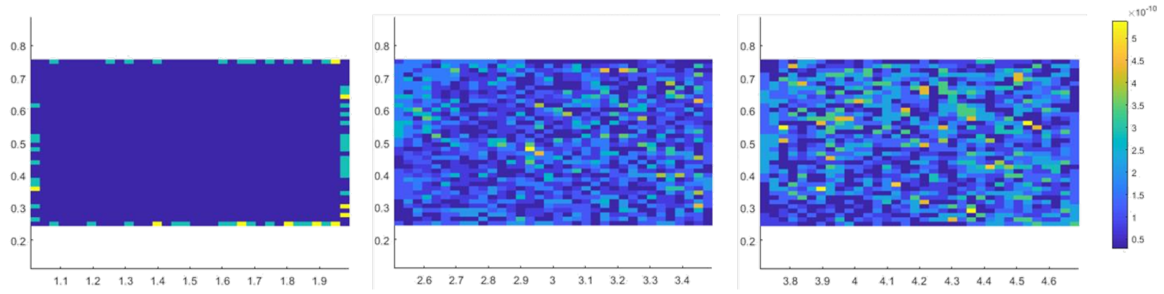


Figure 5.48 Pressure error on overset grid at initial position, mid-trajectory, and at the end of trajectory.

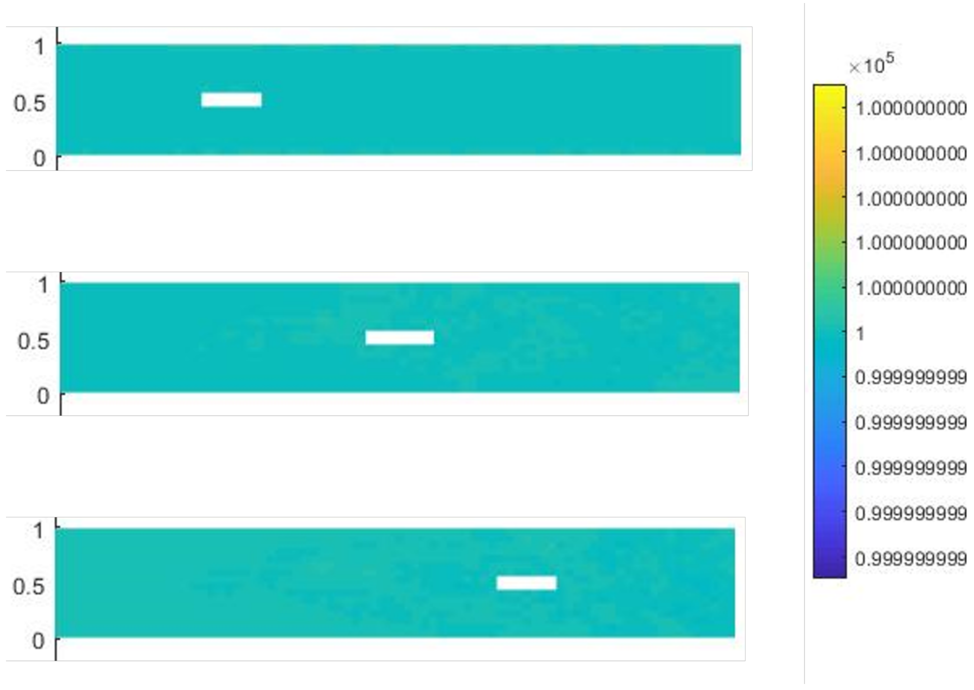


Figure 5.49 Pressure distributions on reference grid at initial position, mid-trajectory, and at the end of trajectory. The position of the hole changes with the mobile grid motion.

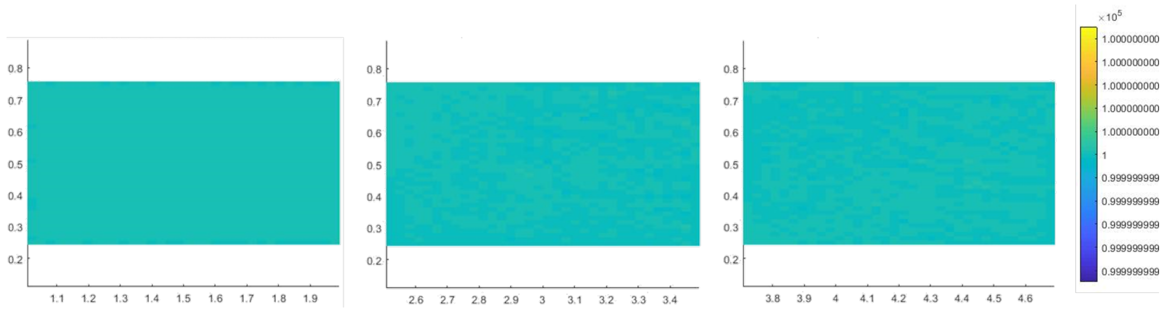


Figure 5.50 Pressure distributions on overset grid at initial position, mid-trajectory, and at the end of trajectory.

5.7 Unsteady flows on overset grids

This category of tests verifies the ability of the developed IBM algorithm to fulfill the required research objective: The ability to solve unsteady compressible flow over a stationary and moving bodies using overset grids.

5.7.1 Unsteady supersonic flow over a wedge with stationary overset grid (CAT41)

The test aims to verify the unsteady flow over a symmetric stationary wedge immersed on an overset grid. The wedge dimensions and operating conditions are the same as CAT23 test, presented in Section 5.5.3. The test verifies the solution on overset grids with respect to the solution of the same conditions and geometry preformed on a single grid in CAT23. Moreover, it verifies the overset reconstruction for the case of perfectly overlapped grids (grids of same cell size, with identical overlap cell center position).

The dimensions of the reference grid are identical to that of CAT23 (3 $\times$ 2 m). The overset grid is placed at 0.5 m far from the domain inlet along the center line of the computational domain, its dimension is (2 $\times$ 1 m), where the topology is centered inside the overset grid. This configuration is illustrated in Fig.5.51

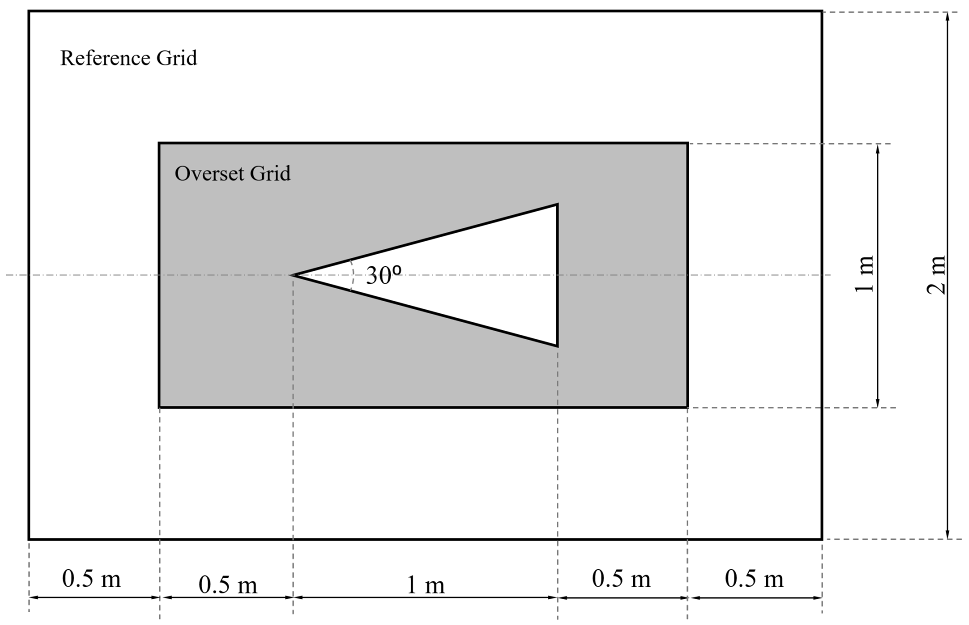
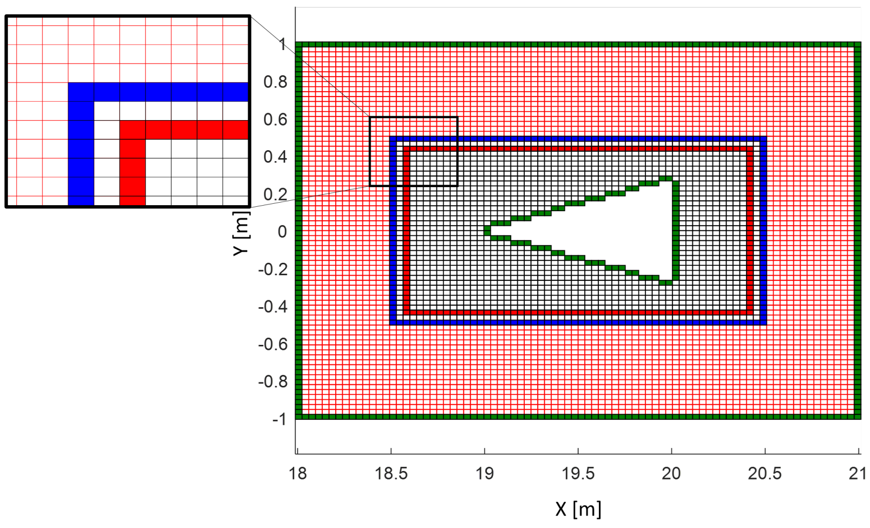
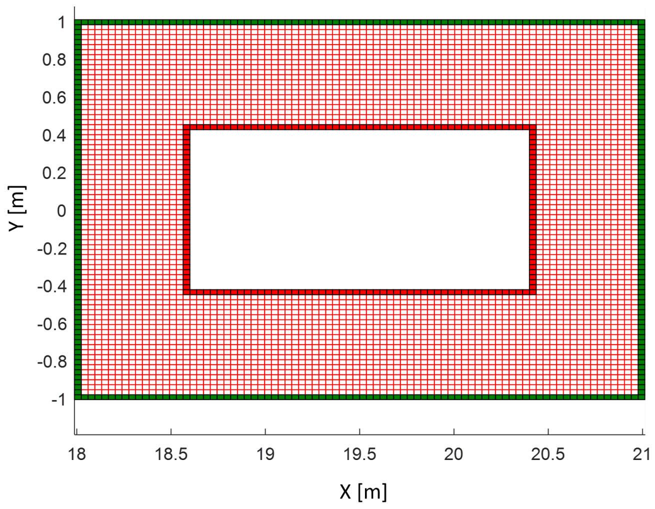


Figure 5.51 Computational domain dimensions for a symmetric wedge on an overset grid.

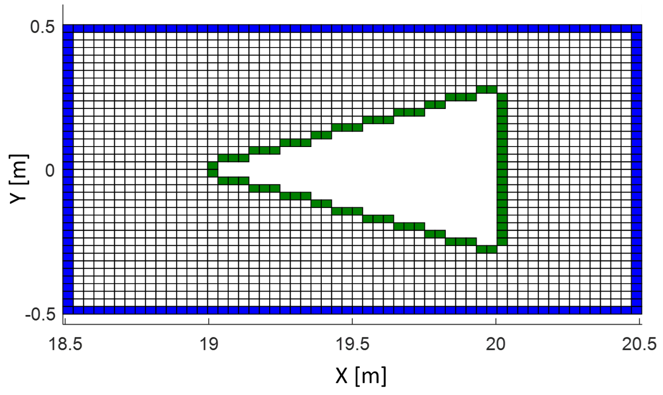
The grid resolution of the reference grid that of CAT23 test (90×80). The overset grid resolution is 60×40, which generates the same grid size (dx, dy) as the reference grid. The overset grid is placed on the reference grid so that the cells are perfectly overlapped, so that the overset grid is seen as the complementary of the reference grid with zero overlap between the two grids. The generated grids with the tagging are illustrated in Fig. 5.52. In Fig. 5.52a the transition between the two grids is zoomed, to verify that the cells are perfectly aligned.



(a) Overset grids with perfect alignment at grid boundaries.



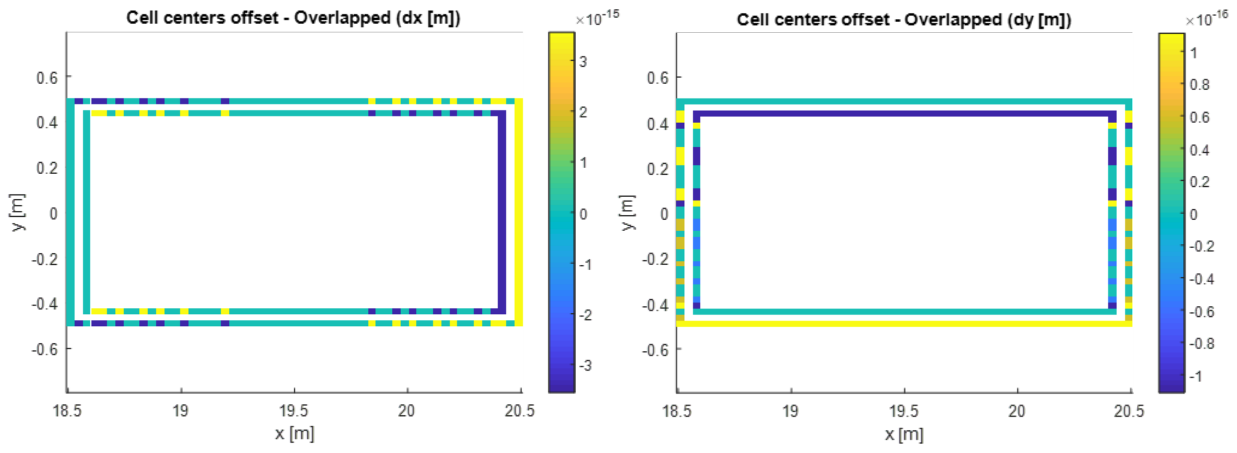
(b) Reference grid with hole cutting. The inner red cells layer are the interpolated cells, the outer green cells layer are interface cells. Grid resolution (90×80).



(c) Overset grid. The outer blue cells layer are interpolated cells, the inner green cells layer are interface cells. Grid resolution (60×40).

Figure 5.52 Computational domain with perfectly aligned overset grids.

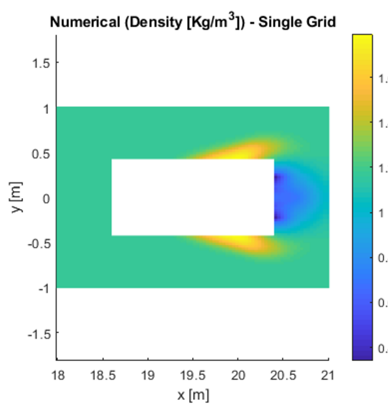
In order to assess the overset grid alignment, the offset between the interpolated cell centers with the corresponding cells in the opposite grid is evaluated, and illustrated in Fig. 5.58. The outer layer in this figure is the offset of the interpolated cells of the overset grid with its corresponding Inner Cells on the reference grid. While, the inner layer is the interpolated cells offset with the corresponding Inner Cells of the overset grid. The figure shows a precise overlap with an offset error of machine zero, which ensures a perfect alignment of the two grids.



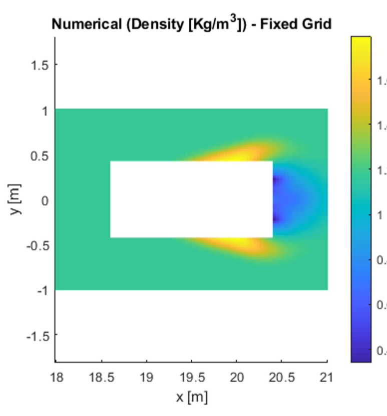
(a) Grid offset - dx. (b) Grid offset - dy.

Figure 5.53 Grid offset of interpolated cell centers.

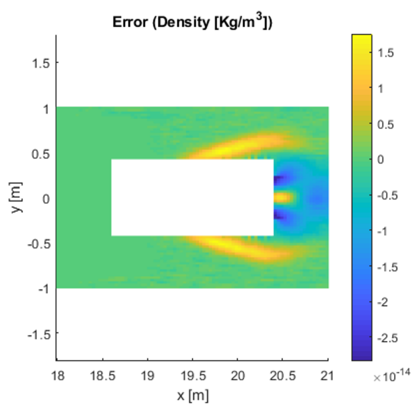
Since the generated overset grid is identical to the single grid presented in the CAT23 test. The comparison between the two solutions could be accomplished $vis - \grave{a} - vis$, such that, each grid is compared to its corresponding region located on the single grid. The comparison between a single grid versus an overset grid solution is illustrated in Figs. 5.54-5.57 after 100 time steps solution. The error between the overset solution and the single grid solution is near machine zero for all flow properties at all time steps. It is important to note is that the solver setup for the two cases has to be carried out with the same solver parameters (CFL, number of time steps) to ensure that the solutions can be always compared at the same time level.



(a) Density distribution on single grid.

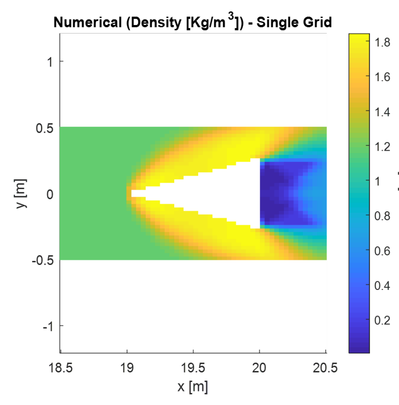


(b) Density distribution on reference grid - Reference part.

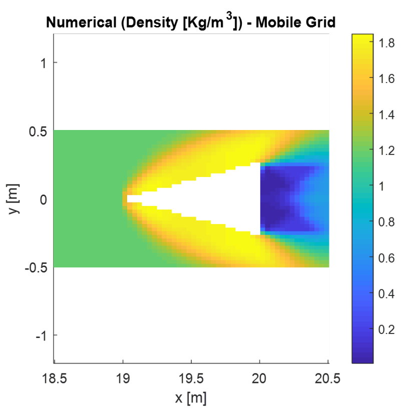


(c) Density error between single and reference grids.

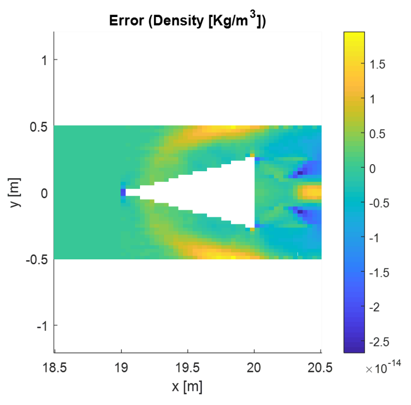
Figure 5.54 Density distribution on single and reference grids, $n_t = 100$.



(a) Density distribution on single grid - Overset part.

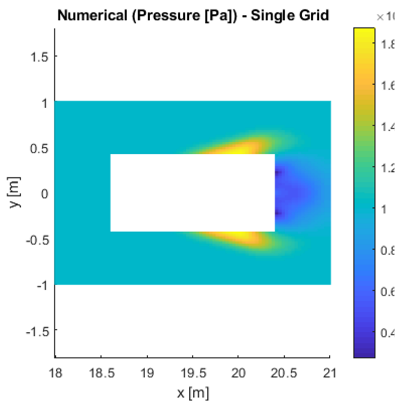


(b) Density distribution on overset grid.

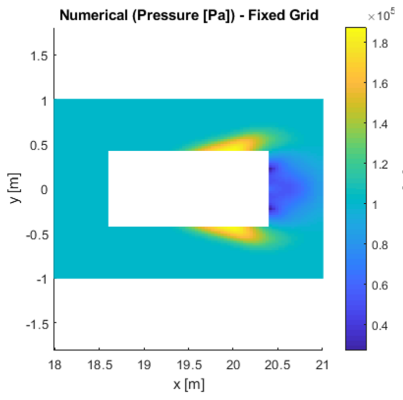


(c) Density error between single and overset grids.

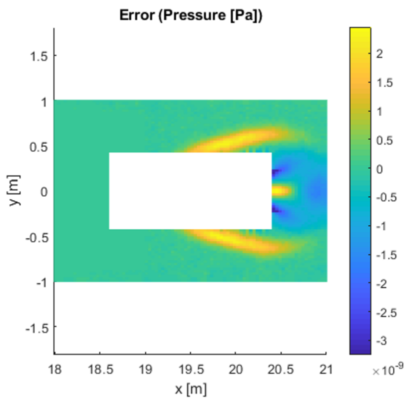
Figure 5.55 Density distribution on single and overset Cartesian grids, $n_t = 100$.



(a) Pressure distribution on single grid.

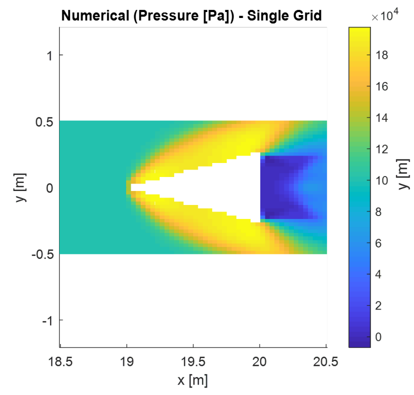


(b) Pressure distribution on reference grid - Reference part.

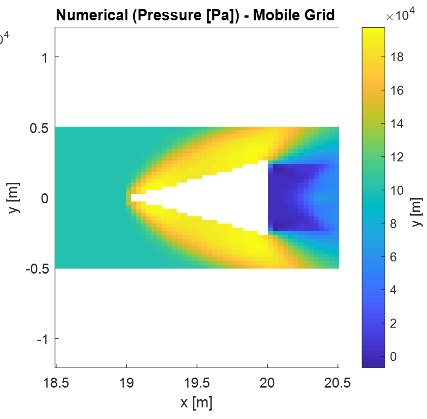


(c) Pressure error between single and reference grids.

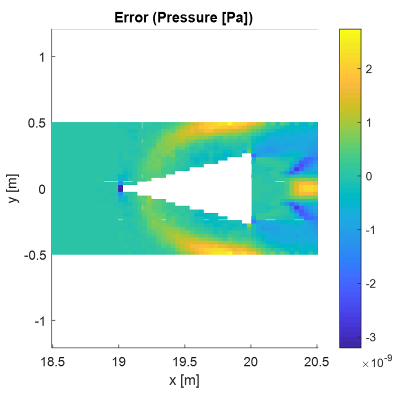
Figure 5.56 Pressure distribution on single and reference girds, $n_t = 100$.



(a) Pressure distribution on single grid - Overset part.



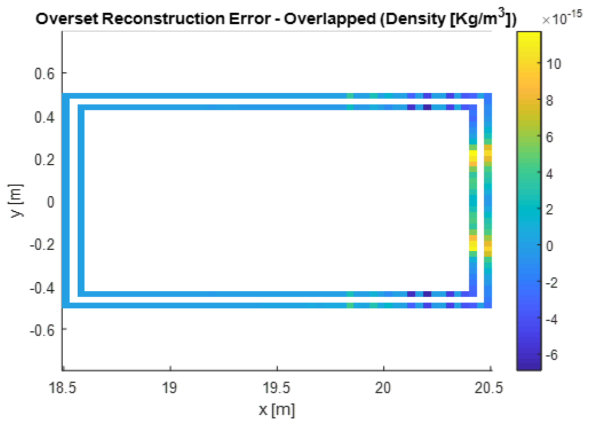
(b) Pressure distribution on overset grid.



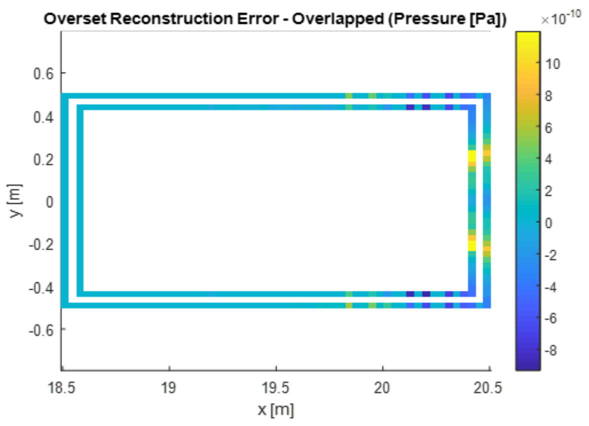
(c) Pressure error between single and overset grids.

Figure 5.57 Pressure distribution on single and overset Cartesian girds, $n_t = 100$.

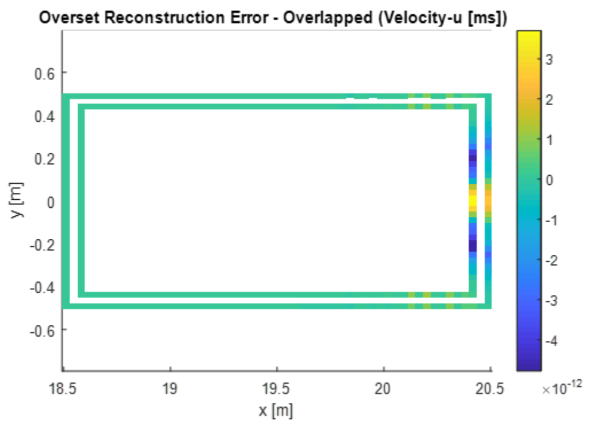
From results it can be concluded that the adopted overset approach does not impact the solution if the overset grids are perfectly aligned, *i.e.*, the overset reconstruction is exact. To demonstrates the error introduced by the overset reconstruction, the solution on the interpolated interface of the two grids is analyzed and illustrated in Fig. 5.58. The overset reconstruction error is of the order of machine zero in the case of perfectly aligned cells, which agrees the analytical synthesis of the reconstruction scheme presented in Section 4.4.



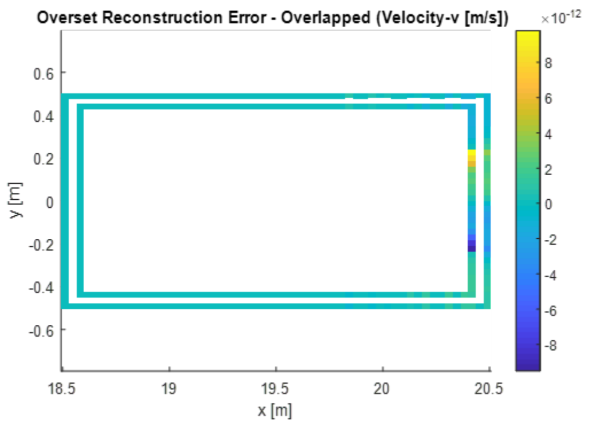
(a) Density error.



(b) Pressure error.



(c) u-velocity error.



(d) v-velocity error.

Figure 5.58 Overset reconstruction error of flow properties. Outer loop: overset grid. Inner loop: reference grid.

5.7.2 Unsteady supersonic flow over a wedge with moving overset grid (CAT42)

This test aims to demonstrate the capability of the developed code to simulate moving geometries in compressible flows using the proposed IBM-Overset-ALE integration. The test consists of an immersed topology of a symmetric wedge, the same wedge geometry used in previous tests, immersed in an overset grid. This overset grid will move with constant velocity that corresponds to Mach 2 inside a 21 m long tube, filled with a stagnant gas. The test case setup is shown in Fig. 5.59.

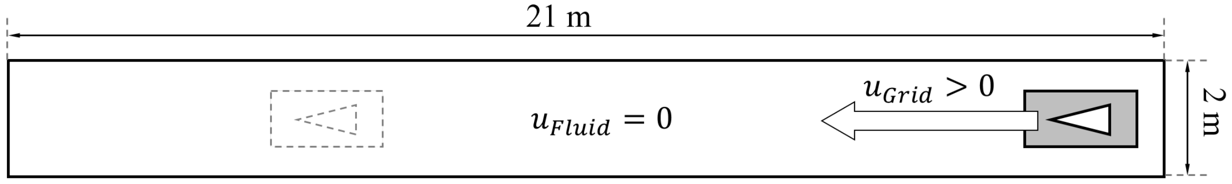


Figure 5.59 Computational domain of moving overset supersonic tube.

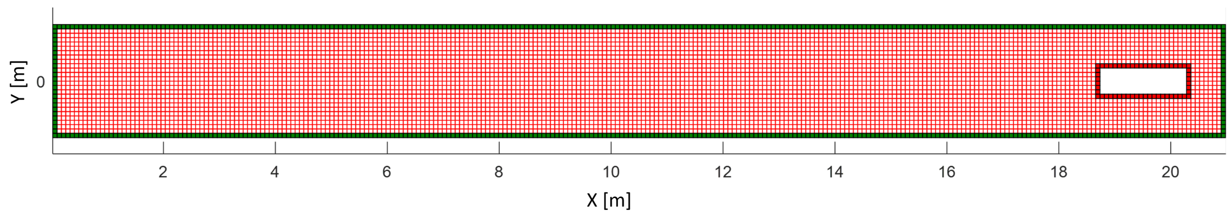
Two grid resolutions are tested, (630×80, and 1260×160) for reference grid discretization, and (60×40, and 120×80) for overset grid discretization. The implementation has been realized with *MatLab* software, and the grid refinement is limited by the computational resources available and by the computation time required for the finer grid. The coarser grids are illustrated in Fig. 5.60.

The test case operating conditions are listed in Table 5.8.

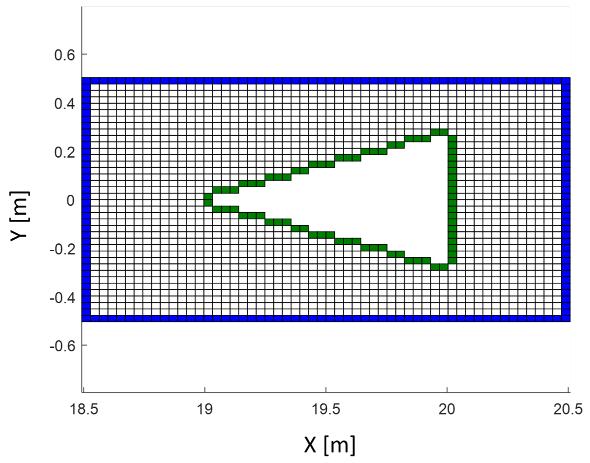
Table 5.8 Operating conditions of moving supersonic wedge on overset grid

Parameter	Value
p	100 KPa
M_{Fluid}	0
T	300 K
α	0 deg
u_{Grid}	-694.377 m/s
v_{Grid}	0 m/s

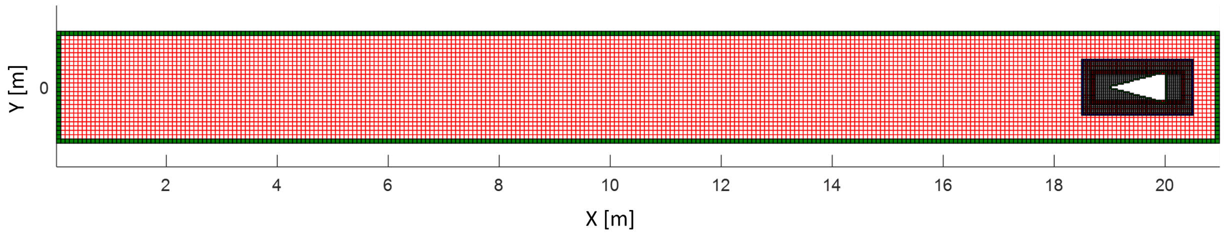
The unsteady evolution of flow field around the moving wedge is shown in Fig.5.61 at different times. The shock strength developed at the leading edge of the wedge becomes stronger, so that after 2500 time steps the shock is fully developed. At this stage, the flow simulation time is 0.089 seconds.



(a) Reference grid discretization (630 × 80), the hole corresponds to the initial position of overset grid.



(b) Mobile grid discretization (60 × 40).



(c) Assembled overset grids.

Figure 5.60 Computational domain discretization at initial position $t = 0$.

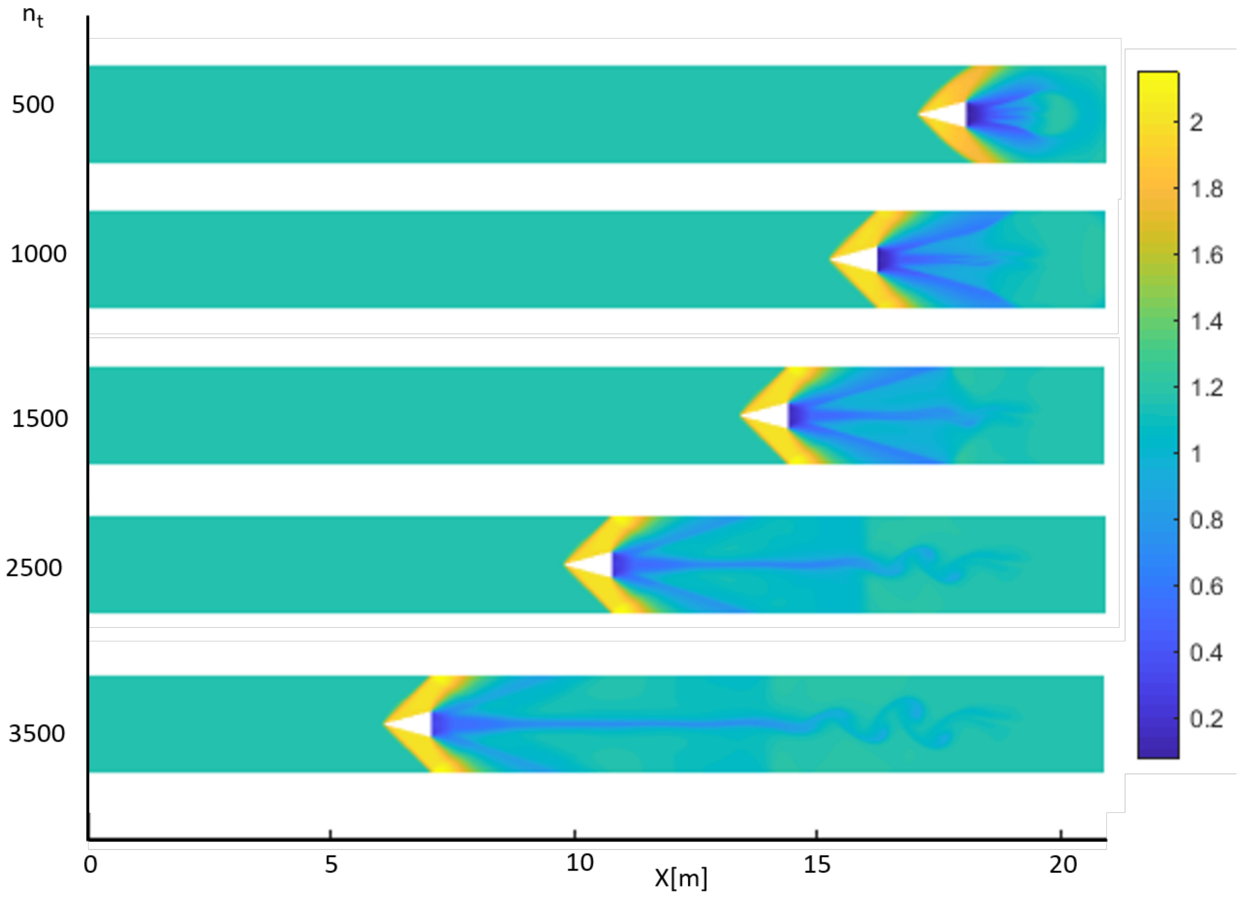


Figure 5.61 Unsteady density distribution of supersonic moving wedge at Mach 2, grid resolution $1260 \times 160/\ 120 \times 80$ at different time steps.

5.7.3 Unsteady supersonic flow over three moving accelerated wedges on overset grids (CAT43)

This test illustrates the ability of the developed code to manage multiple grids in accelerated motion for arbitrary trajectories. It consists of three independent symmetric wedges immersed in the center of an (2×1 m) overset grid, aligned initially at the same position ($x = 0$). Each overset grid is defined by its trajectory and velocity profile along a reference grid. The designed trajectories do not allow the overset grids to overlaps on each other. The mobile grid trajectories and the corresponding velocity profiles are shown in Figs. 5.62 and 5.63.

The computational domain consists of a reference grid of dimensions (15×8 m), discretized by 315×180, and the overlapping grids are discretized by 60×40, as illustrated in Fig. 5.64. The overlapping grids are initially aligned at $x = 0$, referred to the overlapping grid centers. The middle overlapping grid is centered vertically to the reference grid, and each grid distant

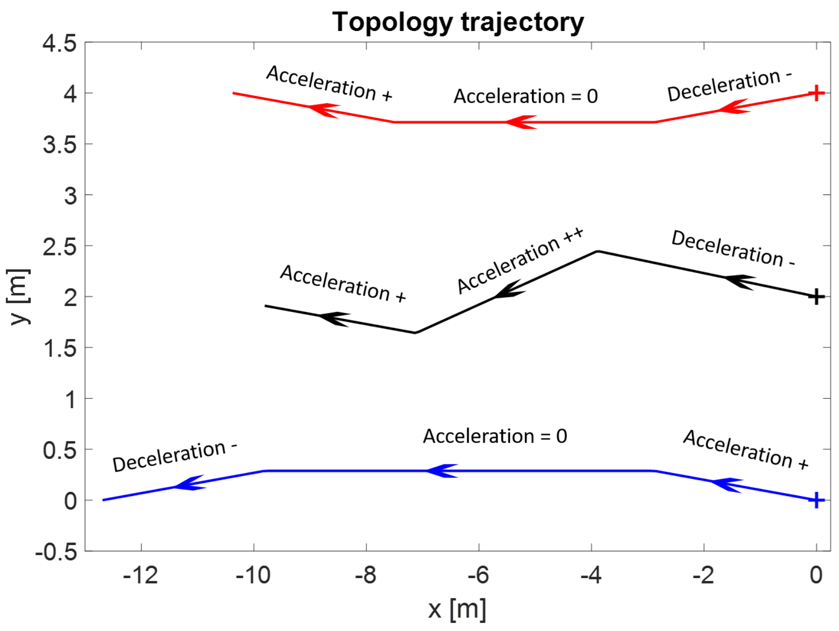
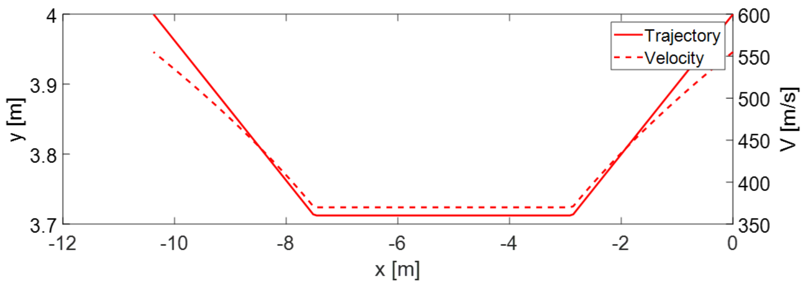


Figure 5.62 Overset grids trajectories for three wedges, aligned initially at $x = 0$.

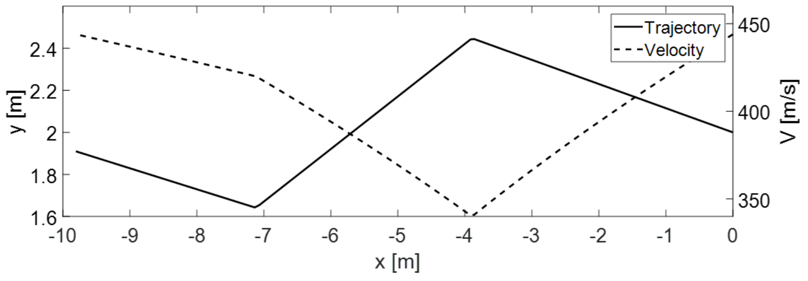
1 m apart.

The grid velocity components (u_{grid} , v_{Grid}) are determined by the decomposition of the velocity magnitude to the Cartesian coordinate given by the slope of the trajectory at any grid position.

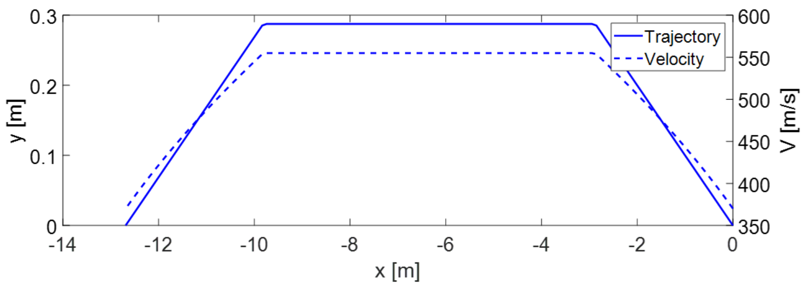
A set of sample frames from the unsteady solution are illustrated in Fig. 5.65. The density distribution at $n_t = 200$ shows a wave interference between the middle and the bottom wedge, which becomes more pronounced at $n_t = 300$. At $n_t = 800$ a strong shock begin to propagates from the lower wall of the upper wedge, where a shock reflection occurs between the upper and the middle wedge that was identified afterward at (n_t>800). The interpretation of the flow field is based on the observation of the results, which can be verified in details later in future work. The goal of this test is to show the capabilities to manage multiple grids with non-uniform motion in compressible flow conditions, and the preliminary results agree with the physical comprehension of the presented phenomena.



(a) Upper wedge.

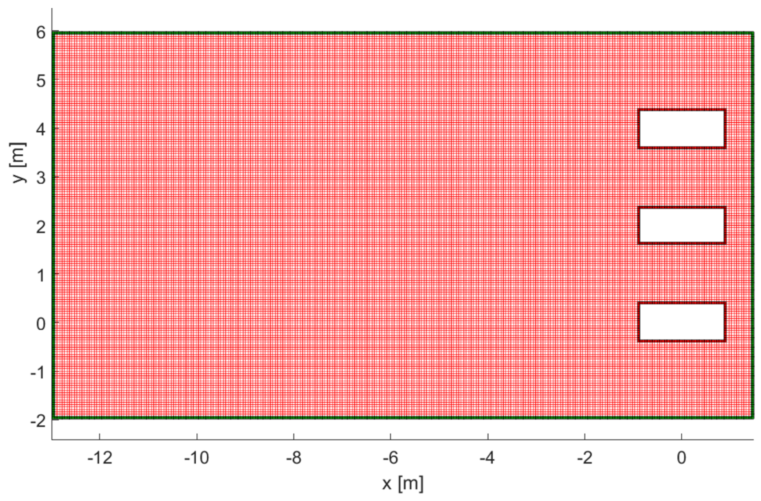


(b) Middle wedge.

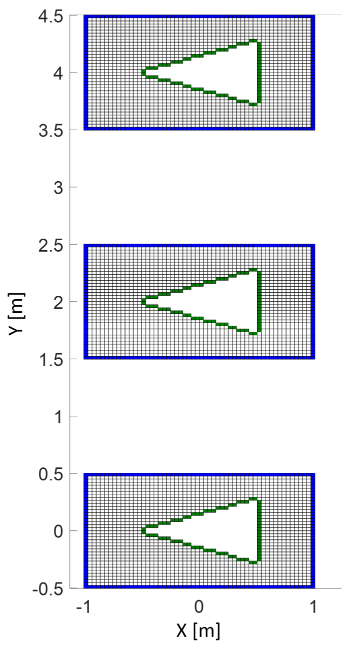


(c) Lower wedge.

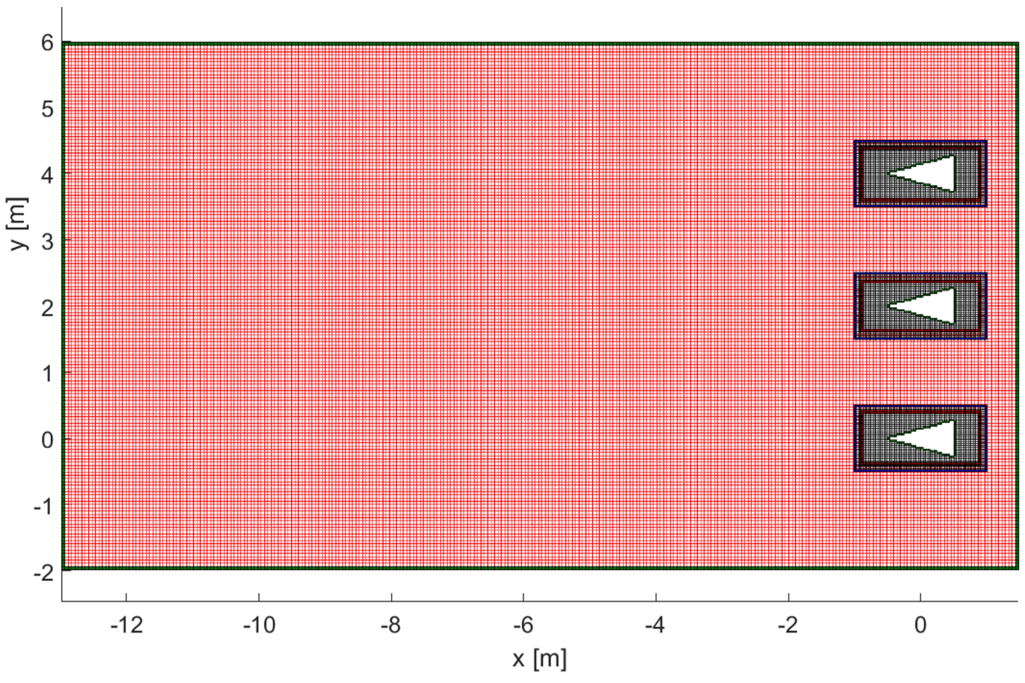
Figure 5.63 Trajectory and corresponding velocity profiles for the three moving wedges



(a) Reference grid (315 × 180).

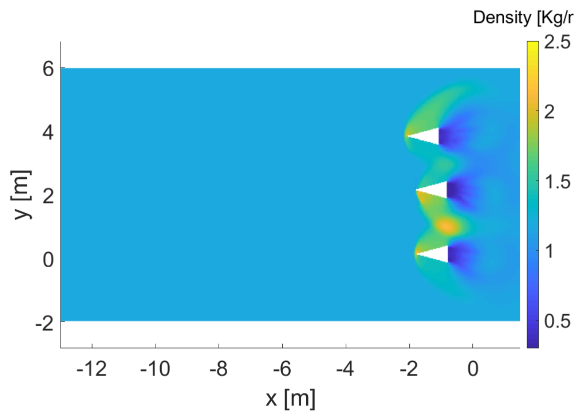


(b) Overlapping grids (60 × 40).

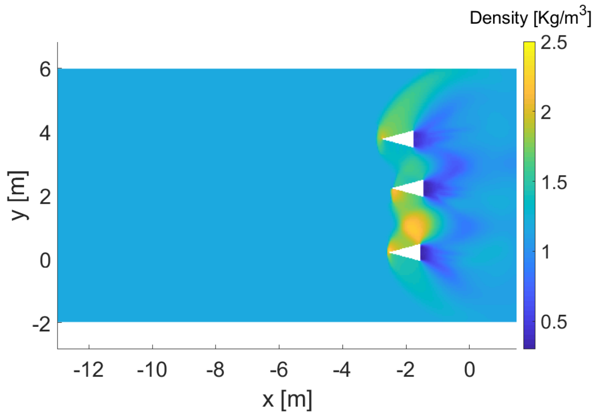


(c) Assembled overset grids.

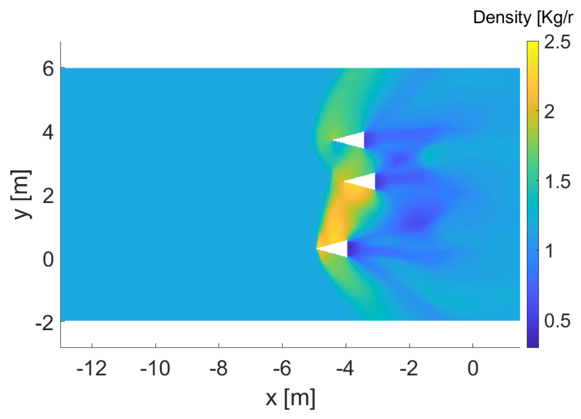
Figure 5.64 Computational overset domain at $t = 0$.



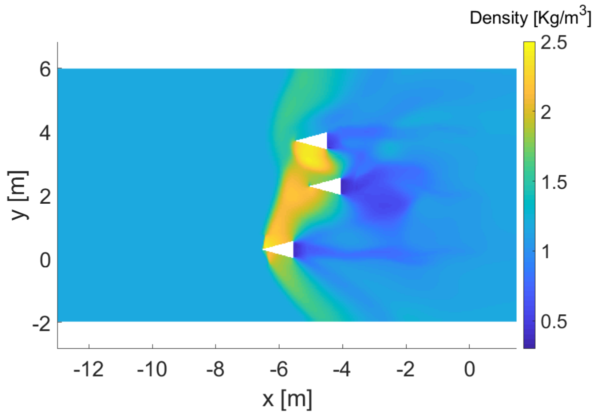
(a) $n_t = 200$.



(b) $n_t = 300$.

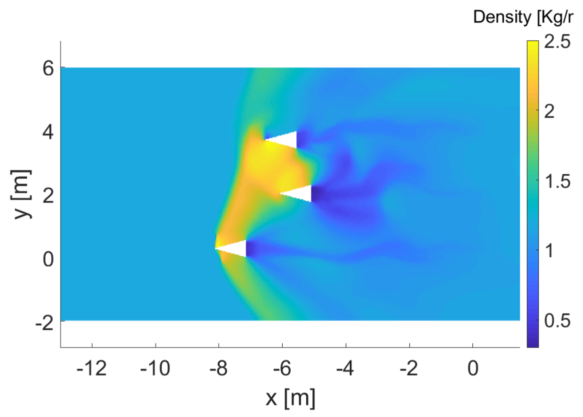


(c) $n_t = 600$.

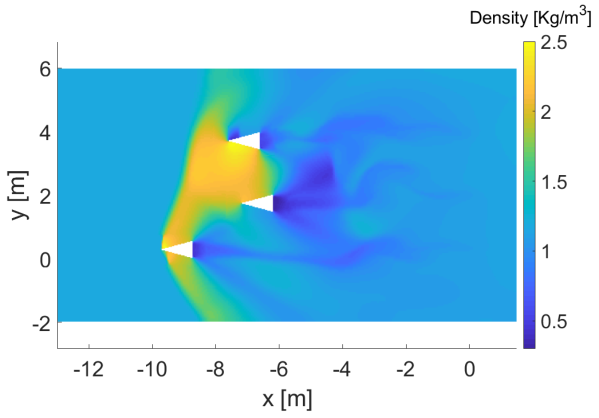


(d) $n_t = 800$.

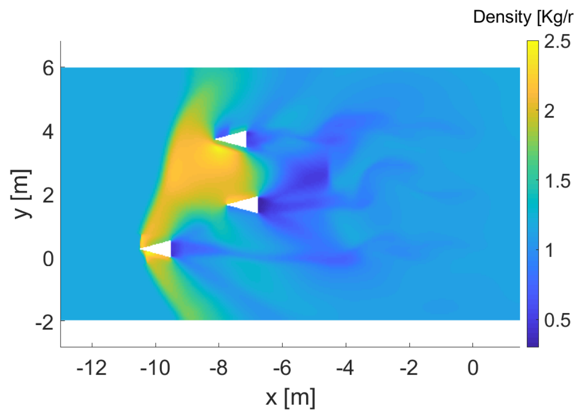
Figure 5.65 Unsteady density distribution of an accelerated three wedges.



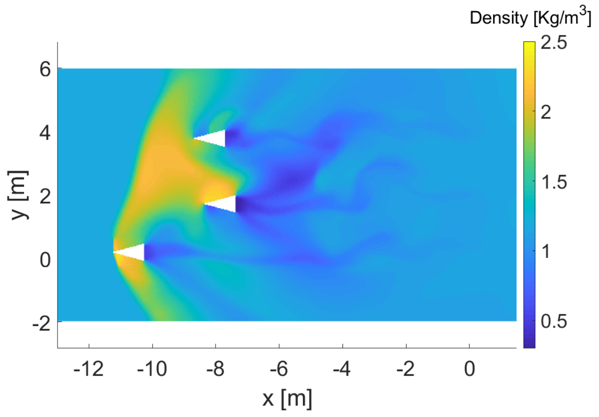
(e) $n_t = 1000$.



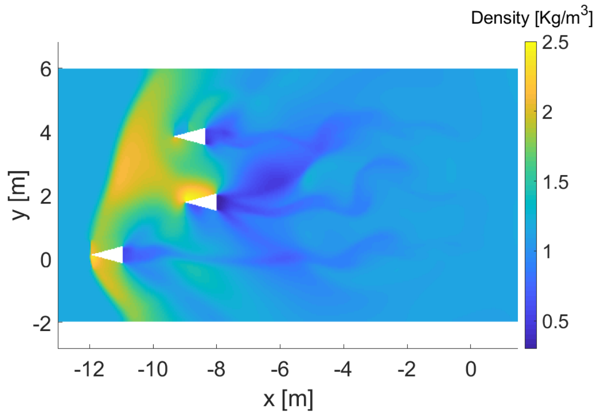
(f) $n_t = 1200$.



(g) $n_t = 1300$.



(h) $n_t = 1400$.



(i) $n_t = 1500$.

Figure 5.65 Unsteady density distribution of an accelerated three wedges.

CHAPTER 6 CONCLUSION

This chapter summarizes the presented work, discusses the limitation of the proposed solution, and suggests future work that could enhance the proposed methodology.

6.1 Summary of work

This work introduced the development of a 2D algorithm for compressible flow simulation over stationary and moving bodies using Immersed Boundary Method integrated with hierarchical overset grids, as the first step towards 3D extended version with a multi-physics solver based on the IBM-Overset approach.

The development methodology and theoretical representation of each building block of the algorithm (Discrete topology, Hierarchical grid, Mesh tagging, Overset tagging, Refinement criteria/grid controls, Solution reconstruction, and Flow solver) have been successfully developed and presented. A defined format for the discrete topology representation is generated. A linear quad-tree data structure is employed to generate hierarchical grids. The adopted spatial numbering scheme is used to leverage the computational efficiency of hierarchical grid generation and management.

A robust tagging approach, based on "*geometry marching*" showed a capability to tag different geometrical configurations in accordance with hierarchical refinements to tag complex geometries. The tagging class is responsible to create the link between the immersed geometry and the hierarchical grid, as it identifies the different types of cells inside the computational domain. In addition, it controls the grid refinement through various criteria, and other grid management features. The tagging algorithm succeeded in tagging different topologies with different configurations and can support complex geometric scenarios without ambiguity, as the tagging of the trailing edge of an airfoil (NACA0012). The difficulty of this case is that the geometric line enters and exits from the same cell side, which is regarded as a complex tagging case and was found not supported by some libraries as `AMRex`.

The overset module has shown the ability to manage overlapped grids, under the posed hypothesis of always having the entire overset grid inside the reference grid, with no interference between any overset-to-overset grid. This module is able to perform the necessary grid adaptations (Overset tagging and hole cutting) efficiently for stationary and mobile cases. The integration of Cartesian grids with the overset module simplifies its implementation, where the procedure of the hole cutting stays as simple as finding the four corners of a rectangle that bounds the hole. The challenge that comes with this integration relies with the tagging of interpolated cells, when it shares the characteristic of both (interface & interpolated cells) at the same time. This situation takes place when the outer boundary of the overlapped grid (interpolated layer) coincide an the interface cell boundary on the reference grid, and it needs to be considered for further investigation. For the purpose of this work, the implemented test cases avoids this configuration.

An implicit least square reconstruction scheme is introduced to reconstruct the solution along the interface cell layers and the overset cell layers. The proposed scheme has been verified analytically and numerically to reconstruct Dirichlet and Neumann boundary conditions linked to the flow solver.

An inviscid compressible cell-centered finite volume flow solver is implemented, based on a first order discretization in space and time of Roe's scheme. A set of test cases are reported to demonstrate the ability of the developed algorithm to simulate steady/unsteady compressible flow cases for stationary/moving bodies. The results show good agreement with the available analytical data, as well as the expected order of convergence of the numerical scheme. The test cases span tests on steady/unsteady flows over single grids, stationary overset grids, uniformly moving overset grids, and accelerated overset grids.

This research succeeded to lay the foundation of a first version of a 2D fully IBM-Overset algorithm to simulate compressible flows with bodies in relative motion with a first order accuracy.

6.2 Limitations

The adopted numerical scheme has some limitations for the simulation of subsonic compressible flows, where the simulation time is very long, specially in steady state. The adopted numerical scheme is more efficient for high Mach number flows.

The motion of overset grids in the presented work is limited to translation motion only. Rotation of moving bodies is not supported in the present work. Another constraint is posed for grid motion in that the moving grids are not allowed to overlap the reference grid boundary, or overlap each other, *i.e.* each grid is limited to its free of obstacle path.

The development of the present algorithm on *Matlab* environment puts another limitation on the computing performance, where this interpreted language limits the speed of computations, and put a limitation for a HPC.

The spatial and temporal schemes are of first order. A higher order schemes could be introduced to enhance the solution accuracy.

The reconstruction scheme employs a linear polynomial being in second order accuracy for Dirichlet reconstruction, and first order accuracy for Neumann reconstruction. A higher order polynomial degree could be implemented to achieve higher order of accuracy for the reconstructed values.

Finally, certain parts of the implemented algorithm could be optimized to improve the computational efficiency, and hence for parallelization compatibility.

6.3 Future research

- The implementation of higher order discretization for spatial and temporal numerical scheme. In order to achieve higher order of accuracy for spatial scheme with attenuated oscillations within the discontinuities, several approaches are proposed from the literature. The Monotone Upstream-Centered Scheme for Conservation Laws (MUSCL) scheme can be used to construct the convective flux by higher order extrapolation, and allow the variation of the state variables within the control volume to capture the discontinuity. However, MUSCL schemes usually require a limiter to control the overshoots introduced at higher orders [123], hence the Total Variance Diminishing (TVD) scheme could be used as limiting scheme. In general, the implementation of limiters should be done with care to do not impact the solution accuracy. A MUSCL scheme with limiter was implemented with the IBM solver by [103]. Another high order schemes, extensively used for higher order implementation with IBM are the Essentially Non-Oscillatory (ENO) and Weighted Essentially Non-Oscillatory (WENO) schemes. These schemes are especially suitable for complex problems that include shock waves, shock interactions with complex flows over complex geometries [133]. These approaches are implemented with the IBM for compressible flow problems by [96, 100, 104, 113, 134]. The proposed explicit higher order temporal integration could be Runge-Kutta 4.

- The analysis and introduction of rotational motion for moving bodies. The analysis comprises the ALE approach to verify the Geometric Conservation Laws during the rotational motion.

- The integration of distance field solver to the overset module to support the overlap regions between grids. This distance field solver will be based on solving Eikonal field equation [135].

- Additional verification test cases could be conducted based on the Method of Manufactured Solution (MMS) and other tests found in the literature to assess the different aspects of the developed code.

- The transfer of the developed code into a compiled environment such as C++ to enhance the computing performance, and expand the capability for 3D extension.

REFERENCES

[1] D. W. Newsletter, “Saab gripen e conducts missile firing, fuel tank drop tests,” 2018. [Online]. Available: https://www.defenseworld.net/news/23573/Saab_Gripen_E_Conducts_Missile_Firing___Fuel_Tank_Drop_Tests

[2] A. Cenko *et al.*, “Integrated T&E Approach to Store Separation – Dim Past, Exciting Future,” pp. 541–551, Sep. 1996.

[3] A. Viviani, G. Pezzella, and E. DAmato, “Aerodynamic Analysis with Separation Dynamics of a Launcher at Staging Conditions,” in *30th Congress of the International Council of the Aeronautical Sciences*, Daejeon, Korea, 2016.

[4] D. J. Dalle *et al.*, “Inviscid and Viscous CFD Analysis of Booster Separation for the Space Launch System Vehicle,” in *54th AIAA Aerospace Sciences Meeting.* San Diego, California, USA: American Institute of Aeronautics and Astronautics, Jan. 2016.

[5] C. Srikanth and C. Bhasker, “Flow analysis in valve with moving grids through CFD techniques,” *Advances in Engineering Software*, vol. 40, no. 3, pp. 193–201, Mar. 2009.

[6] X. Song *et al.*, “A CFD analysis of the dynamics of a direct-operated safety relief valve mounted on a pressure vessel,” *Energy Conversion and Management*, vol. 81, pp. 407–419, May 2014.

[7] R. Gomez *et al.*, “STS-107 Investigation Ascent CFD Support,” in *34th AIAA Fluid Dynamics Conference and Exhibit.* Portland, Oregon: American Institute of Aeronautics and Astronautics, Jun. 2004.

[8] L. Peng *et al.*, “Overset structured grids assembly method for numerical simulations of multi-bodies and moving objects,” *Computers & Fluids*, vol. 175, pp. 260–275, Oct. 2018.

[9] W.-X. Huang and F.-B. Tian, “Recent trends and progress in the immersed boundary method,” *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 233, no. 23-24, pp. 7617–7636, Dec. 2019.

[10] 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

[11] M. de Tullio *et al.*, “An immersed boundary method for compressible flows using local grid refinement,” *Journal of Computational Physics*, vol. 225, no. 2, pp. 2098–2117, Aug. 2007.

[12] P. De Palma *et al.*, “An immersed-boundary method for compressible viscous flows,” *Computers & Fluids*, vol. 35, no. 7, pp. 693–702, Aug. 2006.

[13] R. Abgrall, H. Beaugendre, and C. Dobrzynski, “An immersed boundary method using unstructured anisotropic mesh adaptation combined with level-sets and penalization techniques,” *Journal of Computational Physics*, vol. 257, pp. 83–101, Jan. 2014.

[14] M. E. Khalili, M. Larsson, and B. Müller, “High-order ghost-point immersed boundary method for viscous compressible flows based on summation-by-parts operators: High-order immersed boundary method for viscous compressible flows,” *International Journal for Numerical Methods in Fluids*, vol. 89, no. 7, pp. 256–282, Mar. 2019.

[15] 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.

[16] I. Borazjani *et al.*, “A parallel overset-curvilinear-immersed boundary framework for simulating complex 3D incompressible flows,” *Computers & Fluids*, vol. 77, pp. 76–96, Apr. 2013.

[17] A. A. Osman *et al.*, “Numerical Analysis of an External Store Separation From an Airplane,” in *AIAA Modeling and Simulation Technologies Conference.* San Diego, California, USA: American Institute of Aeronautics and Astronautics, Jan. 2016.

[18] O. Mahmood, J. Masud, and Z. G. Toor, “Trajectory Simulation of a Standard Store and Generic Wing Pylon using CFD,” in *2018 AIAA Aerospace Sciences Meeting.* Kissimmee, Florida: American Institute of Aeronautics and Astronautics, Jan. 2018.

[19] Y. Jiang, A. Valdiero, and P. Andrighetto, “Analysis of pneumatic directional proportional valve with CFX mesh motion technique,” *ABCM Sympos Serial Mechatr*, vol. 3, pp. 510–518, 2008.

[20] E. Atta, “Component-adaptive grid interfacing,” in *19th Aerospace Sciences Meeting.* St. Louis,MO,U.S.A.: American Institute of Aeronautics and Astronautics, Jan. 1981.

[21] E. H. Atta and J. Vadyak, “A grid interfacing zonal algorithm for three-dimensional transonic flows about aircraft configurations,” in *Eighth International Conference on*

Numerical Methods in Fluid Dynamics, E. Krause, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1982, vol. 170, pp. 107–114.

[22] J. Benek, J. Steger, and F. Dougherty, "A flexible grid embedding technique with application to the Euler equations," in *6th Computational Fluid Dynamics Conference Danvers*. Danvers,MA,U.S.A.: American Institute of Aeronautics and Astronautics, Jul. 1983.

[23] J. Hooker and J. Gudenkauf, "Application of the Unstructured Chimera Method for Rapid Weapons Trajectory Simulations," in *45th AIAA Aerospace Sciences Meeting and Exhibit*. Reno, Nevada: American Institute of Aeronautics and Astronautics, Jan. 2007.

[24] T. J. Flora *et al.*, "Dynamic Store Release of Ice Models from a Cavity into Mach 2.9 Flow," *Journal of Aircraft*, vol. 51, no. 6, pp. 1927–1941, Nov. 2014.

[25] L. Xuefei, L. Yuan, and Q. Zhansen, "Applications of Overset Grid Technique to CFD Simulation of High Mach Number Multi-body Interaction/Separation Flow," *Procedia Engineering*, vol. 99, pp. 458–476, 2015.

[26] W. Wang *et al.*, "An efficient, robust and automatic overlapping grid assembly approach for partitioned multi-block structured grids," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 233, no. 4, pp. 1217–1236, Mar. 2019.

[27] A. Khaware *et al.*, "Numerical Simulation of Store Separation Trajectories for EGLIN Test Case using Overset Mesh," in *2018 AIAA Aerospace Sciences Meeting*. Kissimmee, Florida: American Institute of Aeronautics and Astronautics, Jan. 2018.

[28] R. Meakin and N. Suhs, "Unsteady aerodynamic simulation of multiple bodies in relative motion," in *9th Computational Fluid Dynamics Conference*. Buffalo,NY,U.S.A.: American Institute of Aeronautics and Astronautics, Jun. 1989.

[29] S. E. Rogers, D. J. Dalle, and W. M. Chan, "CFD Simulations of the Space Launch System Ascent Aerodynamics and Booster Separation," in *53rd AIAA Aerospace Sciences Meeting*. Kissimmee, Florida: American Institute of Aeronautics and Astronautics, Jan. 2015.

[30] J. W. Purvis and J. E. Burkhalter, "Prediction of critical Mach number for store configurations," *AIAA Journal*, vol. 17, no. 11, pp. 1170–1177, Nov. 1979.

[31] B. Wedan and J. South, Jr., “A method for solving the transonic full-potential equation for general configurations,” in *6th Computational Fluid Dynamics Conference Danvers.* Danvers,MA,U.S.A.: American Institute of Aeronautics and Astronautics, Jul. 1983.

[32] D. K. Clarke, M. D. Salas, and H. A. Hassan, “Euler calculations for multielement airfoils using Cartesian grids,” *AIAA Journal*, vol. 24, no. 3, pp. 353–358, Mar. 1986.

[33] R. Gaffney, Jr. and H. Hassan, “Euler calculations for wings using Cartesian grids,” in *25th AIAA Aerospace Sciences Meeting.* Reno,NV,U.S.A.: American Institute of Aeronautics and Astronautics, Mar. 1987.

[34] M. J. Aftosmis, M. J. Berger, and J. E. Melton, “Robust and Efficient Cartesian Mesh Generation for Component-Based Geometry,” *AIAA Journal*, vol. 36, no. 6, pp. 952–960, Jun. 1998.

[35] M. Nemec, M. Aftosmis, and M. Wintzer, “Adjoint-Based Adaptive Mesh Refinement for Complex Geometries,” in *46th AIAA Aerospace Sciences Meeting and Exhibit.* Reno, Nevada: American Institute of Aeronautics and Astronautics, Jan. 2008.

[36] M. W. Johnson, “A novel Cartesian CFD cut cell approach,” *Computers & Fluids*, vol. 79, pp. 105–119, Jun. 2013.

[37] M. Yousuf *et al.*, “Demonstration of Automated CFD Process using Mesh-less Technology,” in *6th European Conference on Computational Fluid Dynamics (ECFD VI)*, Barcelona, Spain, Jul. 2014.

[38] C. S. Peskin, “Flow patterns around heart valves: A numerical method,” *Journal of Computational Physics*, vol. 10, no. 2, pp. 252–271, Oct. 1972.

[39] Y. Kim and C. S. Peskin, “Penalty immersed boundary method for an elastic boundary with mass,” *Physics of Fluids*, vol. 19, no. 5, p. 053103, May 2007.

[40] R. Mittal and G. Iaccarino, “IMMERSED BOUNDARY METHODS,” *Annual Review of Fluid Mechanics*, vol. 37, no. 1, pp. 239–261, Jan. 2005.

[41] Y. Jianming, “Sharp interface direct forcing immersed boundary methods: A summary of some algorithms and applications,” *Journal of Hydrodynamics*, vol. 28, no. 5, pp. 713–730, Oct. 2016.

[42] D. Goldstein, R. Handler, and L. Sirovich, “Modeling a No-Slip Flow Boundary with an External Force Field,” *Journal of Computational Physics*, vol. 105, no. 2, pp. 354–366, Apr. 1993.

[43] W. Kim and H. Choi, "Immersed boundary methods for fluid-structure interaction: A review," *International Journal of Heat and Fluid Flow*, vol. 75, pp. 301–309, Feb. 2019.

[44] J. Wu and C. Shu, "Implicit velocity correction-based immersed boundary-lattice Boltzmann method and its applications," *Journal of Computational Physics*, vol. 228, no. 6, pp. 1963–1979, Apr. 2009.

[45] Y. Qiu *et al.*, "A boundary condition-enforced immersed boundary method for compressible viscous flows," *Computers & Fluids*, vol. 136, pp. 104–113, Sep. 2016.

[46] J. R. Edwards *et al.*, "An Immersed Boundary Method for General Flow Applications," in *ASME 2010 3rd Joint US-European Fluids Engineering Summer Meeting: Volume 1, Symposia – Parts A, B, and C.* Montreal, Quebec, Canada: ASMEDC, Jan. 2010, pp. 2461–2469.

[47] H. Luo *et al.*, "On the numerical oscillation of the direct-forcing immersed-boundary method for moving boundaries," *Computers & Fluids*, vol. 56, pp. 61–76, Mar. 2012.

[48] F.-B. Tian *et al.*, "Fluid–structure interaction involving large deformations: 3d simulations and applications to biological systems," *Journal of Computational Physics*, vol. 258, pp. 451–469, Feb. 2014.

[49] S. Manoorkar *et al.*, "Suspension flow through an asymmetric T-junction," *Journal of Fluid Mechanics*, vol. 844, pp. 247–273, Jun. 2018.

[50] E. Stavropoulos Vasilakis *et al.*, "Cavitation induction by projectile impacting on a water jet," *International Journal of Multiphase Flow*, vol. 114, pp. 128–139, May 2019.

[51] A. Nasar *et al.*, "Eulerian weakly compressible smoothed particle hydrodynamics (SPH) with the immersed boundary method for thin slender bodies," *Journal of Fluids and Structures*, vol. 84, pp. 263–282, Jan. 2019.

[52] A. Piquet, O. Roussel, and A. Hadjadj, "A comparative study of Brinkman penalization and direct-forcing immersed boundary methods for compressible viscous flows," *Computers & Fluids*, vol. 136, pp. 272–284, Sep. 2016.

[53] J. Mohd-Yusof, "Combined immersed-boundary / B-spline methods for simulations of flow in complex geometries," pp. 317–327, 1997.

[54] E. Fadlun *et al.*, "Combined Immersed-Boundary Finite-Difference Methods for Three-Dimensional Complex Flow Simulations," *Journal of Computational Physics*, vol. 161, no. 1, pp. 35–60, Jun. 2000.

[55] F. Sotiropoulos and X. Yang, "Immersed boundary methods for simulating fluid–structure interaction," *Progress in Aerospace Sciences*, vol. 65, pp. 1–21, Feb. 2014.

[56] B. Khalighi, S. Jindal, and G. Iaccarino, "Aerodynamic flow around a sport utility vehicle—Computational and experimental investigation," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 107-108, pp. 140–148, Aug. 2012.

[57] C. S. Peskin and D. M. McQueen, "A three-dimensional computational method for blood flow in the heart I. Immersed elastic fibers in a viscous incompressible fluid," *Journal of Computational Physics*, vol. 81, no. 2, pp. 372–405, Apr. 1989.

[58] 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.

[59] Z. Wang *et al.*, "A 2^N Tree Based Automated Viscous Cartesian Grid Methodology For Feature Capturing," 1999.

[60] 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.

[61] A. Dadone and B. Grossman, "Ghost-Cell Method with far-field coarsening and mesh adaptation for Cartesian grids," *Computers & Fluids*, vol. 35, no. 7, pp. 676–687, Aug. 2006.

[62] 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.

[63] Y. Cho, J. Chopra, and P. Morris, "Immersed Boundary Method for Compressible High-Reynolds Number Viscous Flow around Moving Bodies," in *45th AIAA Aerospace Sciences Meeting and Exhibit*. Reno, Nevada: American Institute of Aeronautics and Astronautics, Jan. 2007.

[64] A. Dadone and B. Grossman, "Ghost-cell method for analysis of inviscid three-dimensional flows on Cartesian-grids," *Computers & Fluids*, vol. 36, no. 10, pp. 1513–1528, Dec. 2007.

[65] K. Karagiozis, R. Kamakoti, and C. Pantano, "A low numerical dissipation immersed interface method for the compressible Navier–Stokes equations," *Journal of Computational Physics*, vol. 229, no. 3, pp. 701–727, Feb. 2010.

[66] D. Hartmann, M. Meinke, and W. Schröder, "A strictly conservative Cartesian cut-cell method for compressible viscous flows on adaptive grids," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 9-12, pp. 1038–1052, Feb. 2011.

[67] J. H. Seo and R. Mittal, "A high-order immersed boundary method for acoustic wave scattering and low-Mach number flow-induced sound in complex geometries," *Journal of Computational Physics*, vol. 230, no. 4, pp. 1000–1019, Feb. 2011.

[68] M. Berger and M. Aftosmis, "Progress Towards a Cartesian Cut-Cell Method for Viscous Compressible Flow," in *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition.* Nashville, Tennessee: American Institute of Aeronautics and Astronautics, Jan. 2012.

[69] J. P. Johnson *et al.*, "High Reynolds Number Airfoil Simulations Using the Immersed Boundary Method," in *Volume 1: Symposia, Parts A and B.* Rio Grande, Puerto Rico, USA: American Society of Mechanical Engineers, Jul. 2012, pp. 1359–1368.

[70] R. E. Harris, "Adaptive Cartesian Immersed Boundary Method for Simulation of Flow over Flexible Geometries," *AIAA Journal*, vol. 51, no. 1, pp. 53–69, Jan. 2013.

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

[72] 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.

[73] P. H. Tran and F. Plourde, "Computing compressible internal flows by means of an Immersed Boundary Method," *Computers & Fluids*, vol. 97, pp. 21–30, Jun. 2014.

[74] F. Capizzano and E. Iuliano, "A Eulerian Method for Water Droplet Impingement by Means of an Immersed Boundary Technique," *Journal of Fluids Engineering*, vol. 136, no. 4, p. 040906, Apr. 2014.

[75] C. Brehm, C. Hader, and H. Fasel, "A locally stabilized immersed boundary method for the compressible Navier–Stokes equations," *Journal of Computational Physics*, vol. 295, pp. 475–504, Aug. 2015.

[76] F. Capizzano, “Coupling a Wall Diffusion Model with an Immersed Boundary Technique,” *AIAA Journal*, vol. 54, no. 2, pp. 728–734, Feb. 2016.

[77] C. Chi, B. J. Lee, and H. G. Im, “An improved ghost-cell immersed boundary method for compressible flow simulations: AN IMPROVED GHOST-CELL IMMERSED BOUNDARY METHOD,” *International Journal for Numerical Methods in Fluids*, vol. 83, no. 2, pp. 132–148, Jan. 2017.

[78] 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.

[79] C. Zhu, H. Luo, and G. Li, “High-Order Immersed-Boundary Method for Incompressible Flows,” *AIAA Journal*, vol. 54, no. 9, pp. 2734–2741, Sep. 2016.

[80] K. Luo *et al.*, “A ghost-cell immersed boundary method for the simulations of heat transfer in compressible flows under different boundary conditions Part-II: Complex geometries,” *International Journal of Heat and Mass Transfer*, vol. 104, pp. 98–111, Jan. 2017.

[81] Y. Tamaki, M. Harada, and T. Imamura, “Near-Wall Modification of Spalart–Allmaras Turbulence Model for Immersed Boundary Method,” *AIAA Journal*, vol. 55, no. 9, pp. 3027–3039, Sep. 2017.

[82] B. Muralidharan and S. Menon, “Simulation of moving boundaries interacting with compressible reacting flows using a second-order adaptive Cartesian cut-cell method,” *Journal of Computational Physics*, vol. 357, pp. 230–262, Mar. 2018.

[83] R. Yuan and C. Zhong, “An immersed-boundary method for compressible viscous flows and its application in the gas-kinetic BGK scheme,” *Applied Mathematical Modelling*, vol. 55, pp. 417–446, Mar. 2018.

[84] P. J. Frey and P. L. George, *Mesh generation: application to finite elements*, 2nd ed. Hoboken, NJ: Wiley [u.a.], 2008, oCLC: 836701426.

[85] W. Zhang *et al.*, “AMReX: a framework for block-structured adaptive mesh refinement,” *Journal of Open Source Software*, vol. 4, no. 37, p. 1370, May 2019.

[86] J. Bell *et al.*, “BoxLib User Guide,” Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory, Berkeley, CA, Technical Report, 2013.

[87] M. Adams *et al.*, "Chombo Software Package for AMR Applications Design Document," Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA, Technical Report, Dec. 2015.

[88] P. Durbin and G. Iaccarino, "An Approach to Local Refinement of Structured Grids," *Journal of Computational Physics*, vol. 181, no. 2, pp. 639–653, Sep. 2002.

[89] 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.

[90] 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.

[91] 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, Mar. 2013.

[92] M. Ehsan Khalili, M. Larsson, and B. Müller, "Immersed boundary method for viscous compressible flows around moving bodies," *Computers & Fluids*, vol. 170, pp. 77–92, Jul. 2018.

[93] L. Sun, S. R. Mathur, and J. Y. Murthy, "An Unstructured Finite-Volume Method for Incompressible Flows with Complex Immersed Boundaries," *Numerical Heat Transfer, Part B: Fundamentals*, vol. 58, no. 4, pp. 217–241, Sep. 2010.

[94] P.-j. Ming *et al.*, "Unstructured grid immersed boundary method for numerical simulation of fluid structure interaction," *Journal of Marine Science and Application*, vol. 9, no. 2, pp. 181–186, Jun. 2010.

[95] P. Ouro *et al.*, "An immersed boundary method for unstructured meshes in depth averaged shallow water models," *International Journal for Numerical Methods in Fluids*, vol. 81, no. 11, pp. 672–688, Aug. 2015.

[96] 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.

[97] 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.

[98] X. Hu *et al.*, "A conservative interface method for compressible flows," *Journal of Computational Physics*, vol. 219, no. 2, pp. 553–578, Dec. 2006. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0021999106001926

[99] O. Hu, N. Zhao, and J. Liu, "A Ghost Cell Method for Turbulent Compressible Viscous Flows on Adaptive Cartesian Grids," *Procedia Engineering*, vol. 67, pp. 241–249, 2013. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1877705813018560

[100] 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.

[101] J. O'Rourke, *Computational Geometry in C*, 2nd ed. Cambridge University Press, Oct. 1998.

[102] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations," *Journal of Computational Physics*, vol. 79, no. 1, pp. 12–49, Nov. 1988.

[103] Y. Gorsse *et al.*, "A simple second order cartesian scheme for compressible Euler flows," *Journal of Computational Physics*, vol. 231, no. 23, pp. 7780–7794, Oct. 2012.

[104] C. Liu and C. Hu, "An immersed boundary solver for inviscid compressible flows," *International Journal for Numerical Methods in Fluids*, vol. 85, no. 11, pp. 619–640, Dec. 2017.

[105] Y. Mizuno *et al.*, "A Simple Immersed Boundary Method for Compressible Flow Simulation around a Stationary and Moving Sphere," *Mathematical Problems in Engineering*, vol. 2015, pp. 1–17, 2015.

[106] S. Takahashi, T. Nonomura, and K. Fukuda, "A Numerical Scheme Based on an Immersed Boundary Method for Compressible Turbulent Flows with Shocks: Application to Two-Dimensional Flows around Cylinders," *Journal of Applied Mathematics*, vol. 2014, pp. 1–21, 2014.

[107] 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.

[108] H. Mo *et al.*, "An immersed boundary method for solving compressible flow with arbitrarily irregular and moving geometry: Immersed boundary method for flow with irregular and moving geometry," *International Journal for Numerical Methods in Fluids*, vol. 88, no. 5, pp. 239–263, Oct. 2018.

[109] R. Ramakrishnan, A. Girdhar, and S. Ghosh, "Immersed boundary methods for compressible laminar flows," 01 2016, pp. 7029–7044.

[110] 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, Jan. 2016.

[111] G. Iaccarino and R. Verzicco, "Immersed boundary technique for turbulent flow simulations," *Applied Mechanics Reviews*, vol. 56, no. 3, pp. 331–347, May 2003.

[112] Y. He *et al.*, "An Immersed Boundary Method Based on Volume Fraction," *Procedia Engineering*, vol. 99, pp. 677–685, 2015.

[113] 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, Feb. 2018.

[114] H. Udaykumar, R. Mittal, and W. Shyy, "Computation of Solid–Liquid Phase Fronts in the Sharp Interface Limit on Fixed Grids," *Journal of Computational Physics*, vol. 153, no. 2, pp. 535–574, Aug. 1999.

[115] J. Yang and E. Balaras, "An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries," *Journal of Computational Physics*, vol. 215, no. 1, pp. 12–40, Jun. 2006.

[116] H. Udaykumar *et al.*, "A Sharp Interface Cartesian Grid Method for Simulating Flows with Complex Moving Boundaries," *Journal of Computational Physics*, vol. 174, no. 1, pp. 345–380, Nov. 2001.

[117] S. Péron *et al.*, "A mixed overset grid/immersed boundary approach for CFD simulations of complex geometries," in *54th AIAA Aerospace Sciences Meeting*. San Diego, California, USA: American Institute of Aeronautics and Astronautics, Jan. 2016.

[118] J. R. Aarnes *et al.*, “Treatment of solid objects in the Pencil Code using an immersed boundary method and overset grids,” *Geophysical & Astrophysical Fluid Dynamics*, vol. 114, no. 1-2, pp. 35–57, Mar. 2020.

[119] K. Aizawa and S. Tanaka, “A Constant-Time Algorithm for Finding Neighbors in Quadtrees,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 7, pp. 1178–1183, Jul. 2009. [Online]. Available: http://ieeexplore.ieee.org/document/4538229/

[120] S. W. and A. A., “Cardinal Neighbor Quadtree: a New Quadtree-based Structure for Constant-Time Neighbor Finding,” *International Journal of Computer Applications*, vol. 132, no. 8, pp. 22–30, Dec. 2015. [Online]. Available: http://www.ijcaonline.org/research/volume132/number8/qasem-2015-ijca-907501.pdf

[121] H. Samet, “Neighbor finding in images represented by octrees,” *Computer Vision, Graphics, and Image Processing*, vol. 46, no. 3, pp. 367–386, Jun. 1989. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0734189X89900388

[122] R. Meakin, “Object X-rays for cutting holes in composite overset structured grids,” in *15th AIAA Computational Fluid Dynamics Conference*, ser. Fluid Dynamics and Co-located Conferences. American Institute of Aeronautics and Astronautics, Jun. 2001. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2001-2537

[123] J. Blazek, *Computational fluid dynamics: principles and applications*, 1st ed. Amsterdam ; New York: Elsevier, 2001.

[124] G. Dahlquist, B. Sj{öberg, and P. Svensson, “Comparison of the method of averages with the method of least squares.” *Mathematics of Computation*, vol. 22, no. 104, pp. 833–833, Jan. 1968. [Online]. Available: http://www.ams.org/jourcgi/jour-getitem?pii=S0025-5718-1968-0239742-X

[125] F. M. White, *Fluid mechanics*, 4th ed., ser. McGraw-Hill series in mechanical engineering. Boston, Mass: WCB/McGraw-Hill, 1999.

[126] S. Arabi, J.-Y. Trépanier, and R. Camarero, “A simple extension of Roe’s scheme for multi-component real gas flows,” *Journal of Computational Physics*, vol. 388, pp. 178–194, Jul. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0021999119301901

[127] W. L. Oberkampf and C. J. Roy, *Verification and Validation in Scientific Computing.* Cambridge: Cambridge University Press, 2010.

[128] R.-H. Ni, “A multiple grid scheme for solving the Euler equations,” in *5th Computational Fluid Dynamics Conference*, ser. Fluid Dynamics and Co-located Conferences. American Institute of Aeronautics and Astronautics, Jun. 1981. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.1981-1025

[129] A. Madadi, M. Kermani, and H. Khazaei, “Improvement of a solution of inviscid compressible flows using a mixed wall boundary condition,” *Engineering Applications of Computational Fluid Mechanics*, vol. 9, no. 1, pp. 126–138, Jan. 2015. [Online]. Available: http://www.tandfonline.com/doi/full/10.1080/19942060.2015.1004815

[130] “Procedure for Estimation and Reporting of Uncertainty Due to Discretization in CFD Applications,” *Journal of Fluids Engineering*, vol. 130, no. 7, p. 078001, 2008. [Online]. Available: http://FluidsEngineering.asmedigitalcollection.asme.org/article.aspx?articleid=1434171

[131] E. F. Toro, *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*, 3rd ed. Dordrecht ; New York: Springer, 2009, oCLC: ocn401321914.

[132] J. D. Anderson and J. D. Anderson, *Fundamentals of aerodynamics*, 5th ed., ser. Anderson series. New York: McGraw-Hill, 2011, oCLC: ocn463634144.

[133] C.-W. Shu, “Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws,” in *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations*, A. Quarteroni, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, vol. 1697, pp. 325–432.

[134] T. Ershova *et al.*, “Numerical Simulation of Heterogeneous Flows and Heat-Mass Transfer in Complex Domains on Rectangular Grids,” in *2010 14th International Heat Transfer Conference, IHTC 14*, vol. 1, 2010.

[135] H. Xia and P. G. Tucker, “Finite volume distance field and its application to medial axis transforms,” *International Journal for Numerical Methods in Engineering*, vol. 82, no. 1, pp. 114–134, 2010. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2762

APPENDIX A ALGORITHM IMPLEMENTATION

In this section, the implementation of the developed code is presented in more details. The architecture of the code is object-oriented and its Unified Modeling Language (UML) diagram is illustrated in Fig. A.2.

The algorithm is built basically on seven main classes, namely, `DiscreteTopo`, `HierarchicalGrid`, `MeshTagging`, `OversetGrid`, `GridControls`, `Reconstruction`, and `FlowSolver`. Three supplementary classes are implemented to facilitate data handling and visualization, such as, `MobileDiscreteTopo`, `BoundaryConditions`, and `Solution` classes. The type of all implemented classes is *handle*, where the instances of all classes refer to the class object, with no duplicate data copies.

The discrete topology is managed by a parent class `DiscreteTopo` that reads the topology **.tdt* file and instantiates the topology elements (points, sides, loops and faces). A subclass `MobileDiscreteTopo` is derived from the parent class `DiscreteTopo` to manage moving topology, where the topology trajectory is defined within this class. This subclass is a composition of the `OversetGrid` class that passes the spatial information of the moving topology to the overset grid.

Hierarchical grids are generated by the parent class `HierarchicalGrid`. This class manages the hierarchical grid generation through its fundamental methods, including elements generation, neighbor localization, siblings finding, parent cell finding, cell refinement, and grid balancing. The grid is defined by its border extents (x_{min}, x_{max}, y_{min}, y_{max}) and its initial Cartesian resolution (N_x, N_y). Grid element is defined by its lower-left (x_1,y_1) and upper-right vertices (x_2,y_2) , level of refinement, level difference in cardinal directions ΔL_d, and local index (i,j). The hierarchical grid is stored in hash-table or container using a built-in library in *Matlab* called *containers.Map*. This type of data structure facilitates the grid management, where each `cellID` contains its own information within the hash-table. A sample of the stored data in grid hash-table is illustrated in Table A.1 and Fig. A.1.

Table A.1 Hash-table information of a given key (`cellID`) in hierarchical grid.

`cellID`	cell Level	x1	x2	y1	y2	ΔL_W	ΔL_E	ΔL_S	ΔL_N	i	j

The tagging algorithm is implemented through the `MeshTagging` class which is a subclass of `HierarchicalGrid` and inherits all its functionality and attributes to perform the tagging process, flood fill, and grid controls. Grid controls is executed by the `GridControls` class,

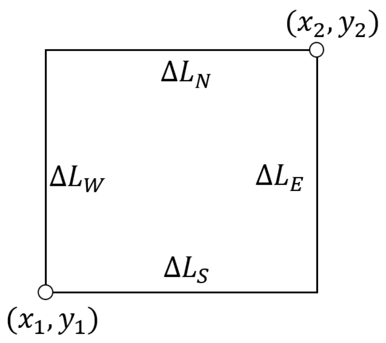


Figure A.1 Hierarchical grid element definition.

as a part of `MeshTagging` by an aggregation relationship. The four classes `DiscreteTopo`, `HierarchicalGrid`, `MeshTagging`, and `GridControls` represent the computational grid constituted by the tagged immersed boundary and the adapted hierarchical grid.

The tagging is performed for each computational face and stored in hash-table, with a key-value of `faceID`. Each key-value in the hash-table returns an array of `cellID` integers of the corresponding type. This tagging information are presented in Table A.2

Table A.2 Hash-table tagging information of a given key `faceID`.

`faceID`	Intersected Interface Cells Non-Intersected Interface Cells Face Cells Interface Cells Inside Cells Intersected Outside Cells Outside Cells

Boundary conditions are managed by the `BoundaryConditions` class, where the imposed boundary condition value and type are linked to their corresponding interface cells that represent the boundary. The reconstruction class, `Reconstruction`, employs the boundary conditions and tagging information to reconstruct flow solution in Interface Cells and Interpolated Cells, as part of the flow solver.

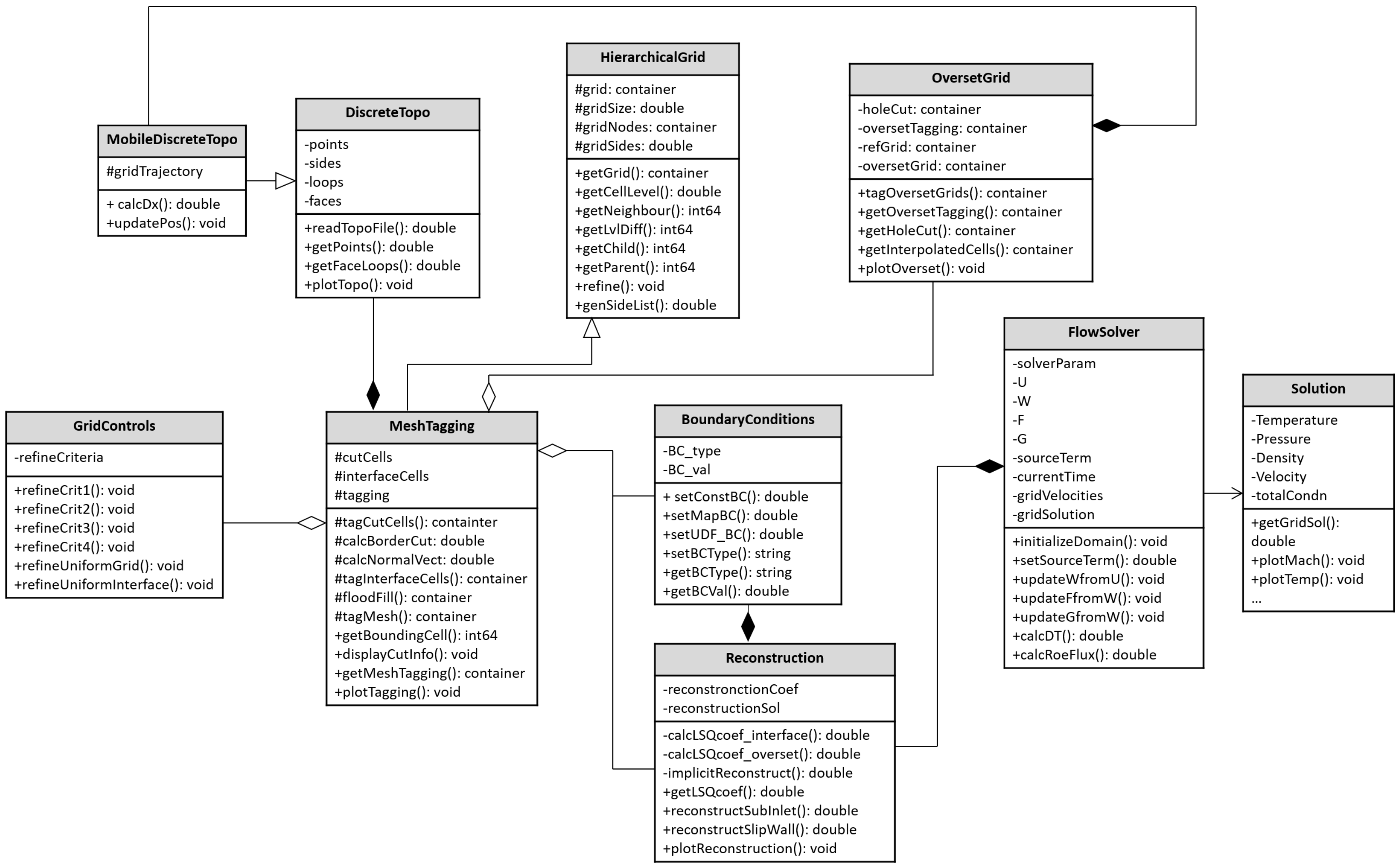


Figure A.2 UML diagram of the developed code