**POLYTECHNIQUE MONTRÉAL**
affiliée à l'Université de Montréal

# Conservative Immersed Boundary Method for Three-Dimensional Compressible Flows Simulation in Complex Geometries

**EL HADJI ABDOU AZIZ NDIAYE**
Département de génie mécanique

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

Juin 2025

**POLYTECHNIQUE MONTRÉAL**
affiliée à l'Université de Montréal

Cette thèse intitulée :

**Conservative Immersed Boundary Method for Three-Dimensional Compressible Flows Simulation in Complex Geometries**

présentée par **El Hadji Abdou Aziz NDIAYE**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

**Huu Duc VO**, président
**Sébastien LECLAIRE**, membre et directeur de recherche
**Jean-Yves TRÉPANIER**, membre et codirecteur de recherche
**Renan DE HOLANDA SOUSA**, membre et codirecteur de recherche
**Roberto PAOLI**, membre
**Marlène SANJOSÉ**, membre externe

# DEDICATION

*To my parents,*

# ACKNOWLEDGEMENTS

I would like to express my profound gratitude to Prof. Sébastien Leclaire and Prof. Jean-Yves Trépanier for their invaluable guidance and constant support throughout this doctoral research. Their constructive feedback and encouragement, especially during challenging times, have been instrumental in helping me navigate the complexities of this project and achieve my research objectives. I am particularly thankful for the opportunity to assist with teaching Computational Fluid Dynamics (CFD) courses, an experience that substantially enriched my understanding of the subject while allowing me to share knowledge with students.

I acknowledge the essential financial support provided by General Electric (GE) Vernova for this research project. My sincere appreciation extends to the entire GE team, with particular thanks to Dr. Renan De Holanda Sousa and Dr. Philippe Robin-Jouan for their collaboration and insights. The visit to the GE facility in Lyon offered valuable perspective on the real-world applications relevant to this project and meaningfully enhanced my understanding of industrial requirements.

I also extend my sincere thanks to the current and former members of the IBM research group at Polytechnique Montréal. I am particularly grateful to Mr. Eddy Petro for his support during the development of the numerical methods presented in this thesis. My appreciation also goes to my friends and colleagues at Polytechnique for their camaraderie and encouragement, which made my time here enjoyable and fulfilling.

Finally, I express my deepest appreciation to my parents, siblings, and friends who have supported me throughout my studies. Your unwavering encouragement and belief in my abilities have been a constant source of motivation and I am truly grateful for your presence in my life. I also wish to acknowledge everyone who has contributed to my academic journey, whether directly or indirectly. Your kindness and support have been instrumental to the completion of this work.

# RÉSUMÉ

La simulation numérique des disjoncteurs à haute tension constitue une tâche complexe nécessitant la modélisation de divers phénomènes physiques, incluant entre autres de l'écoulement de fluide compressible, des phénomènes de transfert de chaleur par radiation, ainsi que des interactions avec des champs électromagnétiques. Ces simulations sont essentielles pour comprendre le comportement des disjoncteurs, mais elles présentent d'importants défis en raison du caractère multiphysique du problème et de la complexité des géométries en jeu. Afin de simplifier la génération de maillage, la Méthode des Frontières Immergées (IBM), associée à des maillages cartésiens, est employé pour transférer les difficultés de la génération de maillage vers le solveur numérique. Toutefois, des défis demeurent, notamment en ce qui concerne l'imposition adéquate des conditions aux limites tout en garantissant la conservation des grandeurs physiques (masse, quantité de mouvement et énergie) à l'intérieur du domaine de calcul. Cette recherche se concentre spécifiquement sur la résolution de l'écoulement de fluide compressible au sein des disjoncteurs, en considérant l'hypothèse d'un écoulement non-visqueux obéissant à la loi des gaz parfaits dans un domaine ne comportant pas de frontières mobiles.

L'objectif général est de développer des méthodes numériques garantissant la conservation des variables conservées tout en assurant une convergence à l'ordre deux des solutions numériques. Trois contributions majeures sont présentées dans cette thèse, chacune abordant un objectif spécifique permettant d'atteindre l'objectif général. La première contribution introduit des méthodes conservatives implémentées sur des maillages cartésiens de type cut-cells où la méthodologie IBM est appliquée pour imposer les conditions limites. Des cas tests sur des géométries bidimensionnelles sont réalisés pour démontrer l'effectivité de ces méthodes pour la conservation des propriétés physiques. La deuxième contribution se concentre sur l'extension de ces méthodes conservatives au deuxième ordre. Grâce à l'implémentation d'une reconstruction quadratique et semi-implicite pour l'imposition des conditions aux limites, une précision d'ordre deux est atteinte tout en évitant les problèmes de stabilité liées aux petites cellules coupées. Des études de convergence utilisant la Méthode des Solutions Manufacturées confirment la précision d'ordre deux de la méthode, tant dans le domaine intérieur qu'aux niveaux des régions de cellules coupées. Des tests numériques sur des géométries bidimensionnelles complexes, y compris une simulation d'un écoulement instationnaire à l'intérieur d'un disjoncteur simplifié, permettent de vérifier la robustesse et la précision des méthodes développées. Enfin, la troisième contribution présente l'implémentation d'un solveur IBM tridimensionnel basé sur une reconstruction semi-implicite des cellules frontières pour

l'imposition des conditions aux limites. Une triangulation de surface décrite à travers une structure de données hiérarchique est utilisée pour définir la géométrie du domaine de calcul. Cette géométrie est ensuite intégrée dans une grille cartésienne adaptative où les cellules sont raffinées en fonction de critères géométriques. Des cas tests numériques en trois dimensions sont réalisés pour vérifier la précision et la robustesse du solveur développé. Un cas test avec une géométrie de disjoncteur à haute tension a été simulé pour démontrer l'applicabilité de la méthode développée à des géométries complexes et à des conditions physiques plus élaborées.

Les méthodes développées dans cette thèse représentent une avancée notable dans la simulation numérique des disjoncteurs haute tension utilisant des maillages cartésien. Les perspectives de recherche incluent l'extension de ces approches pour prendre en compte les frontières mobiles et les autres phénomènes physiques, renforçant ainsi l'applicabilité industrielle de cette méthodologie pour la simulation de disjoncteurs réels.

# ABSTRACT

The numerical simulation of high-voltage circuit breakers (HVCBs) represents a complex task requiring the modeling of various physical phenomena, including compressible fluid flow, radiative heat transfer, and electromagnetic interactions. Such simulations are essential for understanding HVCB behavior but present significant challenges due to the multiphysics nature of the problem and the complex geometry of circuit breakers, which include moving components. To simplify the mesh generation process for these complex geometries, the Immersed Boundary Method (IBM) with Cartesian grids is employed. This approach shifts the complexity from the mesh generation to the numerical solver itself. However, some challenges remain, particularly in robustly enforcing boundary conditions while preserving conservation at the immersed boundaries. This research places a specific focus on fluid flow simulation within circuit breakers by assuming inviscid compressible flow of a perfect gas with stationary boundaries.

The primary objectives are to develop numerical methods that ensure conservation of fluid quantities (mass, momentum, and energy) while achieving second-order accuracy in numerical solutions. Three contributions are presented in this thesis, each addressing a specific aspect of the numerical methodology. The first contribution introduces conservative methods based on cut-cells where boundary conditions are imposed using the IBM approach. These methods effectively preserve conservation for all relevant physical quantities, as demonstrated through multiple two-dimensional test cases. The second contribution proposes a fully second-order accurate two-dimensional cut-cells method. Through implementation of a high-order semi-implicit approach for boundary condition enforcement, second-order accuracy is achieved while avoiding the stability issues typically associated with small cut-cells. Convergence studies using the Method of Manufactured Solutions confirm the second-order accuracy of the method both in the interior domain and at the immersed boundaries. Numerical tests on complex two-dimensional geometries, including a simulation of unsteady flow within a simplified HVCB model, illustrate the robustness and accuracy of the developed methods. Finally, the third contribution details the implementation of a robust IBM solver for three-dimensional geometries. The geometry is defined through a boundary representation approach with surface triangulation, and the mesh generation uses an adaptive Cartesian grid where grid cells are refined based on geometric criteria. Application of the semi-implicit method for boundary condition enforcement in three dimensions yields highly accurate numerical solutions. Verification using a simplified three-dimensional HVCB geometry demonstrates accurate results in realistic contexts involving sophisticated geometries and complex flow features.

The methods developed in this thesis represent a significant advancement in the numerical simulation of high-voltage circuit breakers using Cartesian grid methods. Future research directions include extending these methods to handle moving boundaries and implementing solvers for other relevant physical phenomena, which would enhance the industrial applicability of this approach to real circuit breaker simulations.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF SYMBOLS AND ACRONYMS

| | |
|---|---|
| 2D | Two-Dimensional |
| 3D | Three-Dimensional |
| CB | Circuit Breaker |
| CC | Cut-Cells |
| CFD | Computational Fluid Dynamics |
| CFL | Courant-Friedrichs-Lewy |
| FCM | Flux Correction Method |
| FIM | Flux Interpolation Method |
| FRM | Flux Redistribution Method |
| FVM | Finite Volume Method |
| HVCB | High-Voltage Circuit Breaker |
| IB | Immersed Boundary |
| IBM | Immersed Boundary Method |
| $MC^3$ | Modélisation et Calcul de Coupure de Courant |
| MMS | Method of Manufactured Solutions |
| PDE | Partial Differential Equation |
| RCCM | Reconstructed Cut-Cells Method |

# LIST OF APPENDICES

# CHAPTER 1 INTRODUCTION

Fluid mechanics is the branch of physics devoted to the study of fluids and their behavior. The fluid is typically modeled as a continuum characterized by a set of properties such as density, velocity, pressure, and temperature. Conservation principles (*e.g.* mass, momentum, and energy) give rise to the governing equations of fluid mechanics that relate these properties. The renowned Navier-Stokes equations represent one such system that describes the motion of viscous fluids. These governing equations constitute a system of nonlinear Partial Differential Equation (PDE) that, when solved, provide a comprehensive description of the fluid flow evolution. Many applications across engineering, astrophysics, environmental science, biology, and other disciplines can be analyzed through this framework by first modeling the fluid flow and then solving the corresponding governing equations. The solutions of these equations can after be used to make predictions, analysis and design of the systems under study. Since the governing equations are complex and difficult to solve analytically even for simple geometries, numerical methods have become the primary approach for solving them through computational algorithms. Computational Fluid Dynamics (CFD) is the field dedicated to developing and applying numerical methods for solving the governing equations arising in fluid mechanics. With the significant advances in computer hardware and software technologies, CFD has become increasingly essential in scientific research and industrial applications, serving as an indispensable tool across engineering disciplines including aerospace, automotive, and energy. While experimental methods represent an alternative approach to studying fluid flow, they often prove expensive, time-consuming, and sometimes infeasible. Consequently, numerical approaches remain predominant in fluid flow analysis. Nevertheless, experimental data remains valuable for validating assumptions incorporated into CFD models. This thesis adopts a CFD approach to fluid flow analysis and is part of a research project aimed at developing numerical methods to simulate the behavior of high-voltage circuit breakers (HVCBs).

The chapter is organized into three sections. The first section introduces HVCBs and explains their critical role in power networks. The second section examines the modeling of the physical phenomena and the numerical methods employed to simulate HVCBs. The final section outlines the general goal and structure of the thesis.

## 1.1 High-Voltage Circuit Breaker

HVCBs are specialized devices used in high-voltage power grid lines to protect the networks from faults while ensuring continuous power supply. During normal operation, HVCBs allow current to flow through the network with minimal losses. However, when faults such as short circuits or overloads are detected, the HVCB functions as a fuse by interrupting the current flow to prevent network damage. Operating at extreme conditions (with voltages up to 800 kV and currents up to 63 kA), HVCBs face intense mechanical, thermal, and electrical stresses, making their design and operation particularly challenging. This section describes the operation and components of HVCBs and discusses the challenges associated with their numerical simulation.

### 1.1.1 Operation and Components

Figure 1.1 illustrates a schematic representation of an HVCB interrupting chamber and its key components. During normal operation, the circuit breaker remains closed, allowing current to flow through its main contacts, which offer low electrical resistance. The interrupting chamber is filled with an insulating medium that has good dielectric properties and that will facilitates current interruption during the opening process. For high-voltage applications, the employed medium is usually sulfur hexafluoride ($SF_6$) gas, although research is ongoing to identify more environmentally friendly alternatives (*e.g.* $g^3$-*green gas for grid* [4]). When a fault is detected in the network, the circuit breaker is triggered by a network protection device to open, and an operating mechanism (*e.g.* a spring mechanism) of the circuit breaker separates the contacts to interrupt the current flow. The interruption sequence begins with the separation of the main contacts, which redirects current through the arcing contacts. These arcing contacts subsequently separate, establishing an electric arc between them. The arc is a hot ionized plasma that can reach temperatures of approximately $25,000$ K. Since current can continue flowing through this arc, it must be extinguished to complete the interruption process. This is accomplished through a quenching gas blast that cools the arc, enabling the insulating medium to recombine and restore its dielectric strength. The gas blast is self-generated by the arc itself, which heats the surrounding gas, causing it to pressurize within the chamber's expansion volume. When the current intensity approaches the zero of the alternating current, the arc's power diminishes, causing the gas flow toward the expansion volume to reverse. Guided by specialized nozzles, this reversed gas flow extinguishes the arc by cooling it and capturing ionized plasma particles. The interruption is deemed successful when the insulating medium regains sufficient dielectric strength to prevent arc reignition due to the transient recovery voltage that appears post-interruption. The circuit breaker is designed to operate

within a very short time frame to protect the network from damage (usually, the interruption process lasts only 10 to 20 milliseconds).

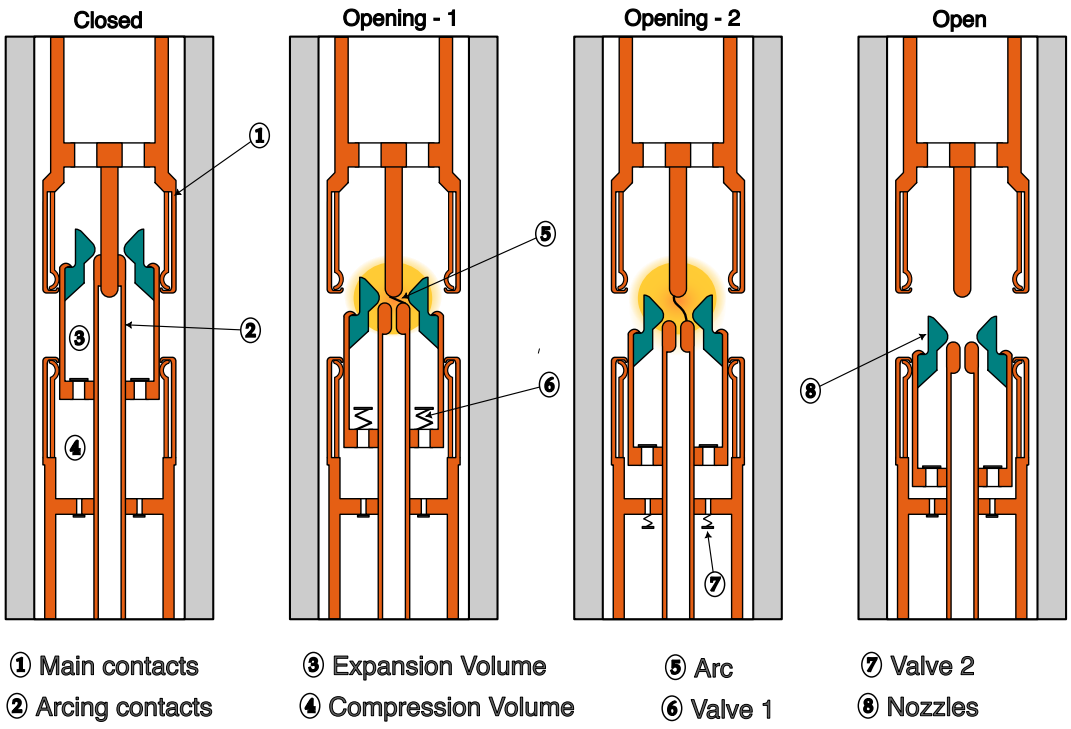



Figure 1.1 Illustration of the different steps of the operation of an HVCB. Adapted from [5].

The other components of the HVCB are also crucial for the operation of the circuit breaker. The two valves allow the circuit breaker to operate in low and high current modes. In low-current situations, the valve in the expansion volume opens, allowing high-pressure gas from the compression volume to flow into the expansion volume and participate in the arc extinction. For the high-current conditions, the pressure in the expansion volume is sufficiently high to quench the arc. In such cases, the valve in the expansion volume closes while the compression volume valve opens to prevent overpressure. Finally, the nozzles, usually made from polytetrafluoroethylene (PTFE), channel gas flow from the expansion volume to the arc region. Also, when the intense radiative flux from the high-temperature arc plasma reaches the nozzles surfaces, the material ablates. This vaporized material contributes to gas pressure build-up in the expansion volume and in the arc extinction. An auxiliary nozzle (a protective cap around the female arcing contact) may also be employed to shield the arcing contacts from erosion.

From the above description, it is clear that the geometry, mechanisms and operation of the HVCB are complex. From an engineering perspective, it is important to understand

the physical phenomena involved in the operation of the circuit breaker to make informed decisions about its design and operation. This is where numerical simulations come into play, as experiments can be very expensive and time-consuming. The CFD approach provides a way to simulate the operation of the HVCB and make valuable insights about its behavior, particularly in the early stages of the design process. The next section will lay out the challenges encountered in the numerical simulations of HVCBs.

### 1.1.2 Modeling and Simulation Challenges

Simulating HVCBs presents numerous challenges that can be classified into two categories: modeling the complex physical phenomena involved in the HVCB operation and addressing the geometric complexities of the circuit breaker design. The operation of an HVCB involves diverse physical phenomena including fluid flow, heat transfer, chemical reactions, and electromagnetic fields interactions. The fluid flow corresponds to a supersonic compressible flow of a real gas that may exhibits turbulent behavior. Joule heating, generated by the resistance to the current flow in the arc plasma, serves as the primary energy source that sustains the arc's high temperature. This intense heat produces significant radiative flux throughout the interrupting chamber, causing ablation of the nozzle materials. Additionally, the arc plasma experiences Lorentz forces resulting from the electromagnetic field generated by the current. These interrelated phenomena are tightly coupled and significantly influence the circuit breaker's performance. Therefore, a comprehensive numerical model must account for all these processes to accurately predict HVCB behavior.

Once the physical phenomena are appropriately modeled, the next challenge involves developing numerical methods to solve the governing equations of the these phenomena. This process typically involves two stages: mesh generation followed by equation solving on the resulting mesh. Various approaches can be employed to solve the governing equations, including the Finite Volume Method (FVM) or the Finite Element Method. Similarly, mesh generation can utilize structured or unstructured techniques. However, the mesh generation can become particularly challenging for complex geometries such as the Three-Dimensional (3D) geometry of the HVCB, which is characterized by the presence of moving components (contacts, valves), narrow gaps, multiple geometric scales, and curved surfaces.

In summary, the numerical simulation of HVCBs represents a significant challenge requiring both sophisticated modeling of the underlying physical phenomena and advanced numerical methods for domain discretization and equation solving. The following section presents a model used in the Modélisation et Calcul de Coupure de Courant ($MC^3$) software developed at Polytechnique Montréal in collaboration with Alstom Grid (now part of GE Vernova) to

simulate the operation of HVCBs [6–14]. The scope and numerical approaches employed in this thesis will also be detailed.

## 1.2 Numerical Simulation of High-Voltage Circuit Breakers

The numerical modeling of the HVCB is centered around the fluid flow and its interaction with the arc plasma. Viscous effects and heat conduction are neglected in the fluid flow. Consequently, the fluid behavior is governed by the Euler Equations, while the arc plasma interacts with the fluid flow through several physical processes: Joule heating, radiation, Lorentz forces, and nozzle ablation. These interactions are incorporated as source terms in the Euler Equations.

Equation 1.1 expresses the balance laws of mass, momentum and energy within a control volume $\Omega$ of the fluid flow:

$$\frac{\partial}{\partial t}\int_{\Omega}\mathbf{U}d\Omega + \oint_{\partial\Omega}\mathbf{F}(\mathbf{U})dS = \int_{\Omega}\mathbf{Q}d\Omega \tag{1.1}$$

Here, $\mathbf{U}$ represents the vector of conserved variables (mass, momentum, and energy), $\mathbf{F}$ denotes the associated inviscid flux vector, and $\mathbf{Q}$ encompasses the arc plasma source term. The conserved variables and the flux vector are defined in Equations 1.2:

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho\mathbf{v} \\ \rho E \end{bmatrix} \qquad \mathbf{F} = \begin{bmatrix} \rho(\mathbf{v}\cdot\mathbf{n}) \\ \rho(\mathbf{v}\cdot\mathbf{n})\mathbf{v} + p\mathbf{n} \\ \rho E(\mathbf{v}\cdot\mathbf{n}) + p(\mathbf{v}\cdot\mathbf{n}) \end{bmatrix} \tag{1.2}$$

Where $\rho$ is the density, $\mathbf{v}$ is the velocity vector, $E$ represents the total energy, $p$ denotes the pressure, and $\mathbf{n}$ is the outward-pointing unit normal vector at the boundary $\partial\Omega$ of the control volume $\Omega$. To close this system of equations, an equation of state $p = p(\rho, e)$ relates pressure to density and internal energy $e$ (defined as $e = E - \frac{1}{2}\|\mathbf{v}\|^2$). For perfect gases, this relationship simplifies to $p = (\gamma - 1)\rho e$, where $\gamma$ is the specific heat ratio. However, for circuit breaker simulations, the perfect gas assumption is no longer valid. Instead, a real gas equation of state is employed, which is implemented through interpolation from thermodynamic tables specific to the insulating medium used in the device.

The arc plasma source term $\mathbf{Q}$ is decomposed into different contributions as shown in Equation 1.3.

$$\mathbf{Q} = \begin{bmatrix} 0 \\ \mathbf{0} \\ q_E^{\text{Joule}} \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{q}_\mathbf{v}^{\text{Lorentz}} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{0} \\ q_E^{\text{radiation}} \end{bmatrix} + \begin{bmatrix} q_\rho^{\text{ablation}} \\ \mathbf{q}_\mathbf{v}^{\text{ablation}} \\ q_E^{\text{ablation}} \end{bmatrix} \tag{1.3}$$

The Joule heating term $q_E^{\text{Joule}}$ and the Lorentz forces terms are defined in Equation 1.4.

$$q_E^{\text{Joule}} = \mathbf{J} \cdot \mathbf{E}, \qquad \mathbf{q}_\mathbf{v}^{\text{Lorentz}} = \mathbf{J} \times \mathbf{B} \tag{1.4}$$

Where $\mathbf{J}$ represents current density determined by Ohm's law ($\mathbf{J} = \sigma\mathbf{E}$, with $\sigma$ being the electrical conductivity of the arc plasma). Vectors $\mathbf{E}$ (not to be confused with the total energy $E$) and $\mathbf{B}$ are the electric and magnetic fields generated by the current flow. By neglecting the transient effects of the electromagnetic field, these fields can be computed from Maxwell's Equations as shown in Equations 1.5 and 1.6.

$$\mathbf{E} = -\nabla\phi, \qquad \nabla \cdot (-\sigma\nabla\phi) = 0 \tag{1.5}$$

$$-\nabla^2\mathbf{B} = \mu_0 \nabla \times \mathbf{J} \tag{1.6}$$

here $\phi$ is the electric potential and $\mu_0$ is the permeability of free space.

The radiation term $q_E^{\text{radiation}}$ can be computed using various models with different levels of complexity and accuracy. One simple approach is the $P1$ model, in which the radiative flux is given by Equation 1.7.

$$q_E^{\text{radiation}} = \int_0^\infty \kappa_\lambda \left(4\pi I_{b\lambda} - G_\lambda\right) d\lambda \tag{1.7}$$

The spectral parameters $\kappa_\lambda$, $I_{b\lambda}$ and $G_\lambda$ correspond to the radiation absorption coefficient, blackbody radiative intensity and incident radiation at the wavelength $\lambda$, respectively. The blackbody spectral radiative intensity follows Planck's law, while the incident spectral radiation is computed from the simplified radiative transfer equation given by Equation 1.8.

$$\nabla \cdot \left(\frac{1}{\kappa_\lambda}\nabla G_\lambda\right) = 3\kappa_\lambda \left(G_\lambda - 4\pi I_{b\lambda}\right) \tag{1.8}$$

It is worth noting that Equations 1.5, 1.6 and 1.8 can be solved using the same numerical solver as they all correspond to the Helmholtz equation given by Equation 1.9.

$$\oint_{\partial\Omega} \kappa \nabla\psi \cdot \mathbf{n} dS + \int_{\Omega} \alpha\psi d\Omega = \int_{\Omega} q d\Omega \tag{1.9}$$

Where $\psi$ is the unknown field, and $\kappa$, $\alpha$, and $q$ are parameters independent of $\psi$ corresponding to a diffusion coefficient, an absorption coefficient and a source term, respectively.

Finally, the nozzle ablation terms are computed using the radiation flux from the arc plasma to the nozzle surfaces. Equation 1.10 provides a simple model for calculating the rate of mass ablation $\dot{m}_{\text{ablation}}$.

$$\dot{m}_{\text{ablation}} = \frac{q^{\text{nozzle}}_{\text{radiation}}}{h_v + \delta h_a} \tag{1.10}$$

Where $q^{\text{nozzle}}_{\text{radiation}}$ is the radiation flux from the arc plasma to the nozzle surfaces, $h_v$ is the vaporization enthalpy of the nozzle material and $\delta h_a$ is the difference in internal energy required to heat the fluid to the nozzle vaporization temperature. The ablated mass is then used to compute the ablation source term given by Equation 1.11.

$$\begin{bmatrix} q_{\rho}^{\text{ablation}} \\ \mathbf{q}_{\mathbf{v}}^{\text{ablation}} \\ q_{E}^{\text{ablation}} \end{bmatrix} = \begin{bmatrix} \frac{1}{|\Omega|}\int_{\Gamma} \dot{m}_{\text{ablation}} dS \\ \frac{1}{|\Omega|}\int_{\Gamma} \dot{m}_{\text{ablation}} \mathbf{v} dS \\ \frac{1}{|\Omega|}\int_{\Gamma} \dot{m}_{\text{ablation}} E dS \end{bmatrix} \tag{1.11}$$

Where $\Gamma$ represents the ablated surface of the nozzle and $|\Omega|$ is the volume of the control volume $\Omega$.

This comprehensive modeling framework captures the essential physical phenomena involved in the operation of the HVCB. Figures 1.2 and 1.3 demonstrate the results of an HVCB simulation using the MC$^3$ software and the validation of the numerical results using experimental data, respectively. The close agreement between numerical and experimental data confirms the accuracy and reliability of the numerical model implemented in the MC$^3$ software. After many decades of research and development, the MC$^3$ software has become a well-established tool used by engineers in their everyday work to simulate the operation of HVCBs. The code is based on a 2D axisymmetric model of the HVCB and uses the FVM method with an adaptive unstructured triangular mesh to solve the governing equations of the flow. In recent years, since the 2D axisymmetric model has some limitations in capturing 3D effects presents in the circuit breaker, a new generation of software tools is being developed to simulate the HVCB in 3D. For these new software tools, the FVM is still used to solve the governing equations of the flow but the mesh generation is done using the Immersed Boundary Method (IBM) approach to handle the complex geometry of the HVCB. The IBM

approach significantly simplifies the mesh generation process and offers great potential for handling the moving components within the circuit breaker. This thesis is part of this new project, which aims to employ Cartesian grid methods and the IBM approach to simulate the operation of the HVCB in 3D [15–18].

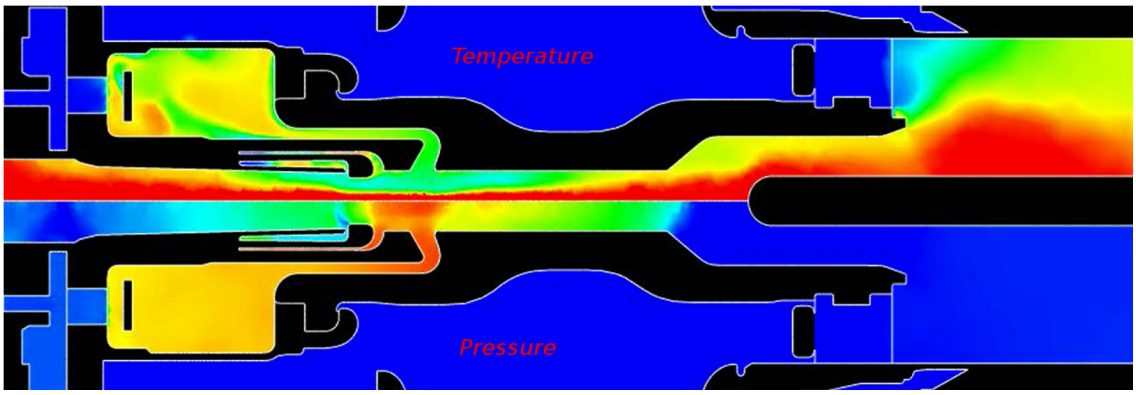



Figure 1.2 Solution of an HVCB simulation using the MC$^3$ software [12].

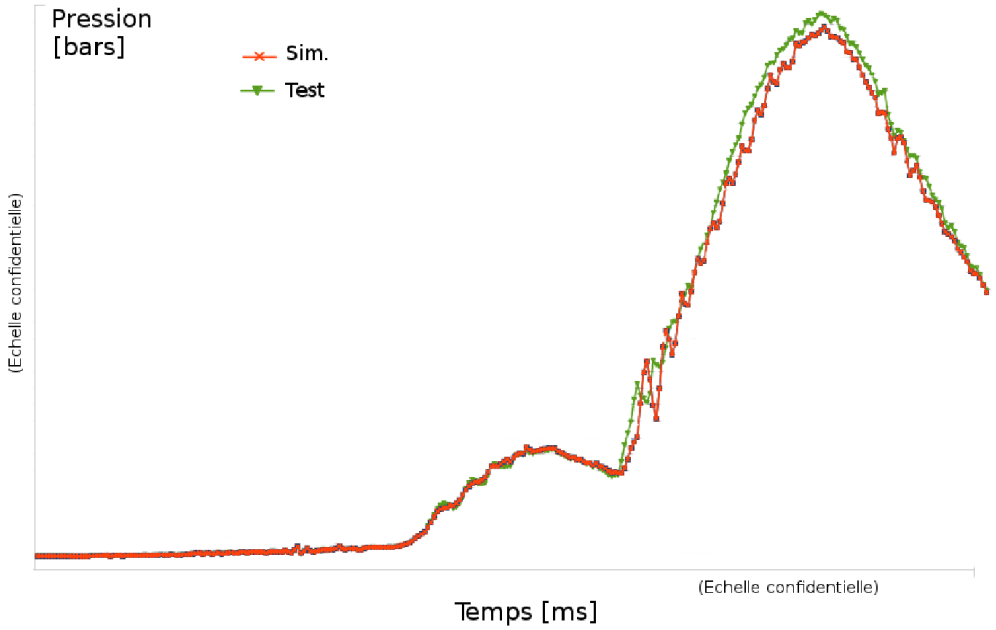



Figure 1.3 Validation of numerical results from an HVCB simulation against experimental pressure measurements over time [12]. Axis scales are confidential.

## 1.3 Research Hypotheses and Outline of the Thesis

This section will present the general goal along with the research hypotheses and the outline of the thesis.

### Research Hypotheses and General Goal

The ultimate objective of the broader research project is to contribute to the development of numerical methods for simulating the operation of HVCBs. However, the scope of this specific thesis is narrower, concentrating on the development of numerical techniques for simulating compressible flows within complex geometries using Cartesian grid methods combined with the IBM approach. The following assumptions are adopted throughout this work:

- The flow is considered compressible and inviscid, governed by the Euler equations.
- The perfect gas equation of state is used to close the system of governing equations.
- Source terms related to arc plasma phenomena (Joule heating, Lorentz forces, radiation, and ablation) are neglected.
- The computational domain does not include moving components.

These simplifications allow for a focused investigation into the core numerical methods for flow simulation using the IBM approach in the context of complex geometries. The primary technical challenge involves preserving essential properties of the FVM, notably conservation and numerical accuracy, when implementing the IBM approach. Therefore, the central goal of this thesis is to develop numerical methods that effectively combine the FVM and IBM for solving the Euler equations in 3D. These methods must maintain critical numerical properties, including conservation, second-order spatial accuracy, and computational stability, while demonstrating the capability to handle intricate geometries and complex flow patterns representative of HVCBs.

### Thesis Outline

The thesis is organized as follows:

- Chapter 2 presents a literature review of the Cartesian grid methods for the numerical simulation of fluid flow.

- Chapter 3 outlines the specific research objectives and details the scientific contributions of the thesis.
- Chapter 4 and Chapter 5 introduce novel numerical methods that integrate FVM and IBM approaches to solve the Euler Equations in two dimensions. Chapter 4 focuses on ensuring conservation properties, while Chapter 5 addresses second-order accuracy requirements.
- Chapter 6 present an implementation in 3D of the IBM approach to solve the Euler Equations.
- Chapter 7 concludes the thesis with a critical assessment of the developed methods, discussing their limitations and providing perspectives for future research.
- Appendices A and B provide technical details on the specialized data structures used to define the computational domain and the efficient algorithms developed for Cartesian grid generation, respectively.

# CHAPTER 2 LITERATURE REVIEW

Mesh generation is one of the most fundamental step in the development of a numerical method for solving fluid flow problems. The quality of the mesh has a direct impact on the accuracy and performance of the numerical method. For complex geometries, generating a high-quality mesh is a challenging and time-consuming task that often requires significant manual intervention from users. Over the past several decades, the development of automated mesh generation techniques has been an active area of research, leading to the creation of various mesh generation approaches.

The structured mesh generation approach have been one of the most widely used mesh generation techniques. For simple geometries, structured mesh are easy to generate and provide good quality results. However, for complex geometries, structured mesh generation becomes increasingly difficult and often necessitates the use of more advanced mesh generation techniques such as multi-block structured or overset mesh generation. In the multi-block structured mesh generation approach, the domain is divided into several blocks, and a structured mesh is generated within each block. The structured meshes are then stitched together to form a single mesh. The decomposition of the domain into blocks depends heavily on the geometry of the particular problem at hand and can quickly become complex to automate. The overset mesh generation approach, on the other hand, is similar to the multi-block structured mesh generation approach, but the blocks are allowed to overlap each other. This provides greater flexibility in assembling the final mesh but requires the use of interpolation to transfer information between overlapping blocks.

The complexity of the structured mesh generation approach has led to the development of unstructured mesh generation techniques. Unstructured mesh offers more flexibility than their structured counterparts and are well-suited for generating meshes for complex geometries. As a result, unstructured mesh generation techniques has become the preferred choice for generating meshes in many applications involving complex geometries. However, unstructured mesh generation is not without its challenges. Unstructured mesh require complex data structures and more memory usage to store the mesh connectivity information. The discretization of the governing equations on unstructured mesh is more complex and necessitates more computational power to reach the same level of accuracy as structured mesh. In addition, for very complex 3D geometries that contain moving parts, even unstructured mesh generation can become a highly challenging task.

In recent years, Cartesian grid methods have gained popularity as an alternative to the

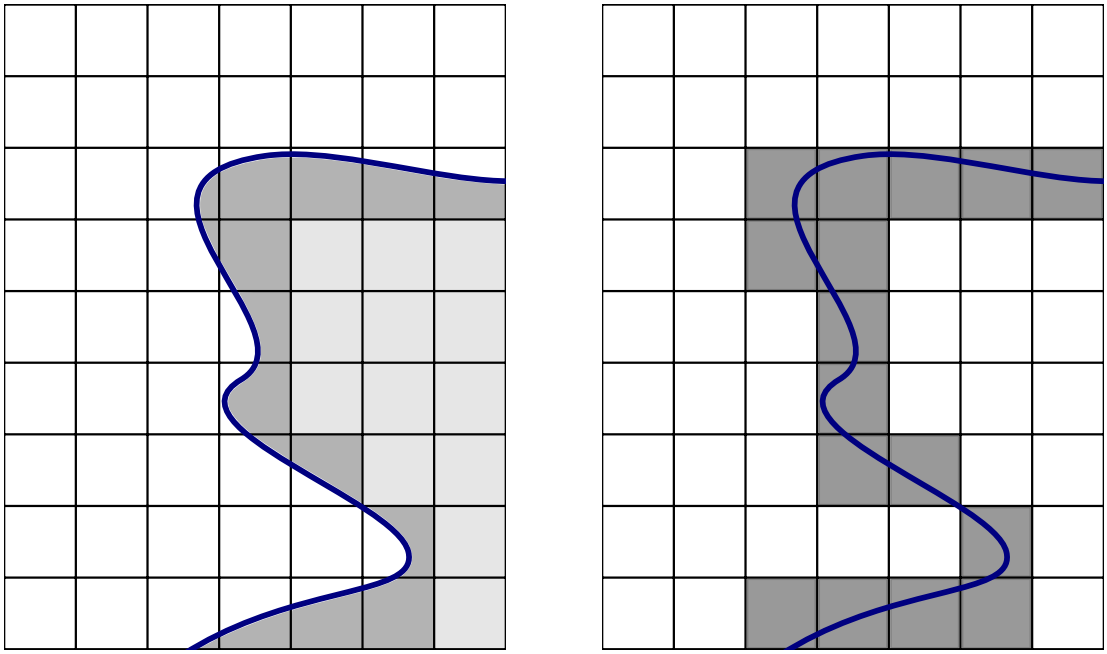
traditional mesh generation techniques for tackling CFD simulations involving complex geometries. Cartesian grid methods are based on the use of a regular Cartesian grid that is non-conformal to the geometry of the problem, and the discretization of the governing equations is performed directly on the Cartesian grid. The use of Cartesian grid methods simplifies the mesh data structure and discretization of the governing equations associated with unstructured mesh generation while retaining the good numerical properties of structured mesh generation. Furthermore, since the Cartesian grid can be generated independently of the geometry of the problem, the mesh generation step for Cartesian grid methods can be fully automated, eliminating the need for user intervention. This makes Cartesian grid methods an attractive choice for automated mesh generation techniques in complex geometries.

The Cartesian grid methods can be broadly classified into two categories: Cut-Cells (CC) and Immersed Boundary Method (IBM). In the CC method, cells that intersect the geometry are clipped to make them conformal to the geometry. These clipped cells, referred to as cut-cells, along with uncut cells inside the geometry, are assembled to form the final mesh. As a result, discretization methods designed for unstructured meshes can be directly applied without modification. In contrast, the IBM method does not require the mesh to conform to the geometry. Instead, the discretization of the governing equations is performed directly on the Cartesian grid, with additional terms added to account for the presence of the geometry. While the IBM method offers greater flexibility in the mesh generation process, it requires more sophisticated discretization techniques to handle boundary conditions effectively. Figure 2.1 illustrates a comparison of the grids generated by the CC and IBM methods for a simple geometry.

In this chapter, a critical review of the literature of the Cartesian Grid methods is presented. The chapter is divided into two sections, each focusing on a detailed review of the CC and IBM methods, respectively.

## 2.1 Cut-Cells Method

This section presents a review of the literature on the CC method. It is divided into two parts: the first part describes the CC method, covering its history, advantages, and disadvantages; the second part discusses the stability issues associated with the CC method and reviews the techniques developed to address them.



(a) Cut-Cells (CC) (b) Immersed Boundary Method (IBM)

Figure 2.1 Comparison of the grids generated by the CC and IBM methods for a simple geometry. In the CC method (left), cells intersecting the boundary (blue line) are clipped to create a body-conformal mesh composed of regular and cut-cells. In the IBM method (right), the underlying Cartesian grid remains unaltered. The boundary's presence is accounted for by modifying the numerical scheme in the vicinity of the immersed geometry.

### 2.1.1 Description of the Method

#### History

The use of Cartesian grid methods with cut-cells can be traced back to the late 1970s and early 1980s, beginning with the work of Purvis *et al.* [19] and Wedan *et al.* [20], who employed it to solve the potential equations. In the mid-1980s, the method was extended to solve the Euler equations by Clarke *et al.* [21] for two-dimensional geometries and by Gaffney *et al.* [22] for three-dimensional geometries. Over the following decades, the CC method has been further developed and applied to a wide range of applications involving complex geometries. Notable achievements during this period include the development of successful software tools based on the CC approach, such as CART3D [23] and TRANAIR [24].

### Advantages

The CC method offers several advantages over traditional mesh generation techniques (structured multi-block, unstructured, overset) for solving fluid flow problems involving complex geometries. One of the key advantages of the CC method is its suitability for automated mesh generation. The grid generation process requires only the computation of cut-cells, which is a geometric operation that can be fully automated. This makes the CC method particularly well-suited for applications requiring efficient and automated workflows. Within the computational domain, the CC method employs regular Cartesian cells, which possess desirable numerical properties such as orthogonality, implicit connectivity, and data locality. These properties are highly beneficial for the discretization of governing equations, improving both accuracy and computational efficiency. Lastly, the CC method decouples the geometry generation process from the mesh generation process. This allows for greater flexibility in geometry representation, as the geometry can be generated independently without constraints imposed by the mesh. Various geometry representations, such as Boundary Representation (B-rep) [25] and Constructive Solid Geometry (CSG) [26], can be used. This decoupling of geometry generation from mesh generation processes further emphasizes the adaptability of the CC method for complex geometries. The only requirement for the CC method is an intersection algorithm to compute the cut-cells from the geometry representation and the Cartesian grid.

### Limitations

Although the CC method possesses many advantages, it also faces significant limitations that have prompted numerous research efforts. The first major challenge of the CC method concerns the creation of the cut-cells for complex 3D geometries. The geometric operation required to compute cut-cells can be intricate and prone to robustness issues due to finite-precision computer arithmetic. Moreover, for moving geometries, cut-cells must be recomputed at each time step, increasing computational cost and potentially exacerbating numerical robustness concerns. In reference [27], several strategies to compute the cut-cells for 3D geometries composed by triangulated surfaces are presented, along with strategies to improve the robustness of the cut-cells computation. Figure 2.2 illustrates three 2D examples showing different scenarios encountered when computing cut-cells. The middle example depicts a “split-cell”, where a regular cell generates two disconnected cut-cells. Similar or even more complex situations can arise in 3D geometries. Therefore, the mesh generation process for the CC method must be sufficiently robust to handle all such cases effectively.

Furthermore, since the cut-cells can have arbitrary polyhedral shapes, mesh quality near

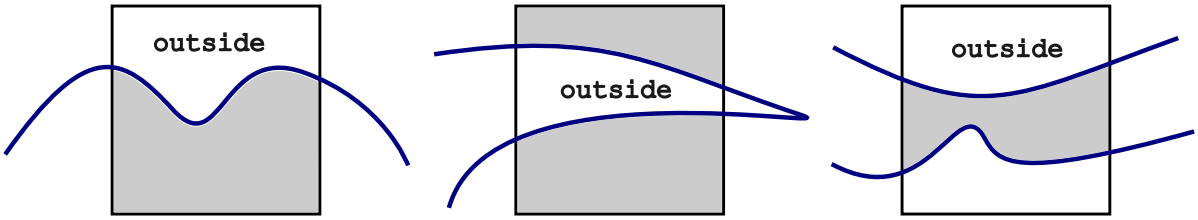



Figure 2.2 Examples illustrating various scenarios encountered during cut-cells computation in 2D.

the boundaries may be poor, potentially affecting the accuracy of the numerical method. Additionally, storing cut-cells requires more advanced data structures than those used for regular Cartesian cells. As a result, in practice, this often leads to treating the entire CC grid as an unstructured mesh to uniformly handle the discretization of governing equations across both cut-cells and regular cells.

Another notable issue with the CC method is that it is generally not well suited for viscous flow simulations. Cartesian grids typically do not align with boundaries and lack anisotropic refinement capabilities, resulting in poor boundary layer resolution and potentially inaccurate results. Although some isotropic refinement techniques (e.g., octree) can enhance grid resolution near boundaries, they rarely provide satisfactory resolution of boundary layers. Some recent studies [28] have proposed wall modeling techniques to alleviate boundary-layer resolution problems in viscous flow simulations with the CC method. Other works [29] have explored hybrid mesh approaches, where a body-fitted mesh is used near the boundary to accurately resolve the boundary layer, while Cartesian grids cover the remaining domain. These body-fitted meshes, designed to satisfy boundary layer resolution requirements, are generated using traditional techniques. The two mesh types are then either combined into a single mesh using techniques such as Laplacian smoothing or treated as overset meshes. Although this approach can effectively resolve boundary layers, it may compromise the automation of the mesh generation process. Alternatively, the Cartesian cut-cells grid can be replaced by an unstructured cut-cells grid [30]. Since only a constraint on accurate wall resolution is required, generating an unstructured cut-cells grid can be simpler and more automated than creating a fully body-fitted mesh. However, unstructured cut-cells grid discards some of the positive attributes of the Cartesian cut-cells grid, such as orthogonality, implicit connectivity, and data locality, and requires more complex algorithms for cut-cells computation.

A final limitation of the CC method is the small cell stability problem [31], which arises when an explicit time integration scheme is used to solve the governing equations. In this case, the maximum allowable time step may be severely restricted by the smallest cell in the mesh,

which is generally a very small cut-cell. While implicit time integration schemes could address this issue, they are prohibitively expensive and impractical for large-scale simulations.

Since this thesis focuses on inviscid flow simulations, the limitations of the CC method regarding the viscous flows simulations will not be addressed. Instead, attention will be directed toward the challenges associated with cut-cells, particularly the small cell problem. The next section reviews the literature on stability issues associated with small cut-cells and the methods developed to overcome them.

### 2.1.2 Small Cell Problem

The small cell problem can be understood by examining the FVM discretization of the Euler equations, as shown in Equation (2.1), where the solution is updated at each time step using an explicit time integration scheme:

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\Delta t}{\Delta V_i} \sum_{f \in \partial V_i} \mathbf{F}_f^n A_f \tag{2.1}$$

where $\mathbf{U}_i^n$ and $\mathbf{U}_i^{n+1}$ represent the cell-averaged solution at time iterations $n$ and $n+1$, respectively. $\Delta t$ is the time step, $\Delta V_i$ is the volume of cell $i$, $\partial V_i$ denotes the set of faces of cell $i$, $\mathbf{F}_f^n$ is the numerical flux at face $f$, and $A_f$ is the area of face $f$. The time step $\Delta t$ is constrained by the Courant-Friedrichs-Lewy (CFL) [32] condition given by Equation (2.2).

$$\Delta t \leq \min_i \frac{\Delta x_i}{|\mathbf{u}_i| + c_i} \tag{2.2}$$

where $\Delta x_i \approx \Delta V_i^{1/3}$ is the characteristic length of cell $i$, $\mathbf{u}_i$ is the velocity at cell $i$, and $c_i$ is the local speed of sound. For cells with small volumes, particularly cut-cells, the characteristic length $\Delta x_i$ becomes very small, resulting in a severely restricted time step according to the CFL condition. When dealing with geometries that are not aligned with the grid, there will almost inevitably be cut-cells with extremely small volumes that dramatically constrain the global time step of the simulation. This is the essence of the small cell problem, which represents a significant limitation of the CC method. If the global time step restriction dictated by these small cells is not enforced, the numerical algorithm becomes unstable, causing the simulation to fail to converge. Therefore, a systematic approach to addressing the small cell problem is essential for the CC method to be viable in practical applications. The following paragraphs provide a review of various methods that have been developed to address this challenge.

### Cell-Merging and Related Methods

The cell-merging technique represents one of the earliest approaches developed to address the small cell problem. This method was already implemented in one of the pioneering works on the CC method by Clarke *et al.* [21]. The cell-merging approach involves merging the small cut-cells with their neighbors to form larger cells that can be updated with a larger time step. A typical criterion for determining which cells to merge is to use a threshold (e.g., 1/2) applied to the ratio between the cut-cell volume and the volume of its corresponding regular cell. When this ratio falls below the threshold, the cut-cell is merged with an adjacent cell. The cell-merging technique offers several advantages, including its conceptual simplicity and its applicability to moving geometries. For moving boundary problems, new cells without solution history emerge as the geometry moves. In such cases, cell-merging provides an elegant solution by incorporating these new cells with their neighbors, thereby providing them with a solution history [33]. The effectiveness of the cell-merging technique in handling the small cell problem has been demonstrated in numerous studies [22, 34–38]. The main drawback of the technique is that, despite its conceptual simplicity, its implementation can be challenging and may require complex data structures for robustly managing the merging process, especially in three dimensions [31, 34, 38]. Additionally, the cell-merging technique can lead to a slight loss of accuracy near the boundary of the geometry [39].

To simplify the implementation of the cell-merging technique, a variant known as cell-linking has been proposed in references [40–43]. In the cell-linking technique, the update of the cut-cells is linked to the update of their neighbors without physically merging them. This strategy circumvents many implementation challenges of traditional cell-merging while maintaining computational stability. However, cell-linking techniques still exhibit the same fundamental limitation of reduced accuracy near boundaries as their cell-merging counterparts [39, 40].

A more recent variant of the cell-merging technique is the state redistribution (SRD) technique [44–46]. In the SRD technique, multiple, potentially overlapping, sets of cells around the cut-cells are formed. The solution is computed separately for each set and subsequently redistributed to determine the final solution for the cut-cells. Similar to cell-linking, SRD eliminates the need to create new merged cells, thereby circumventing many implementation issues of traditional cell-merging algorithms. The SRD technique has shown promising results across various test cases [44,47,48]; for instance, [44], reports that SRD maintains global conservation while achieving boundary accuracy between 1.4 and 1.5 order for a second-order scheme.

### Flux Redistribution

The flux redistribution technique is perhaps the simplest technique for addressing the small cell problem [49–52]. From the FVM discretization of the Euler equations in Equation (2.1), the flux redistribution technique consists of updating the cut-cells using only a small fraction of the computed flux (an amount that allows stability with a larger time step). The remaining flux is then redistributed to the neighboring cells in a manner that preserves the global conservation of mass, momentum, and energy. The proportion of the flux used to update the cut-cells can be set to the ratio of the cut-cell's volume to that of the corresponding regular cell. For the redistribution of the flux to the neighboring cells, a volume-weighted or mass-weighted averaging scheme is commonly employed. The flux redistribution technique has been successfully implemented in both 2D and 3D simulations [52, 53] and the redistribution has been shown to be computationally inexpensive compared to the cell-merging techniques. Despite its computational advantages, the flux redistribution technique can lead to a loss of accuracy especially near the boundaries of the geometry. For instance, in reference [52] where flux redistribution technique is used, the method maintained second-order accuracy throughout most of the computational domain, but the accuracy near boundaries deteriorated to first-order.

### Large Time Step Method And Similar Approaches

The small cell problem can also be addressed by using methods that allow to take larger time steps. In references [54–56], a wave-propagation approach that expands the stability region of the finite volume scheme on the cut-cells to allow larger time steps is proposed. This approach permits waves to propagate further in the cut-cells than standard Godunov-type methods would allow. The solution update incorporates all waves crossing the cut-cells in a linear way (i.e. assuming the waves pass through each other without interacting or creating new waves). This allows for larger time steps on cut-cells, and thus alleviate the small cell problem. Unfortunately, the method maintains stability only for cells with volume fractions greater than 3% [55]. This constraint is problematic since many cut-cells in complex geometries can have volume fractions several orders of magnitude smaller than this threshold. For cells with volume fractions below this critical value, a weighted average interpolation similar to the flux redistribution technique is used in reference [55].

An enhancement of the large time steps method called h-box method has been proposed in references [57, 58]. While the large time steps method focuses on the domain of influence, the h-box method utilizes the domain of dependence of cut-cells to permit larger time steps. The method constructs boxes of size h (corresponding to regular cells size) around each cut-cell,

computes finite volume fluxes on these boxes, and then uses these fluxes to update the cut-cells. The method's stability derives from carefully chosen boxes that create a cancellation property among the fluxes, effectively eliminating the stability constraint on cut-cells. The h-box method has been well-formulated for one-dimensional and two-dimensional problems. Although the two-dimensional implementation is quite complex, it effectively enables larger time steps for cut-cells while maintaining second-order accuracy throughout the entire domain, including near boundaries. Despite its success in lower dimensions, the h-box method has not yet been extended to three-dimensional problems due to the substantial complexity involved in formulating such an extension.

Finally, the use of implicit time integration schemes can offer another solution to the small cell problem. By ruling out a fully implicit time integration scheme due to its computational cost, a semi-implicit time integration scheme can be a viable alternative. In reference [59], a semi-implicit method for the 2D scalar advection equation is proposed. The method discretizes the cut-cells implicitly and the regular cells explicitly, with appropriate coupling at the interfaces between these two cell types. Although this approach has not yet been extended to the Euler equations, it shows promise for addressing the small cell problem. One implementation strategy could involve creating a thin layer of implicitly treated cells around the cut-cells, while discretizing the remainder of the domain explicitly. This layer would be designed to be as thin as possible to minimize computational overhead while remaining sufficiently thick to maintain stability for the small cut-cells.

**Summary**

This section has presented a comprehensive review of stability issues associated with small cut-cells and the various methods developed to address them. Each approach offers distinct advantages and limitations that make it suitable for specific applications. Table 2.1 summarizes the capabilities of these methods based on several criteria including accuracy, computational efficiency, implementation complexity, and robustness. The cell-merging and related techniques provide conceptual simplicity but can be challenging to implement robustly. Flux redistribution methods offer computational efficiency but may sacrifice accuracy near boundaries. Alternative time integration approaches can maintain higher-order accuracy but typically involve more complex formulations, especially in three dimensions. The development of novel methods to address the small cell problem remains an active area of research, with significant advancements still emerging. From a practical standpoint, the selection of an appropriate method depends on the specific application requirements and the acceptable trade-offs between accuracy, computational cost, and robustness. Despite its po-

tential accuracy limitations near boundaries, the flux redistribution method often represents a viable compromise due to its relative simplicity and computational efficiency, making it a common choice for handling the small cell problem.

In the next section, a review of the literature of the IBM methods is presented. The IBM approach represents an alternative to the Cut-Cells method that can potentially simplify the treatment of small cells or even eliminate this concern entirely. Additionally, IBM method addresses several other limitations inherent to the Cut-Cells approach, providing a complementary perspective on handling complex geometries in CFD simulations.

Table 2.1 Summary of the capabilities of the various methods to address the small cell problem.

| Method | Accuracy | Computational Cost | Robustness | Simplicity | Conservation |
|---|---|---|---|---|---|
| Cell-Merging | ✘ | ✘ | ✘ | ✔✔ | ✔✔ |
| Cell-Linking | ✘ | ✔ | ✔ | ✘ | ✔ |
| State Redistribution | ✔ | ✔ | ✔ | ✘ | ✔ |
| Flux Redistribution | ✘ | ✔✔✔ | ✔ | ✔✔ | ✔ |
| Large Time Steps | ✔✔✔ | ✘✘ | ✘ | ✘✘ | ✔✔✔ |
| H-Box | ✔✔✔ | ✘✘ | ✔ | ✘✘✘ | ✔✔✔ |
| Explicit-Implicit | ✔✔✔ | ✘✘✘ | ✔ | ✘✘ | ✔✔✔ |

## 2.2 Immersed Boundary Method

This section presents a brief review of the literature on the IBM method. The section is divided into two parts: first, an overview and characteristics of the IBM approach, and second, a discussion of its various implementations.

### 2.2.1 Description of the Method

This subsection provides an overview of the IBM method, covering its advantages, disadvantages, and a classification of its various approaches.

#### Overview

The IBM method fundamentally simplifies the mesh generation process by removing the body-fitted constraint of the mesh. In this approach, the physical geometry is immersed within a background grid (typically Cartesian for computational efficiency) and can intersect

this grid arbitrarily. The IBM approach will now consist of discretizing the governing equations directly on the Cartesian grid, with additional terms incorporated to account for the presence of the immersed boundary. While this approach simplifies mesh generation, it introduces new challenges in boundary condition implementation. Consequently, the IBM can be viewed as a trade-off: simplified mesh generation versus more complex boundary treatment. The primary challenge lies in accurately imposing boundary conditions while maintaining important numerical properties such as conservation of physical conserved quantities such as mass, momentum, and energy. The CC method can be considered a specialized case of the IBM method where the geometry is used to clip the cells of the Cartesian grid, and boundary conditions are imposed using the same techniques to those in body-fitted methods. In this thesis, however, the CC method will be considered to be distinct from the IBM method. Here, IBM method refers specifically to methods that do not require cells to conform to the geometry for discretizing the governing equations.

The IBM method was introduced by Peskin in the early 1970s [60] to simulate cardiovascular flows. Historical analysis of the IBM method by Verzicco [61] identified even earlier works that can be considered as precursors of the IBM method dating back to the late 1950s, where non-body-fitted methods were employed for fluid flow simulations [62–65]. Since its introduction, the IBM method has undergone significant development and has been applied to diverse applications including fluid-structure interaction [66, 67], biological flows [68–73], multi-phase flows [74], incompressible and compressible flows [75–78], heat transfer [79, 80], and many others.

**Advantages and Disadvantages**

The IBM offers several advantages compared to traditional body-fitted mesh methods. Most significantly, it further simplifies mesh generation compared to the CC method since the Cartesian grid cells do not need to be clipped by the geometry. This provides greater flexibility in geometry representation and accommodates more complex configurations. For moving geometries, the IBM offers particular advantages as the geometry can be relocated within the same background grid without regenerating a new mesh. Additional benefits include the use of Cartesian grids that possess favorable numerical properties such as orthogonality, implicit connectivity, data locality, reduced CPU usage and memory requirements, and easier parallelization. The mesh quality near boundaries also surpasses that of the CC method since the Cartesian structure is preserved throughout the domain.

Despite these advantages, the IBM has also some limitations. Like with the CC method, the IBM Cartesian grid will typically not be aligned with the boundaries. This, combined

with the lack of anisotropic refinement capabilities, can result in inadequate resolution of boundary layers, leading to inaccurate results for viscous flow simulations. Furthermore, boundary condition implementation is more complex than in body-fitted approaches and may require sophisticated techniques to maintain numerical accuracy. Additionally, since the geometry does not conform to the mesh, preserving conservation of mass, momentum, and energy presents greater challenges.

#### Classification of the IBM Methods

The different approaches to the IBM method are typically classified according to how boundary conditions are imposed. In the comprehensive review of the IBM method by Mittal and Iaccarino [81], the IBM methods are categorized into two main groups: continuous forcing methods and discrete forcing methods. Continuous forcing methods impose boundary conditions by adding source terms to the governing equations that force the solution to satisfy the boundary conditions. This extra term is called a forcing term (hence the terminology "forcing") and is commonly computed by modeling the interaction between the fluid and the boundary. Conversely, discrete forcing methods modify the discretized form of the governing equations for cells near the boundary to account for the boundary's presence. Alternative classifications of the IBM methods exist, such as that proposed in reference [66], which divides IBM approaches into diffuse interface methods and sharp interface methods. In diffuse interface methods, boundary conditions are enforced by distributing source terms across a region surrounding the boundaries. Sharp interface methods, however, directly act on the cells adjacent to the boundary to enforce the boundary conditions without using a source term that is spread over a region around the boundaries. These two classifications largely overlap with each other, with continuous forcing methods corresponding to diffuse interface methods and discrete forcing methods aligning with sharp interface methods. The following sections provide an overview of both continuous forcing and discrete forcing approaches to the IBM method.

### 2.2.2 Continuous Forcing Methods

#### Classical IBM Method of Peskin

In Peskin's classical IBM method [60], boundaries $\Gamma$ are represented by a set of Lagrangian points $\mathbf{X}_k$ connected by springs. If the incompressible Navier-Stokes equations are considered for example, boundary conditions are imposed by adding a forcing term to the momentum

equation given by Equation (2.3).

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f} \tag{2.3}$$

where $\mathbf{u}$ is the velocity field, $p$ is the pressure field, $\nu$ is the kinematic viscosity, and $\mathbf{f}$ is the forcing term derived from forces $\mathbf{F}_k$ acting on the Lagrangian points $\mathbf{X}_k$. These forces are calculated using constitutive laws (e.g., Hooke's law [82]) that model the interactions between the fluid and the boundary. The forcing term $\mathbf{f}$ is given by Equation (2.4) where each forcing term $\mathbf{F}_k$ on the Lagrangian points is spread to the Eulerian grid using a smooth kernel function $\delta(|\mathbf{x} - \mathbf{X}_k|)$ centered at the Lagrangian points.

$$\mathbf{f}(\mathbf{x}, t) = \int_\Gamma \mathbf{F}(\mathbf{q}, t)\delta(|\mathbf{x} - \mathbf{X}(\mathbf{q}, t)|)d\mathbf{q} = \sum_k \mathbf{F}_k(t)\delta(|\mathbf{x} - \mathbf{X}_k(t)|)\Delta\mathbf{X} \tag{2.4}$$

By completing the momentum equation with the continuity equation and the tracking of the Lagrangian points given by Equations (2.5) and (2.6) respectively, the classical IBM method of Peskin can be used to simulate fluid flows around complex geometries.

$$\nabla \cdot \mathbf{u} = 0 \tag{2.5}$$

$$\frac{\partial \mathbf{X}_k}{\partial t} = \mathbf{u}(\mathbf{X}_k, t) = \int_\Omega \mathbf{u}(\mathbf{x}, t)\delta(|\mathbf{x} - \mathbf{X}_k(, t)|)d\mathbf{x} \tag{2.6}$$

$$= \sum_i \mathbf{u}(\mathbf{x}_i, t)\delta(|\mathbf{x}_i - \mathbf{X}_k(t)|)\Delta\mathbf{x} \tag{2.7}$$

The classical IBM method of Peskin is theoretically well-founded [83] and particularly suitable for flows with elastic boundaries, such as those encountered in biological flows applications [67]. The main limitation of this method is its inapplicability to rigid boundaries, where large forcing terms can create numerical instabilities leading to erroneous solutions. Additionally, due to the diffuse nature of the forcing term, this method achieves only first-order accuracy at boundaries.

### Continuous Forcing Method for Rigid Boundaries

To simulate flows interacting with rigid boundaries, a penalization method has been proposed in references [84, 85] where the computational domain is treated as a porous medium with fluid and solid phases. The coupled Navier-Stokes and Brinkman equations [86, 87] are used to model the problem. In this method, the forcing term is expressed as $\mathbf{f} = c\mathbf{u}$, where $c$ represents the ratio between the fluid viscosity and the permeability of the solid phase. In the fluid phase, $c$ is set to a value close to zero and in the solid phase, $c$ is set to a large

value. The penalization method can be written in a more general form by using the forcing term given by Equation (2.8) proposed by Goldstein *et al.* [88].

$$\mathbf{f}(\mathbf{x},t) = \alpha \int_0^t \mathbf{u}(\mathbf{x},t')dt' + \beta \mathbf{u}(\mathbf{x},t) \tag{2.8}$$

where $\alpha$ and $\beta$ are user-defined parameters. When $\alpha = 0$, this reduces to the basic penalization method described earlier. The forcing term given by Equation (2.8) can be interpreted as a damped oscillator system that enforces boundary conditions through a feedback mechanism [89].

The penalization method and its variants have proven effective for simulating flows around rigid boundaries in numerous applications [90]. However, these methods can suffer from instabilities due to the stiffness of the discrete system, potentially imposing severe time step restrictions. Additionally, they require carefully tuned parameters that must be adjusted for each specific application [89, 91–93].

### 2.2.3 Discrete Forcing Methods

Discrete forcing methods account for embedded boundaries by modifying the discretized form of the governing equations. Typically, the discretization of cells adjacent to the boundary is adjusted to satisfy boundary conditions. Unlike continuous forcing methods, discrete forcing methods retain the boundaries sharpness and impose boundary conditions directly on the computational grid. By precisely controlling the discretization of near-boundary cells, discrete forcing methods can enforce boundary conditions with greater accuracy than continuous forcing approaches.

While numerous implementations of discrete forcing methods exist, they generally operate according to similar principles. Figure 2.3 illustrates a simple geometry immersed in a Cartesian grid. The computational cells are categorized into three types: inside cells, outside cells and boundary cells. Inside cells are the cells entirely contained within the domain of interest, and are discretized using standard methods such as the FVM method. Outside cells are completely outside the domain and are not considered in the solution process. Boundary cells are the cells that intersect the immersed boundary, requiring special discretization techniques to enforce boundary conditions. Two principal approaches are employed to discretize the boundary cells: direct imposition and ghost cell method, which are detailed below.

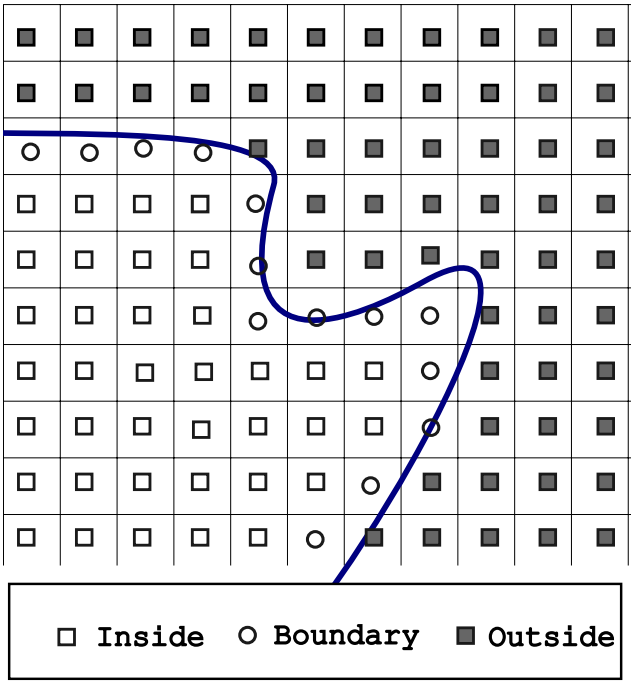



Figure 2.3 Classification of cells in discrete forcing methods: inside, outside, and boundary cells.

**Direct Imposition and Ghost Cell Approaches** Consider a simple governing equation and its discrete form given by Equation (2.9).

$$\frac{\partial \phi}{\partial t} = s(\phi), \qquad \phi_i^{n+1} = \phi_i^n + \Delta t s(\phi_i^n) \tag{2.9}$$

where $\phi$ is the variable of interest, $s$ is the source term, and $\phi_i^n$ denotes the value of $\phi$ at cell $i$ and time iteration $n$. In the direct imposition approach [75, 94–96], the value of $\phi$ at boundary cells is explicitly prescribed ($\phi_i^{n+1} = \phi_{\text{target}}$). When the center of a boundary cell coincides with the physical boundary, the target value $\phi_{\text{target}}$ can be directly set to the boundary condition value $\phi_{\text{bc}}$. More commonly, however, boundary cell centers do not align with the physical boundary. In such cases, $\phi_{\text{target}}$ is determined through interpolation using values from inside cells and boundary conditions. One of the most straightforward and widely used interpolation methods is wall distance interpolation, where $\phi_{\text{target}}$ is linearly interpolated between the boundary condition at a point B and an interpolated value at a point I located inside the domain. Figure 2.4 illustrates this interpolation method for a boundary cell P. The value at the inside point I is computed using the surrounding inside cells and many interpolation schemes have been proposed in the literature such as linear, bilinear, trilinear, quadratic, etc.

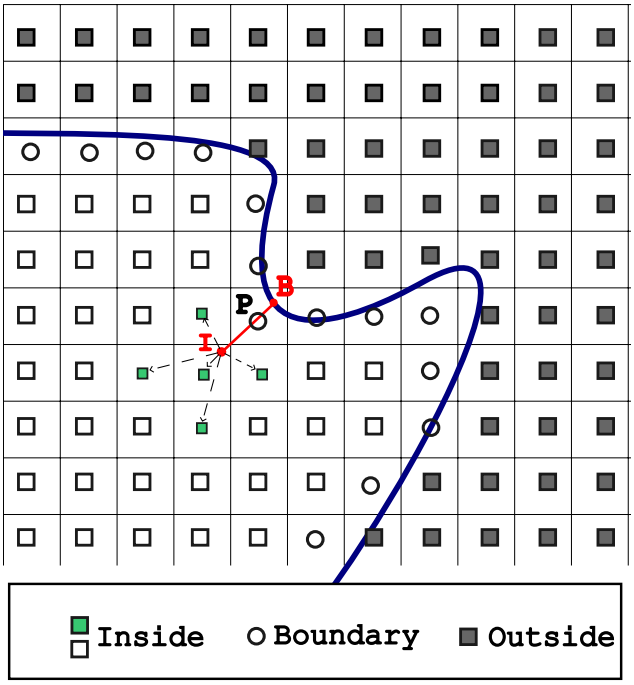



Figure 2.4 Interpolation method for the direct imposition approach.

In the ghost cell approach [76,78,97–102], the values of $\phi$ at the boundary cells are computed using the standard discretization formula in Equation (2.9), but the computational stencil is extended to include outside cells treated as "ghost cells". The value of $\phi$ at these ghost cells are determined through extrapolation schemes similar to those used in direct imposition, incorporating inside cell values and boundary conditions. For FVM discretizations, only the fluxes at boundary cell faces are needed for solution updates. Therefore, a variant of the ghost cell approach directly interpolates field values at boundary cell faces rather than at the ghost cells themselves [103].

Discrete forcing methods have been successfully applied to a wide range of applications involving both incompressible and compressible flows [66,81,104], demonstrating effectiveness in simulating flows interacting with complex geometries. When high-order interpolation schemes are employed for boundary cell discretization, these methods can lead to high-order accurate solutions at boundaries [76,105]. Since only boundary cells and interior cells require solution, the discrete forcing methods can potentially be more computationally efficient than continuous forcing methods that solve for all grid cells.

### 2.2.4 Limitations on the Implementation of the Immersed Boundary Method

Discrete forcing methods offer significant advantages in boundary accuracy while avoiding the drawbacks associated with continuous forcing approaches (such as stiffness, instabilities, reduced accuracy for rigid boundaries, and problem-dependent parameters). These benefits make discrete forcing methods particularly suitable for the types of problems addressed in this thesis. However, several challenges and limitations must be considered when implementing these methods.

The first challenge involves the robust classification of cells (tagging operation) into inside, outside, and boundary categories. This process must be performed efficiently and accurately, especially for complex or moving geometries where reclassification may be required at each time step. For moving geometries, cells that change classification (e.g., from boundary to exterior or from exterior to boundary) can introduce numerical instabilities due to their lack of solution history [76,77,106]. To address this issue, techniques such as solution reconstruction or field extension [77] can provide these cells with appropriate solution values. For cell tagging with moving geometries, computational efficiency can be improved by recognizing that only a thin layer of cells near the boundary changes classification at each time step. Alternatively, overlapping grid approaches can be employed, where classification needs to be performed only once for each grid while simultaneously providing a straightforward framework for handling the solution process with moving geometries [18].

The second challenge concerns interpolation for boundary cell discretization, where certain configurations can lead to ill-posed problems. Figure 2.5 illustrates such problematic case where the majority of interpolation schemes will fail to provide a solution at boundary cells P and Q due to insufficient data points. For complex three-dimensional geometries, ensuring that such pathological cases do not arise can be difficult, necessitating the development of robust interpolation schemes. Recent works in references [17, 18, 107] introduced an implicit interpolation approach that avoids these problematic configurations. In this method, the solution at all boundary cells is computed simultaneously by solving a coupled system of equations. This allows the interpolation stencil for a given boundary cell to include other boundary cells, which effectively resolves the issue of insufficient data points, even in pathological cases like the one shown in Figure 2.5. This technique has proven effective for 2D geometries with both Laplace and Euler equations.

The third and significant challenge for applications involving inviscid compressible flows is maintaining conservation of mass, momentum, and energy. Since boundary cells are typically discretized using non-conservative approaches, the resulting numerical solutions will violate conservation principles. This lack of conservation can lead to significant inaccuracies in the

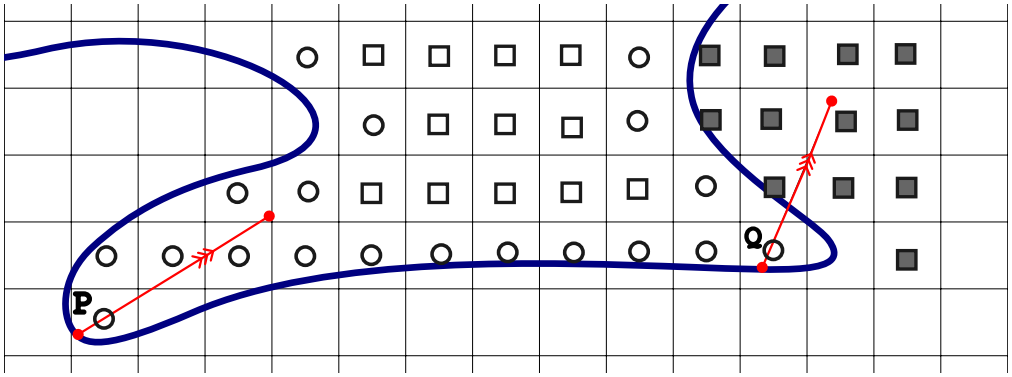



Figure 2.5 Pathological case for the interpolation of the solution at boundary cells P and Q. The geometry is exaggerated for illustrative purposes.

numerical solution. For hyperbolic conservation laws, such as the Euler equations, failure to maintain conservation at the discretization level can lead to incorrect predictions of shock wave positions and other flow discontinuities [108–110].

### 2.2.5 Immersed Boundary Method Literature Review Summary

This section has provided a literature review of the Immersed Boundary Method (IBM), a numerical technique that fundamentally simplifies the mesh generation process for fluid flows simulations in complex geometries by immersing the physical geometry into a non-body-fitted Cartesian grid and adapting the numerical solver to account for the boundary's presence. Two main approaches are used to enforce boundary conditions in the IBM framework: continuous forcing methods and discrete forcing methods. Continuous forcing methods introduce a source term into the governing equations that model the fluid-boundary interactions. Examples of methods in this category include the classical method of Peskin, which is designed for elastic boundaries, and penalization methods, that model the solid as a porous medium with very low permeability. These approaches require accurate modeling of fluid-solid interactions, which presents challenges for rigid bodies where problem-dependent parameter tuning is necessary and the resulting source terms may introduce numerical stiffness. Discrete forcing methods, on the other hand, directly modify the discretization of grid cells adjacent to the immersed boundary to enforce boundary conditions. In this approach, cells are classified as inside, outside, or boundary cells. The solution in boundary cells is determined from an interpolation (reconstruction) using values from neighboring inside cells and boundary conditions, implemented either through direct imposition or by the ghost-cell approach. Discrete forcing methods can achieve higher-order accuracy when used in conjunction with high-order reconstruction techniques and are generally more computationally efficient than continuous

forcing methods. However, they face several implementation challenges. These include robust and efficient cell classification, particularly for complex or moving geometries; Interpolation robustness, which recent implicit interpolation approaches have addressed in 2D; and most importantly, maintaining conservation of mass, momentum, and energy inside the domain, which is crucial for accurate simulations, especially in capturing shock waves and other flow discontinuities.

The Immersed Boundary Method, particularly through discrete forcing techniques, offers a compelling alternative to traditional body-fitted methods by automating and simplifying mesh generation for complex geometries. However, its application requires careful consideration of the challenges related to boundary treatment to ensure accuracy, robustness, and conservation properties in numerical simulations.

This concludes the review of the IBM method. The next chapter will present the objectives and contributions of this thesis.

# CHAPTER 3 OBJECTIVES AND CONTRIBUTIONS OF THE THESIS

This chapter presents the general and specific objectives of the thesis and summarizes its key contributions.

## 3.1 Objectives

The literature review presented in Chapter 2 has demonstrated that the IBM method with Cartesian grids is well-suited for the numerical simulation of compressible flows in complex geometries, such as high-voltage circuit breakers. However, several challenges must be addressed before the IBM method can effectively simulate flows in HVCB applications. These challenges include: preserving conservation of mass, momentum and energy in the discretization of the boundary cells; resolving robustness issues related to the boundary conditions imposition; achieving high-order accuracy to capture complex flow features in 3D geometries using a minimal number of cells; and developing efficient algorithms for geometry definition and mesh generation to handle complex 3D geometries. These considerations lead to the following objectives:

### General Objective

"Develop *conservative* and *second-order accurate* Cartesian grid methods for compressible flows applicable to complex *three-dimensional* geometry simulations."

### Specific Objectives

- **S.O.1** :"Develop conservative methods for compressible flows based on the Immersed Boundary Method."
- **S.O.2** : "Implement a second-order accurate Cartesian grid method for compressible flows simulation."
- **S.O.3** : "Develop efficient mesh generation techniques for complex three-dimensional geometries."
- **S.O.4** : "Verify the numerical methods on academic test cases and on simplified HVCB geometry."

## 3.2 Contributions

The thesis has been structured around the general and specific objectives presented above. The following four contributions have been made to achieve these objectives:

- **Chapter 4** (Article 1) : Conservative Immersed Boundary Methods for compressible flows in 2D using cut-cells. (*S.O.1*, *S.O.4*)
- **Chapter 5** (Article 2) : Second-order accurate Cut-Cells method for compressible flows in 2D. (*S.O.2*, *S.O.4*)
- **Chapter 6** : Robust second-order accurate Immersed Boundary Method for inviscid fluid flows simulation in three-dimensions. (*S.O.2*, *S.O.4*)
- **Appendices A, B** : Detailed description of the geometry and mesh generation techniques in three-dimensions. (*S.O.3*)

The contents of these contributions are briefly summarized in the following sections.

### 3.2.1 Summary of Chapter 4 (Article 1)

The first article addresses the conservation challenges inherent in the IBM method. It develops numerical methods that guarantee the conservation of mass, momentum, and energy when implementing IBM approaches. The methodology employs cut-cells to precisely define the computational domain where conservation laws must be satisfied. In this approach, boundary cells are discretized using IBM techniques for boundary condition imposition. This strategy avoids the small cell stability problem associated with traditional Cut-Cells methods but introduces a loss of conservation for the conserved quantities. To rectify this issue, an additional conservative step is incorporated into the time integration process, restoring conservation properties. The article's main contribution is the introduction of three novel methods, namely Flux Redistribution Method (FRM), Flux Interpolation Method (FIM) and Flux Correction Method (FCM), that effectively recover conservation. These methods are verified through academic test cases that demonstrate their effectiveness in preserving conservation properties and illustrate the detrimental impact that conservation loss can have on numerical results. Additionally, the study examines how the conservative correction step affects numerical accuracy, revealing that it does not compromise the base scheme's accuracy.

Since this article focuses specifically on the conservation problem, the numerical results are limited to two-dimensional geometries using first-order accurate schemes. However, the

methodologies developed are readily applicable to three-dimensional geometries, as they employ a flux formulation that is dimension-independent. While the approach can theoretically be extended to higher-order schemes, further investigation is necessary, as existing literature indicates that many Cut-Cells methods struggle to achieve high-order accuracy at boundary cells. This high-order extension forms the subject of the second article.

### 3.2.2 Summary of Chapter 5 (Article 2)

The second article explores the development of second-order accurate Cut-Cells methods for compressible flows, building upon the conservative methods introduced in the first article. The primary contribution is the implementation of high-order interpolation schemes for IBM boundary condition imposition, which simultaneously addresses the small cell problem and achieves second-order accurate solutions throughout the entire domain, including boundary regions. The Method of Manufactured Solutions (MMS) [111–113] is employed to rigorously assess the numerical accuracy of the scheme. Results confirm that the approach achieves second-order accuracy both in the interior domain and at boundary regions when analyzed separately. The article also presents complex flow test cases, including a simplified two-dimensional HVCB geometry, demonstrating the scheme's effectiveness in capturing important flow features.

The first two articles have focused on developing numerical methods that address conservation challenges and high-order accuracy requirements for compressible flow simulations using Cartesian grid methods. These approaches have been successfully tested on two-dimensional geometries, demonstrating their ability to maintain conservation properties and achieve second-order accuracy. The next logical progression is to study the extension of these methods to three-dimensional geometries, which is addressed in the subsequent chapter and appendices.

### 3.2.3 Summary of Chapter 6 and Appendices A and B

Appendices A and B provide detailed descriptions of the geometry definition and mesh generation techniques for implementing the IBM method in three-dimensional geometries. The geometry definition employs a boundary representation (B-rep) approach using surface triangulation. The Cartesian grid generation algorithm takes this geometry as input and constructs a Cartesian grid around it. The grid is organized as an octree structure, with refinement criteria based on geometric features to selectively refine regions of interest. The grid generation process produces a mesh with cells classified into inside, outside, and boundary cells. Boundary cells contain additional geometric information necessary for boundary

condition application.

Chapter 6 presents the development of a robust, second-order accurate IBM method for compressible flows in three-dimensional geometries. It proposes an extension of the implicit reconstruction approach introduced in [17, 107] from two dimensions to three dimensions. This extension effectively handles boundary cells while avoiding pathological cases commonly encountered in sharp interface methods for boundary condition implementation. The method is verified through academic test cases and applied to a simplified three-dimensional HVCB geometry, demonstrating its efficacy for complex three-dimensional flow simulations. Convergence studies confirm that the scheme achieves second-order accuracy throughout the domain, including boundary regions.

The remainder of this thesis presents these contributions in detail and concludes with a summary of the work, its limitations, and perspectives for future research.

# CHAPTER 4 ARTICLE 1 : CONSERVATIVE IMMERSED BOUNDARY METHODS ON CARTESIAN GRIDS FOR INVISCID COMPRESSIBLE FLOWS SIMULATION

El Hadji Abdou Aziz Ndiaye, Jean-Yves Trépanier, Renan De Holanda Sousa, Sébastien Leclaire, International Journal of Heat and Fluid Flow, Volume 114, 27 February 2025, https://doi.org/10.1016/j.ijheatfluidflow.2025.109775.

Small modifications have been made to the original article to fit the format of this thesis.

**Abstract**

This work introduces three conservative methods based on the Immersed Boundary Method. These methods make use of cut-cells to ensure the conservation properties in the numerical solution. However, since some cut-cells can be very small, they can significantly restrict the time step of an explicit time integration scheme. To circumvent this limitation, a semi-implicit treatment of the small cells is employed. The first method relies on a straightforward flux redistribution procedure that globally restores conservation on the cut-cells grid. The other two methods employ the local conservative discretization form of the finite volume method, along with optimization procedures, to ensure local conservation of the numerical solution within each cell. These methods have been tested on two-dimensional inviscid compressible flow problems, demonstrating results comparable to those obtained with the standard Cut-Cells method in terms of accuracy and conservation. Furthermore, the methods are stable and can be effectively used with an explicit time integration scheme without encountering any stability issues related to the small cut-cells.

Keywords : Conservation; Immersed Boundary Method; Cut-cells; Cartesian grid; Compressible flow

## 4.1 Introduction

Multiphysics simulations of compressible flows involving complex geometries and moving bodies present significant challenges. A critical step in developing simulation codes for such applications is mesh generation. Over the years, several approaches have been proposed to address these challenges, including unstructured grids, overset grids, and Cartesian grids. Unstructured grids are highly flexible and can accommodate complex geometries. However, generating high-quality unstructured grids can be difficult, and they often require remeshing

**Nomenclature**

| | |
|---|---|
| $A_f$ | Area of face $f$ |
| $E$ | Total energy |
| $\mathbf{F}$ | Flux vector |
| $\mathbf{F}_f$ | Numerical fluxes at face $f$ |
| $h$ | Average cell size |
| $\mathbf{n}$ | Normal vector |
| $n_x, n_y$ | Components of the normal vector |
| $p$ | Pressure |
| $S(i)$ | Stencil of neighbors of cell $i$ |
| $u, v$ | Velocity components |
| $\mathbf{U}$ | Vector of conservative variables |
| $\mathbf{v}$ | Velocity vector |

*Greek letters*

| | |
|---|---|
| $\alpha_i$ | Cut-cell volume fraction of cell $i$ |
| $\partial\Omega$ | Boundary of the computational domain |
| $\partial V_i$ | Set of faces of cell $i$ |
| $\Delta M_i$ | Conserved Mass Error at cell $i$ |
| $\Delta t^n$ | Time step at the $n$th time iteration |
| $\Delta V_i$ | Volume of cell $i$ |
| $\epsilon_{L_1}, \epsilon_{L_2}, \epsilon_{L_\infty}$ | $L_1$, $L_2$, and $L_\infty$ error norms |
| $\gamma$ | Ratio of specific heats |
| $\rho$ | Density |
| $\omega$ | Weights for flux redistribution |
| $\Omega$ | Computational domain |

*Abbreviations*

| | |
|---|---|
| CC | Cut-Cells |
| FCM | Flux Correction Method |
| FIM | Flux Interpolation Method |
| FRM | Flux Redistribution Method |
| FVM | Finite Volume Method |
| IBM | Immersed Boundary Method |
| RCCM | Reconstructed Cut-Cells Method |
| `BC-Cell` | Boundary cut-cell |
| `BI-Cell` | Boundary inside cell |
| `C-Cell` | Conserved cell |
| `I-Cell` | Inside cell |
| `N-Cell` | Neighbor (of a reconstructed) cell |
| `NR-Cell` | Non-reconstructed cell |
| `OC-Cell` | Outer cut-cell |
| `R-Cell` | Reconstructed cell |

for problems involving moving boundaries. The overset grids approach can circumvent the need for remeshing in moving boundary problems, but it necessitates the generation of the basic grids that will be superimposed to form the overset grids, which can be time-consuming. Additionally, the data structures required to couple these superimposed grids can be complex. On the other hand, Cartesian grids are very easy to generate and can be used for moving boundaries. However, special treatment is necessary for the discretization of cells near the boundaries to account for their presence. In the context of moving boundaries, Cartesian grids can be directly used as a background grid. Furthermore, due to their simple structure, Cartesian grids can be easily coupled with other Cartesian grids to form a set of overset grids. This work will focus on the Cartesian grids approach due to their simplicity and greater potential for automation.

When using a Cartesian grid, one encounters an initial problem: the cells near the boundaries are typically not aligned with the boundaries. To address this issue, the boundary cells can be approximated using a staircase representation of the actual boundary. However, this approach is not very accurate and requires a very fine mesh to achieve a good approximation of the boundary. An alternative approach is the Cut-Cells (CC) method [21, 31], which involves clipping the cells intersected by the boundary, resulting in a piecewise linear approximation

of the boundary. The discretization on the cut-cells grid is straightforward by using the usual finite volume method (FVM) for unstructured grids. The CC method has the advantage of being fully conservative, but it has two main disadvantages. The first disadvantage is that generating the cut-cells grid in three dimensions is challenging, with high computational costs and potential robustness issues. Additionally, for problems involving moving boundaries, the list of cut-cells must be recalculated at each time step, exacerbating these robustness issues. The second, and perhaps most significant, disadvantage is that the geometry in the CC method can arbitrarily cut the grid cells, leading to the formation of very small cells. These small cells require a very small time step to satisfy the CFL condition of explicit time integration schemes, significantly increasing the computational time of the simulation. This issue is known in the literature as the small cell problem, and various methods have been devised to address it.

One of the earliest techniques proposed to handle the small cell problem is the cell-merging method [35, 37], which involves merging small cells with their larger neighbors. While this method is conceptually simple, it is challenging to implement robustly in three dimensions. Specifically, the selection of neighboring cells for merging is often not systematic [41]. A variation of this approach is the cell-linking method [41–43], which links small cells with their larger neighbors and treats them as a single cell in the discretization process without actually merging them. This method circumvents the complexity associated with the merging procedure. Another technique is the Flux Redistribution Method, which redistributes the fluxes between small cells and their larger neighbors to satisfy the stability condition of the small cells. Collela *et al.* [52] employed this method for simulating compressible flows on cut-cells grids. Although the Flux Redistribution Method is relatively easy to implement, it can result in a loss of accuracy at the boundaries [59]. Other methods, such as the h-box method [57, 58] and the state redistribution method [44], have also been proposed in the literature to overcome the small cell problem. Each of these techniques has its own advantages and disadvantages, and their suitability can vary depending on the specific application.

Instead of employing the Cut-Cells (CC) approach, one can utilize the Immersed Boundary Method (IBM) [60] to handle complex geometries. The IBM method discretizes the equations on the Cartesian grid by applying a special treatment to the cells near the boundaries, thereby accounting for the presence of the boundaries. Mittal and Iaccarino [81] have provided an extensive review of the IBM method and the various techniques used to treat boundaries within this framework. Among these techniques, the sharp interface method [66, 76, 78, 89, 94] is one of the most popular. This method employs an interpolation procedure to impose boundary conditions on the cells near the boundaries. By controlling the accuracy of the interpolation procedure, one can obtain the desired accuracy of the solution at the boundaries.

While the imposition of boundary conditions is generally more complex for the IBM method compared to the CC method, the IBM method offers the significant advantage of not imposing any restrictions on the time step when using an explicit time integration scheme. Additionally, the IBM method is well-suited for problems involving moving boundaries. However, one of the main disadvantages of the IBM method is the difficulty in ensuring numerical conservation at the boundaries.

In this work, we propose three methods that combine the advantages of both the Cut-Cells (CC) and Immersed Boundary Method (IBM) in terms of conservation, accuracy, and stability while avoiding their respective disadvantages. To ensure the conservation of the numerical solution, the cut-cells grid is used to discretize the equations, but a reconstruction procedure similar to the one used in the IBM method is employed to impose the boundary conditions. This reconstruction procedure allows to obtain an accurate solution at the boundaries without imposing any restrictions on the time step. However, the IBM reconstruction introduces a loss of conservation, necessitating an additional step called conservative correction step to ensure the conservation of the various conservative variables. The main contribution of this work is the development of this conservative correction step that restores conservation on the cut-cells grid when using the IBM method for boundary condition treatment. The first method relies on a straightforward flux redistribution procedure that globally restores conservation on the cut-cells grid. The other two methods employ newly developed optimization procedures to ensure local conservation of the numerical solution within each cell. These methods have been tested on two-dimensional inviscid compressible flow problems, demonstrating results comparable to those obtained with the standard Cut-Cells method in terms of accuracy and conservation. Furthermore, the methods are stable and can be effectively used with an explicit time integration scheme without encountering any stability issues related to the small cut-cells.

The paper is organized as follows: Section 4.2 presents the governing equations for inviscid compressible flow and the finite volume discretization of these equations using the Cut-Cells (CC) method. It then details the Immersed Boundary Method (IBM) and the Reconstructed Cut-Cells Method (RCCM), which is a combination of the CC and IBM methods without the conservative correction step. Section 4.3 introduces the methods that employ a correction step that restore conservation on the cut-cells grid when using the IBM method for boundary condition treatment. Section 4.4 presents numerical results demonstrating the accuracy and conservation properties of the different methods using four test cases. Finally, Section 4.5 concludes the paper.

## 4.2 Numerical Method

The goal of this section is to present the governing equations for inviscid compressible flow, described by the Euler equations and their finite volume discretization using the Cut-Cells (CC) method and the Immersed Boundary Method (IBM). It is divided into three parts. The first one briefly presents the Euler equations that govern the inviscid compressible flow. The second part details the finite volume discretization of these equations using the Cut-Cells (CC) method. Finally, the third part introduces the Immersed Boundary Method (IBM) and a combination of the CC and IBM methods called the Reconstructed Cut-Cells Method (RCCM). These three methods will serve as the basis for the development of the new conservative methods presented in Section 4.3.

### 4.2.1 Governing Equations

The integral form of the Euler equations, which encapsulate the conservation laws of mass, momentum, and energy, is given by Equation 4.1.

$$\frac{\partial}{\partial t}\int_{\Omega}\mathbf{U}d\Omega + \oint_{\partial\Omega}\mathbf{F}(\mathbf{U})dA = 0 \tag{4.1}$$

Here, $\mathbf{U}$ denotes the vector of conservative variables, and $\mathbf{F}$ represents the flux vector along the outward unit normal $\mathbf{n}$ to the boundary. The conservative variables and the flux vector are defined in Equation 4.2.

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix} \qquad \mathbf{F} = \begin{bmatrix} \rho(\mathbf{v}\cdot\mathbf{n}) \\ \rho u(\mathbf{v}\cdot\mathbf{n}) + pn_x \\ \rho v(\mathbf{v}\cdot\mathbf{n}) + pn_y \\ \rho E(\mathbf{v}\cdot\mathbf{n}) + p(\mathbf{v}\cdot\mathbf{n}) \end{bmatrix} \tag{4.2}$$

where $\rho$ is the density, $\mathbf{v} = (u, v)$ is the velocity vector, $p$ is the pressure and $E$ is the total energy. The system of equations is closed using the perfect gas relation, given by Equation 4.3.

$$p = (\gamma - 1)\left(\rho E - \frac{1}{2}\rho\|\mathbf{v}\|^2\right) \tag{4.3}$$

where $\gamma$ is the ratio of specific heats.

### 4.2.2 Finite Volume Discretization of the Euler Equations

This section details the finite volume discretization of the Euler equations using the Cut-Cells (CC) method and discusses the conservation properties of the method.

From the finite volume method (FVM) viewpoint, the integral form of the Euler equations can be discretized as shown in Equation 4.4.

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\Delta t^n}{\Delta V_i} \sum_{f \in \partial V_i} \mathbf{F}_f A_f \tag{4.4}$$

where:

- $\mathbf{U}_i^n$ is the vector of conservative variables at the cell $i$ at the $n$th time step.
- $\Delta t^n$ and $\Delta V_i$ are the time step and the volume of the cell $i$, respectively.
- $\mathbf{F}_f$ is the numerical flux at the face $f$, shared by the two cells.
- $A_f = \int_f dA$ is the area of the face $f$.

The numerical flux vector $\mathbf{F}_f$ is an approximation of the flux vector $\mathbf{F}$ at the face $f$:

$$\mathbf{F}_f(\mathbf{U}_{i_1}, \mathbf{U}_{i_2}, \ldots) \approx \frac{1}{A_f} \int_f \mathbf{F}(\mathbf{U}) dA \tag{4.5}$$

where $\mathbf{U}_{i_1}, \mathbf{U}_{i_2}, \ldots$ are the conservative variables at the cells surrounding face $f$.

The discretization form of Equation 4.4 is known as the *conservative discretization form* of the Euler Equations 4.1. Any method that can be written in this form is considered *conservative*. Using Equation 4.4, it can be shown that the sum of all differences between the conservative variables at time steps $n+1$ and $n$ depends only on the fluxes at the domain boundaries $\partial\Omega$, as shown in Equation 4.6. In particular, if there is no flux at the boundaries, the sum of all conservative variables remains constant over time. A method that respects the property given by Equation 4.6 is said to be *globally conservative*.

$$\sum_{i \in \Omega} \Delta V_i \mathbf{U}_i^{n+1} = \sum_{i \in \Omega} \Delta V_i \mathbf{U}_i^n - \Delta t^n \sum_{f \in \partial\Omega} \mathbf{F}_f A_f \tag{4.6}$$

Having a conservative method is crucial as it preserves the conservation properties derived from physical laws in the numerical solution. It also allows for the correct capture of shocks and other discontinuities in the solution of the Euler equations [1, 114].

In this work, the approximate Riemann solver of Roe [115] is used to compute the numerical fluxes $\mathbf{F}_f$ at the faces $f$.

The CC solver is directly based on the FVM discretization described above. First, the cut-cells grid is generated by clipping the cells intersected by the boundary. Then, the FVM discretization is applied to the cut-cells grid. Figure 4.1 shows an example of the cut-cells grid used by the CC solver. Ghost cells are added at the cut-cells to compute the numerical fluxes at the boundaries [116]. Since no special treatment such as cell-merging or flux redistribution is applied to the small cells, the time step $\Delta t^n$ can be very small to satisfy the CFL condition. Nevertheless, the CC solver is accurate and conservative. The only other potential downside is the deterioration of mesh quality near the boundaries due to the arbitrary shape of the cut-cells, which can affect the accuracy of the FVM solver.

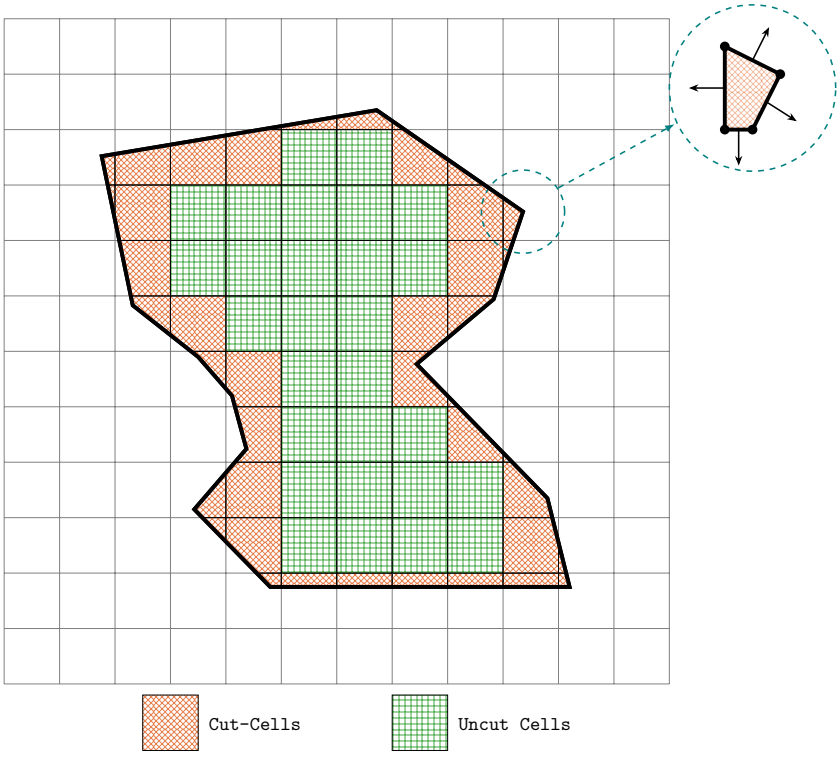



Figure 4.1 Illustration of the cut-cells grid used by the CC solver. The dashed encirclement shows a detailed view of a cut-cell.

The next section will present the IBM method and the Reconstructed Cut-Cells Method (RCCM) method, which is an approach proposed by the authors in reference [1] that combines the advantages of the CC and IBM methods.

### 4.2.3 Immersed Boundary Method

This section[1] introduces the IBM and RCCM methods, which are similar to the CC method but use different grids and a reconstruction procedure to impose the boundary conditions. It is divided into three parts. The first part presents the IBM solver with a focus on the general idea of the method along with the grid used to discretize the equations. The second part presents the details of the reconstruction procedure used by the IBM solver to impose the boundary conditions. Finally, the third part introduces the RCCM solver, which combines the advantages of the CC and IBM methods.

#### IBM solver

The IBM solver is also based on the FVM discretization described in section 4.2.2. However, for the cells near the boundaries, a reconstruction procedure is used to impose the boundary conditions. The reconstruction procedure will be summarized in the following section. But first, the grid used by the IBM solver will be described.

The IBM solver only considers (solves) the cells that have their centers inside the domain $\Omega$. The cells that have their center outside the domain $\Omega$ are not solved by the solver. These cells are categorized into two groups:

- The cells that are completely outside the domain $\Omega$.
- The cells named outer cut-cells (`OC-Cells`) that are partially inside the domain $\Omega$ but have their centers outside the domain $\Omega$.

Although the `OC-Cells` are not solved by the solver, they are used in the reconstruction procedure since they contain some information about the boundary conditions.

Regarding the solved cells, they are also categorized into two subgroups: the boundary cells (`B-Cells`) and the inside cells (`I-Cells`). The `I-Cells` are completely inside the domain $\Omega$ and do not have any `OC-Cells` as neighbors. The `B-Cells` are the remaining unidentified cells and fall into one of the following two classes:

- The cells that are intersected by the boundary $\partial\Omega$ and have their center inside the domain $\Omega$. These cells can be called boundary cut-cells (`BC-Cells`).
- The cells that are completely inside the domain $\Omega$ and have at least one neighbor that is an `OC-Cell`. These cells can be called boundary inside-cells (`BI-Cells`).

[1] The content of this section is based on the conference paper [1] and is presented here for the sake of completeness.

The `B-Cells` form then a watertight approximation of the boundary $\partial\Omega$. Figure 4.2 shows an example of the grid used by the IBM solver. The IBM grid can be efficiently generated without explicitly clipping the cells with the boundary. Indeed, the reconstruction procedure used by the IBM solver to impose the boundary conditions does not need the explicit geometric definition of the boundary cells.

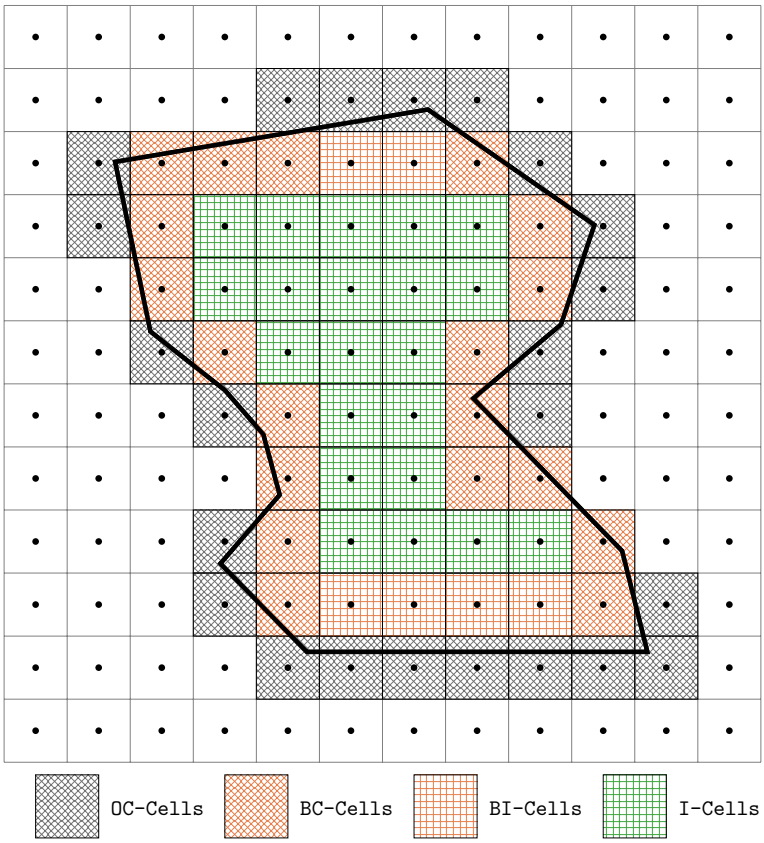



Figure 4.2 Illustration of the different types of cells in the IBM grid.

The `I-Cells` are defined such that they have a complete stencil of neighbors solved by the IBM solver. Thus, the `I-Cells` are directly discretized using the FVM discretization described above. The `B-Cells`, on the other hand, are solved using the reconstruction procedure summarized below. The IBM solver functions then as follows:

- The `I-Cells` are advanced in time using the FVM discretization: $\mathbf{U}^{n}_{\texttt{I-Cells}} \rightarrow \mathbf{U}^{n+1}_{\texttt{I-Cells}}$.
- The `B-Cells` are reconstructed implicitly using the boundary conditions (BC) and the new values of the `I-Cells`:

$$\mathbf{U}^{n+1}_{\texttt{B-Cells}} = \texttt{Reconstruction}(\mathbf{U}^{n+1}_{\texttt{B-Cells}}, \mathbf{U}^{n+1}_{\texttt{I-Cells}}, \text{BC}) \tag{4.7}$$

**Reconstruction Procedure**

The reconstruction procedure, as described in Equation 4.7, performs an interpolation for the primitive variables $\mathbf{W} = (\rho, u, v, p)$ of the `B-Cells` using the updated values of the `I-Cells` and the boundary conditions. The interpolation is carried out using the following steps:

1. For each primitive variable $\phi$ ($\phi = \rho, u, v$ or $p$), it is assumed that the variable $\phi$ is approximated by a linear function around the cell center of the `B-Cell`:

$$\phi = a + b(x - x_0) + c(y - y_0) \tag{4.8}$$

   where $(x_0, y_0)$ is the center of the `B-Cell` to be reconstructed.

2. Now, the coefficients $(a, b, c)$ for each primitive variable and each boundary cell are to be determined. To do that, a set of points (called reconstruction stencil) around the cell center of each boundary cell, where some information about the primitive variables is known, is used. This information can be the value of the primitive variable at the center of an `I-Cell` or another `B-Cell`, or a Dirichlet boundary condition. For points on a boundary with a Neumann boundary condition (e.g., "no-slip" or "supersonic outlet"), the information is the derivative of the primitive variable along the normal direction to the boundary.

   Typically, the reconstruction stencil includes the cell centers of solved cells (`I-Cells` and `B-Cells`) that are neighbors of the current boundary cell, and a set of points on the boundary close to the cell center of the current boundary cell. Figure 4.3 shows an example of a reconstruction stencil for a boundary cell, where the points on the boundary are the closest points to the centers of the current `B-Cell` and its `OC-Cells` neighbors.

3. Once the reconstruction stencil is defined, small linear systems are assembled for each `B-Cell` in order to find the coefficients of the polynomial.

4. Since the stencil of a `B-Cell` can contain others `B-Cells`, all the systems are assembled into one global system that is solved to find the coefficients of the polynomial for all the `B-Cells` simultaneously.

The reconstruction procedure is very flexible and can be used with any type of boundary conditions. Since it is not based on the conservative discretization of Equation 4.4, it will introduce therefore a loss of conservation. Hence, the IBM solver is not conservative.

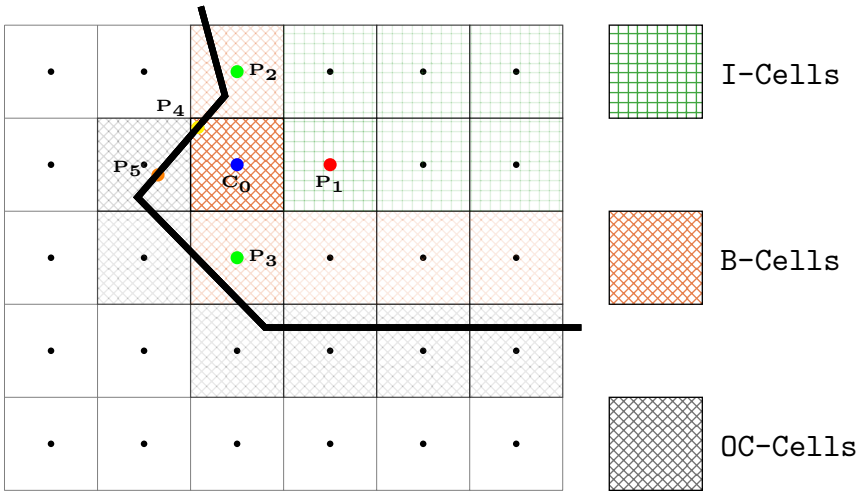



Figure 4.3 Illustration of a reconstruction stencil $\{P_i\}$ of a boundary cell $C_0$.

### Reconstructed Cut-Cells Method

The Reconstructed Cut-Cells Method (RCCM) solver is a combination of the CC and IBM solvers. The idea consists of using the reconstruction procedure of the IBM solver on the cut-cells for the purpose of avoiding the small cell problem.

The RCCM method uses the cut-cells grid shown in Figure 4.1 for discretization. The cells are, this time, categorized into two groups: the non-reconstructed cells (`NR-Cells`), which are discretized using the FVM discretization, and the reconstructed cells (`R-Cells`). The `R-Cells` are defined as the clipped version of the `OC-Cells` in the IBM grid shown in Figure 4.2. This choice is motivated by the fact that the set of `OC-Cells` contains all the small cells and is, on average, half the size of the set of all cut-cells. Once the reconstructed cells are identified, the remaining cells are the `NR-Cells`, which are discretized using the FVM discretization. Figure 4.4 shows the grid used by the RCCM solver.

In summary, the RCCM functions as follows:

1. The non-reconstructed cells are advanced in time using the FVM discretization:

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\Delta t^n}{\Delta V_i} \sum_{f \in \partial V_i} \mathbf{F}_f A_f \qquad \text{for } i \in \texttt{NR} - \texttt{Cells} \tag{4.9}$$

2. The reconstructed cells are reconstructed using the boundary conditions (BC) and the

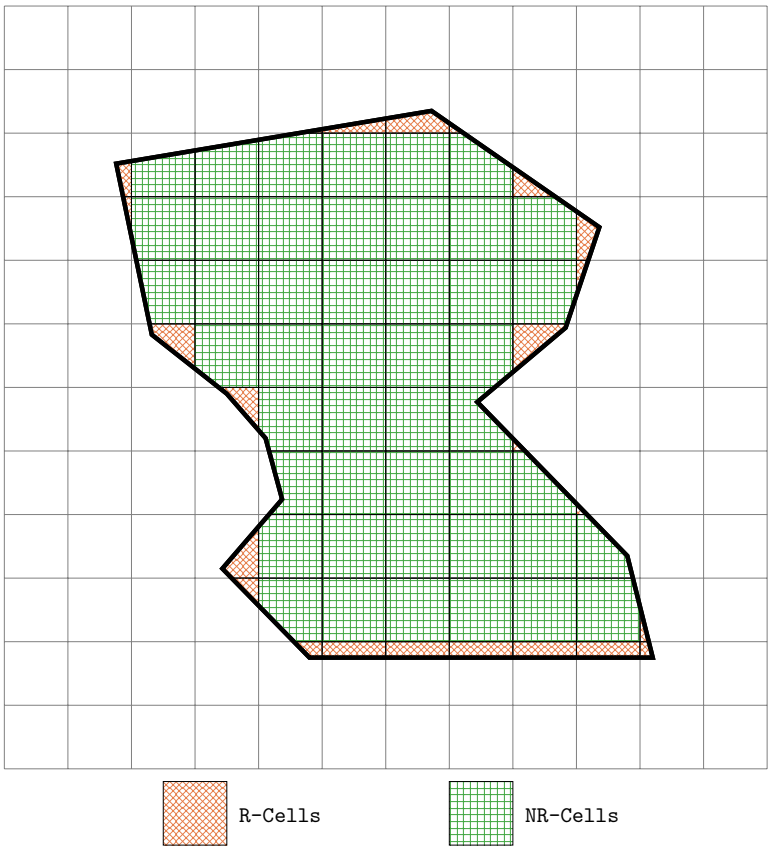



Figure 4.4 Illustration of the RCCM grid.

new values of the non-reconstructed cells:

$$\mathbf{U}_i^{n+1} = \texttt{Reconstruction}(\mathbf{U}^{n+1}_{\texttt{R-Cells}}, \mathbf{U}^{n+1}_{\texttt{NR-Cells}}, \text{BC}) \qquad \text{for } i \in \texttt{R} - \texttt{Cells} \tag{4.10}$$

The reconstruction step described in Equation 4.10 is similar to the one used by the IBM solver. Here, the centroid of the `R-Cells` is used as the center of the reconstruction and the boundary points of the reconstruction stencil are chosen to be the centers of the boundary faces composing the cut-cells. In the RCCM solver, the reconstruction procedure can be viewed as a semi-implicit treatment of the reconstructed cells that will relieve the time step restriction due to the small cells. As in the IBM solver, the reconstruction procedure will also induce a loss of conservation in the RCCM solver. However, since the number and the volume occupied by the reconstructed cells of the RCCM solver are in general smaller than those in the IBM solver, the errors associated with the loss of conservation are expected to be smaller. This is indeed the case, as shown in the numerical results presented in reference [1] and in section 4.4 of this work.

In this section, the governing equations for inviscid compressible flow and their finite volume discretization using the CC, IBM, and RCCM methods have been presented. All three methods divide the cells of the computational domain into two sets that are solved differently. The CC method uses the finite volume discretization directly on all the cells of the grid making it fully conservative but affected by the small cell problem. The IBM method does not consider cut-cells and uses a reconstruction procedure instead to impose the boundary conditions on the cells near the boundaries. The RCCM method combines the advantages of the CC and IBM methods by using the cut-cells grid for discretization and the reconstruction procedure of the IBM method to impose the boundary conditions. The IBM and RCCM methods are not conservative due to the reconstruction procedure but are not affected by the small cell problem. The next section will present various methods based on the RCCM solver that allow to fully restore the conservation properties on the numerical solutions.

## 4.3 Conservative Methods

This section presents three methods designed to restore the conservation properties on the numerical solutions. It is divided into four parts. The first part presents the general description of the conservative methods along with some terminology and the grid used by these methods. The following three parts present the three conservative methods. All these methods are based on the RCCM solver with an additional conservative correction step applied after each iteration of the RCCM solver's time integration. The first method relies on a straightforward flux redistribution procedure that globally restores conservation on the cut-cells grid. The flux redistribution technique used in this method was proposed by [51] and [52]. The last two methods correspond to new innovative approaches developed by the authors that ensure local conservation of the numerical solution within each cell by solving an optimization problem.

### 4.3.1 Overview of the Conservative Methods

#### Grid Description

The conservative methods will use the same grid as the RCCM solver. Hence, the cells are initially divided into two sets: the non-reconstructed cells (`NR-Cells`) and the reconstructed cells (`R-Cells`). But, to apply the conservative correction step that restore the conservation, the `NR-Cells` will be further separated into two subsets:

1. The cells that have at least one `R-Cells` as neighbor. These cells are called the neighbor of the reconstructed cells (`N-Cells`).

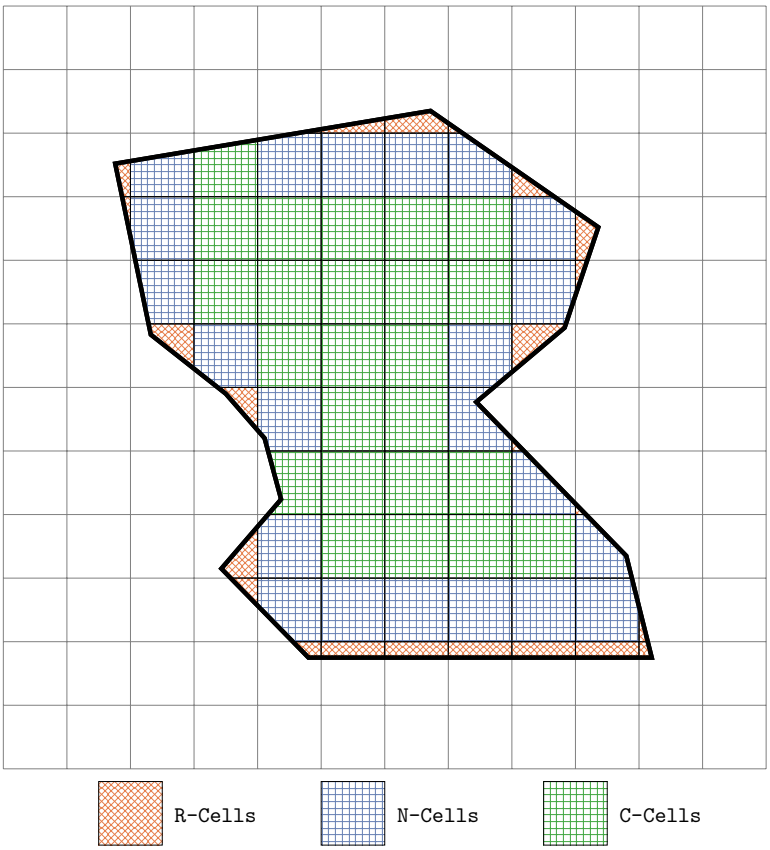



Figure 4.5 Illustration of the different types of cells in the grid used by the conservative methods.

2. The remaining cells that are not `NR-Cells` are called conserved cells (`C-Cells`).

Figure 4.5 illustrates an example of the grid used by the conservative methods.

Since `N-Cells` share a face with at least one `R-Cell`, they are affected by the loss of conservation introduced by the reconstruction of the `R-Cells`. Therefore, the solutions for these cells typically needs correction alongside the solution for the `R-Cells`. Conversely, `C-Cells` are not impacted by the reconstruction of the `R-Cells`, and their solutions does not require any modification during the conservative correction step.

### Notation

Table 4.1 summarizes the different notations used in the following sections to describe the different conservative methods.

Table 4.1 Summary of the different notations used to describe the conservative methods.

| Notation | Description |
|---|---|
| $\mathbf{U}$ | The vector of conservative variables corresponding to the current solution. |
| $\mathbf{U}_{cc}$ | The conservative solution obtained with the CC solver. |
| $\mathbf{U}_{nc}$ | The non-conservative solution obtained with the RCCM solver. |
| $\mathbf{U}_{c}$ | The new conservative solution obtained with the conservative methods. |
| $\mathbf{F}_f$ | The fluxes at face $f$ obtained with the FVM discretization using the current solution $\mathbf{U}$. |
| $\mathbf{F}_{m,f}$ | Modified fluxes at face $f$ used to obtain the conservative solution $\mathbf{U}_c$. |

### General Description of the Conservative Methods

The various conservative methods operate in a similar manner. The objective is to update the current solution $\mathbf{U}^n$ to obtain a new conservative solution $\mathbf{U}^{n+1}$.

First, the new time step $\Delta t^n$ is computed without taking into account the `R-Cells` as shown in Equation 4.11.

$$\Delta t^n = \text{CFL} \min_{i \notin \texttt{R-Cells}} \Delta t_i^n \tag{4.11}$$

Next, a conservative solution $\mathbf{U}_{cc}^{n+1}$ is calculated using the CC solver with Equation 4.12. Since the `R-Cells` were ignored in the computation of the time step, the CFL condition is not satisfied for the CC solver. Thus, the update given by Equation 4.12 is performed without any restriction on the time step, but making it unstable at the same time.

$$\mathbf{U}_{cc,i}^{n+1} = \mathbf{U}_i^n - \frac{\Delta t^n}{\Delta V_i} \sum_{f \in \partial V_i} \mathbf{F}_f A_f \tag{4.12}$$

The next step is to compute the non-conservative solution $\mathbf{U}_{nc}^{n+1}$ using the RCCM solver. Equation 4.13 recalls the computation of the non-conservative solution $\mathbf{U}_{nc}^{n+1}$. This solution is stable due to the reconstruction on the `R-Cells`, but it is not conservative for the same reason.

$$\begin{cases} \mathbf{U}_{nc,i}^{n+1} = \mathbf{U}_{cc,i}^{n+1} & \text{for } i \in \texttt{C-Cells and N-Cells} \\ \mathbf{U}_{nc,i}^{n+1} = \texttt{Reconstruction}(\mathbf{U}_{nc}^{n+1}, \text{BC}) & \text{for } i \in \texttt{R-Cells} \end{cases} \tag{4.13}$$

Finally, the last step is to construct the new conservative solution $\mathbf{U}_c^{n+1}$ using the solution $\mathbf{U}_{nc}^{n+1}$ and $\mathbf{U}_{cc}^{n+1}$.

The following sections describe three different methods that allow to compute the new con-

servative solution $\mathbf{U}_c^{n+1}$ while ensuring stability.

### 4.3.2 Flux Redistribution Method

The Flux Redistribution Method (FRM) is based on the flux redistribution technique [49–52]. In the present work, the FRM is employed for the purpose of restoring conservation in the Immersed Boundary Method. A first optional step involves computing a new stable but still non-conservative solution $\mathbf{U}_{cont}^{n+1}$ using a linear combination of the solutions $\mathbf{U}_{cc}^{n+1}$ and $\mathbf{U}_{nc}^{n+1}$, as shown in Equation 4.14.

$$\mathbf{U}_{cont,i}^{n+1} = \alpha_i \mathbf{U}_{cc,i}^{n+1} + (1 - \alpha_i)\mathbf{U}_{nc,i}^{n+1} \tag{4.14}$$

The parameter $\alpha_i$ is defined as the ratio of the volume of cell $i$ to the volume of the uncut cell $i$, as shown in Equation 4.15.

$$\alpha_i = \frac{\Delta V_i}{\Delta V_{uncut,i}} \tag{4.15}$$

If the cell $i$ is an uncut cell, then $\alpha_i = 1$ and the new solution $\mathbf{U}_{cont,i}^{n+1}$ is equal to the conservative solution $\mathbf{U}_{cc,i}^{n+1}$ given by the CC solver. If the cell $i$ is a cut-cell, then $0 < \alpha_i < 1$ and by combining Equations 4.12, 4.14 and 4.15, it is evident that the division by the small cell volume $\Delta V_i$ is avoided, ensuring the stability of the solution $\mathbf{U}_{cont}^{n+1}$. For the remainder of this section, the solution $\mathbf{U}_{cont}^{n+1}$ will be used as the non-conservative solution $\mathbf{U}_{nc}^{n+1}$.

Next, the conservative solution $\mathbf{U}_c^{n+1}$ is computed using the solutions $\mathbf{U}_{nc}^{n+1}$ and $\mathbf{U}_{cc,i}^{n+1}$ with the flux redistribution technique. The redistribution is performed to satisfy the global conservation relation given by Equation 4.16.

$$\sum_{i \in \Omega} \Delta V_i \mathbf{U}_{c,i}^{n+1} = \sum_{i \in \Omega} \Delta V_i \mathbf{U}_{cc,i}^{n+1} \tag{4.16}$$

Since the solution $\mathbf{U}_{cc}^{n+1}$ is conservative, satisfying the constraint 4.16 ensures that the solution $\mathbf{U}_c^{n+1}$ will also be globally conservative by respecting the conservation relation 4.6. The flux redistribution is carried out as follows:

1. Initialize the conservative solution $\mathbf{U}_c^{n+1}$ with the solution $\mathbf{U}_{nc}^{n+1}$.

$$\mathbf{U}_c^{n+1} = \mathbf{U}_{nc}^{n+1} \tag{4.17}$$

2. After that, compute the mass error $\Delta M_i$ for each cell $i$ that allows to satisfy the

conservation relation 4.16.

$$\Delta M_i = \Delta V_i(\mathbf{U}_{cc,i}^{n+1} - \mathbf{U}_{c,i}^{n+1}) \tag{4.18}$$

Note that the mass error $\Delta M_i$ is equal to zero for the uncut cells. Therefore, only the mass error $\Delta M_i$ of the cut-cells will be redistributed.

3. Finally, redistribute all the mass errors $\Delta M_i$ among the cells such that the relation 4.16 is satisfied.

   (a) Loop over the cut-cells $i$ and generate a stencil $S(i)$ of neighbors of the cell $i$. In this work, the stencil $S(i)$ is taken to be the union of the cell $i$ and the set of cells that share a face with the cell $i$ or one of its direct neighbors.

   (b) Loop over the elements $j$ of the stencil $S(i)$ and distribute a part of the mass error $\Delta M_i$ to the neighbor $j$.

   $$\Delta M_{i,j} = \omega_{i,j}\Delta M_i \tag{4.19}$$

   The weights $\omega_{i,j}$ are chosen such that $\sum_{j\in S(i)} \omega_{i,j} = 1$ for each cell $i$ and are computed as follows.

   $$\omega_{i,j} = \frac{\Delta V_j}{\sum_{k\in S(i)} \Delta V_k} \tag{4.20}$$

   (c) Update the conservative solution $\mathbf{U}_{c,j}^{n+1}$ of the neighbor $j$.

   $$\mathbf{U}_{c,j}^{n+1} = \mathbf{U}_{c,j}^{n+1} + \frac{1}{\Delta V_j}\Delta M_{i,j} \tag{4.21}$$

The solution $\mathbf{U}_c^{n+1}$ obtained with the Flux Redistribution Method is both conservative and stable. However, the conservation is only global. The next section will present the Flux Interpolation Method, which aims to restore local conservation in the numerical solutions.

### 4.3.3 Flux Interpolation Method

To achieve locally conservative solutions, the local conservative discretization form given by Equation 4.4 must be employed. This section introduces a new method developed by the authors and called the Flux Interpolation Method (FIM) to accomplish this goal. The method operates as follows:

For each cell $i$, the new conservative solution $\mathbf{U}_{c,i}^{n+1}$ is expressed in the form of Equation 4.22.

$$\mathbf{U}_{c,i}^{n+1} = \mathbf{U}_i^n - \frac{\Delta t^n}{\Delta V_i} \sum_{f\in\partial V_i} \mathbf{F}_{m,f} A_f \tag{4.22}$$

Thus, it suffices to compute the fluxes $\mathbf{F}_{m,f}$ using the solution $\mathbf{U}_{nc}^{n+1}$ from the RCCM solver to obtain the conservative solution $\mathbf{U}_{c}^{n+1}$.

Since the conservation form 4.22 is already satisfied for the non-reconstructed cells (`C-Cells` and `N-Cells`), the fluxes $\mathbf{F}_{m,f}$ for all faces $f$ shared by two non-reconstructed cells are set equal to the fluxes $\mathbf{F}_f$ obtained with the FVM discretization using the solution $\mathbf{U}^n$. Consequently, the solutions at the `C-Cells` remain unchanged, as shown in Equation 4.23.

$$\mathbf{U}_{c,i}^{n+1} = \mathbf{U}_{nc,i}^{n+1} \qquad \text{for } i \in \texttt{C-Cells} \tag{4.23}$$

For the faces $f$ that are shared by two `R-Cells` or by an `R-Cell` and an `N-Cell`, referred to as interpolated faces (`I-Faces`), the fluxes $\mathbf{F}_{m,f}$ are interpolated using the solution $\mathbf{U}_{nc}^{n+1}$. The interpolation is performed through the following steps:

1. The solution $\mathbf{U}_{c}^{n+1}$ at the `R-Cells` is set to be equal to the solution $\mathbf{U}_{nc}^{n+1}$.

$$\mathbf{U}_{c,i}^{n+1} = \mathbf{U}_{nc,i}^{n+1} \qquad \text{for } i \in \texttt{R-Cells} \tag{4.24}$$

   Only the solution $\mathbf{U}_{c}^{n+1}$ at the `N-Cells` remains then to be computed.

2. A combination of Equations 4.24 and 4.22 is then made for the goal of obtaining a set of equations that constrain the fluxes $\mathbf{F}_{m,f}$ at the `I-Faces`:

$$\sum_{\substack{f \in \partial V_i \\ f \in \texttt{I-Faces}}} \mathbf{F}_{m,f} A_f = \frac{\Delta V_i}{\Delta t^n} \left( \mathbf{U}_i^n - \mathbf{U}_{nc,i}^{n+1} \right) - \sum_{\substack{f \in \partial V_i \\ f \notin \texttt{I-Faces}}} \mathbf{F}_f A_f \qquad \text{for } i \in \texttt{R-Cells} \tag{4.25}$$

   Equation 4.25 is a linear system of equations of the form $Kx = b$ where the unknown $x$ represents the fluxes $\mathbf{F}_{m,f} A_f$ at the `I-Faces`. The right-hand side $b$ depends on the solution $\mathbf{U}_{nc}^{n+1}$, and the matrix $K$ depends only on the discretization grid. The dimensions of the matrix $K$ are $n \times m$ where $n$ is the number of reconstructed cells and $m$ is the number of `I-Faces`.

3. If the number of `R-Cells` is less than the number of `I-Faces`, which is usually the case, then the system of Equations 4.25 can be solved using the least norm method:

$$\begin{aligned} \min_{\mathbf{F}_{m,f}} \quad & \sum_{f \in \texttt{I-Faces}} \left( \mathbf{F}_{m,f} A_f - \mathbf{F}_f A_f \right)^2 \\ \text{s.t.} \quad & \mathbf{U}_{c,i}^{n+1} = \mathbf{U}_{nc,i}^{n+1} \qquad \text{for } i \in \texttt{R-Cells} \end{aligned} \tag{4.26}$$

In matrix form, the optimization problem 4.26 can be written as:

$$\begin{aligned} \min_{x} \quad & \|x - x_0\|^2 \\ \text{s.t.} \quad & Kx = b \end{aligned} \tag{4.27}$$

where $x_0$ is the vector of the standard FVM fluxes $\mathbf{F}_f A_f$ at the `I-Faces`. The least norm solution of the problem 4.27 is given by Equation 4.28.

$$x = x_0 + K^T(KK^T)^{-1}(b - Kx_0) \tag{4.28}$$

4. If the number of `R-Cells` exceeds the number of `I-Faces`, the system of Equations 4.25 becomes over-determined, and not all conservation constraints can be satisfied. In this case, the system of Equations 4.25 is solved using the least squares method:

$$\min_{\mathbf{F}_{m,f}} \quad \sum_{i \in \texttt{R-Cells}} \left(\mathbf{U}_{c,i}^{n+1} - \mathbf{U}_{nc,i}^{n+1}\right)^2 \tag{4.29}$$

In matrix form, the minimization problem 4.29 can be written as:

$$\min_{x} \quad \|Kx - b\|^2 \tag{4.30}$$

The solution to the optimization problem 4.30 is given by Equation 4.31.

$$x = (K^T K)^{-1} K^T b \tag{4.31}$$

5. Once all the fluxes $\mathbf{F}_{m,f}$ are computed, the conservative solution $\mathbf{U}_c^{n+1}$ at the `N-Cells` is obtained using Equation 4.22.

The Flux Interpolation Method is stable since the reconstructed solutions at the `R-Cells` are kept unchanged. However, the method is not always conservative since the system of Equations 4.25 can be over-determined in some cases. Nevertheless, in most situations, the method is conservative since the system 4.25 is usually under-determined. The next section will introduce another new method developed by the authors and called Flux Correction Method, which is a variant of the Flux Interpolation Method that is always locally conservative.

### 4.3.4 Flux Correction Method

The Flux Correction Method (FCM) closely resembles the Flux Interpolation Method but with relaxed constraints from Equation 4.25. Here, an alternative formulation of the method

will be presented. The approach begins with the non-conservative solution $\mathbf{U}_{nc}^{n+1}$, obtained, for example, using the RCCM solver. The solutions at the `C-Cells` are left unchanged since they already satisfy the conservative discretization form 4.4. Conversely, the solutions at the `R-Cells` and `N-Cells` are perturbed to meet the conservative discretization form 4.4. The updated solution $\mathbf{U}_{c}^{n+1}$ for the `R-Cells` and `N-Cells` is expressed as in Equation 4.32.

$$\begin{gathered}\mathbf{U}_{c,i}^{n+1} = \mathbf{U}_{cc,i}^{n+1} - \frac{\Delta t^n}{\Delta V_i} \sum_{\substack{f \in \partial V_i \\ f \in \mathtt{I-Faces}}} \left(\mathbf{F}_{m,f} - \mathbf{F}_f\right) A_f \\ \text{for } i \in \mathtt{R-Cells} \cup \mathtt{N-Cells}\end{gathered} \tag{4.32}$$

The objective is then to compute the perturbation $\Delta\mathbf{F}_f = \left(\mathbf{F}_{m,f} - \mathbf{F}_f\right) A_f$ for all the faces $f \in \mathtt{I-Faces}$. Remark that the update in Equation 4.32 corresponds to a fully conservative update of the form 4.4, since only the values of the fluxes at the interpolated faces $f$ are modified compared to the Cut-Cells solver. If the perturbed fluxes $\Delta\mathbf{F}_f$ vanish, then the solution $\mathbf{U}_{c}^{n+1}$ is exactly identical to the solution $\mathbf{U}_{cc}^{n+1}$ obtained with the CC solver. However, since the solution $\mathbf{U}_{cc}^{n+1}$ is unstable, the perturbed fluxes $\Delta\mathbf{F}_f$ will be non-zero in general so that they can positively affect the stability of the solution $\mathbf{U}_{c}^{n+1}$.

The perturbed fluxes $\Delta\mathbf{F}_f$ are computed using the reconstructed solution $\mathbf{U}_{nc}^{n+1}$ from Equation 4.13, following these steps:

1. Using Equation 4.32, the differences between the solutions $\mathbf{U}_{c}^{n+1}$ and $\mathbf{U}_{nc}^{n+1}$ at the `R-Cells` and `N-Cells` is minimized:

$$\min_{\Delta\mathbf{F}_f} \quad \sum_{i \in \mathtt{R-Cells} \cup \mathtt{N-Cells}} \left(\mathbf{U}_{c,i}^{n+1} - \mathbf{U}_{nc,i}^{n+1}\right)^2 \tag{4.33}$$

2. As in the Flux Interpolation Method, the minimization problem 4.33 can be written in matrix form as:

$$\min_{x} \quad \|Kx - b\|^2 \tag{4.34}$$

   where $x$ is the vector of the perturbed fluxes $\Delta\mathbf{F}_f$ at the `I-Faces`. The right-hand side $b$ depends on the solution $\mathbf{U}_{nc}^{n+1}$, and the matrix $K$ depends only on the discretization grid. The dimensions of the matrix $K$ is $n \times m$ where $n$ is the total number of `R-Cells` and `N-Cells` and $m$ is the number of `I-Faces`. The problem 4.33 is solved using the least squares method to obtain the perturbed fluxes $\Delta\mathbf{F}_f$.

3. Once the perturbed fluxes $\Delta\mathbf{F}_f$ are computed, the conservative solution $\mathbf{U}_{c}^{n+1}$ at the `R-Cells` and `N-Cells` is obtained using Equation 4.32.

The Flux Correction Method is conservative in general since the conservative update 4.32 is applied to every cell. The method is also stable because the reconstructed solutions $\mathbf{U}_{nc}^{n+1}$ are used to constrain the perturbed fluxes $\Delta\mathbf{F}_f$. However, since the solutions at the small cells initially obtained through reconstruction are now perturbed, the stability of the Flux Correction Method may not be as robust as that of the Flux Interpolation Method.

In this section, three conservative methods have been presented that allow to restore the conservation properties on the numerical solutions. The Flux Redistribution Method is the simplest method and ensures global conservation of the solution by redistributing the mass errors among the cells. The Flux Interpolation Method and the Flux Correction Method are more sophisticated methods that ensure local conservation of the solution within each cell by solving an optimization problem. The next section will present numerical results of the different methods using various test cases.

### Summary of the Different Methods

The properties of the methods introduced so far are summarized in Table 4.2.

Table 4.2 Summary of the Different Methods.

| **Method** | **Conservation** | **Small Cells $\Delta t$ Constraint** | **Comments** |
|---|---|---|---|
| CC | YES | YES | Stable only with very small time steps. |
| IBM | NO | NO | Fastest method, but has the largest errors due to loss of conservation and reconstruction. |
| RCCM | NO | NO | Errors from loss of conservation and reconstruction are smaller than those in the IBM method [1]. |
| FRM | YES | NO | Conservation is guaranteed only globally. |
| FIM | YES/NO | NO | In rare cases, the method may not be fully conservative. |
| FCM | YES | NO | Less stable than the Flux Interpolation Method, but always locally conservative. |

## 4.4 Numerical Results

This section presents four test cases to evaluate the performance of the different methods. The first test case is the shock tube problem of [117], which is used to examine the conservation properties of the various methods. The second test case involves a transonic flow through a canal with a bump, aimed at qualitatively assessing the impact of conservation properties on capturing critical flow features. The third test case examines the confluence of two supersonic

flows, providing a measure of the accuracy of the different methods for handling discontinuous solutions. The final test case is the supersonic vortex problem, which is used to study the convergence behavior of the different methods with a smooth solution.

### 4.4.1 Shock Tube

The test case extends the one presented in reference [1]. Here, the shock tube is rotated by an angle of 45° to allow the presence of `R-Cells` at the boundary of the domain. The geometry is illustrated in Figure 4.6, and the parameters of the test case are given in Table 4.3. The final time of the simulation is $t_{final} = 0.15$.

Table 4.3 Initial conditions and domain parameters of the shock tube test case.

| **Left state** $[\rho_L,\ u_L,\ p_L]$ | **Right state** $[\rho_R,\ u_R,\ p_R]$ | **Membrane position** $x_0$ | **Domain** $[L, H]$ | **Grid cell size** $[\Delta x, \Delta y]$ |
|---|---|---|---|---|
| $[1.0,\ 0.0,\ 1.0]$ | $[0.125,\ 0.0,\ 0.1]$ | 0.3 | $[1.0, 1.0]$ | $[0.029, 0.029]$ |

Solid Wall boundary conditions are imposed on all the boundaries, ensuring that the total mass within the system remains constant. Figure 4.7 compares the numerical solutions for density and velocity profiles obtained with different methods against the analytical solution at the final time.

The differences between the methods are very negligible in terms of accuracy, as shown in Figure 4.7. However, significant differences arise in terms of conservation, as illustrated in Figure 4.8. Figures 4.8a and 4.8b depict the evolution of the errors on the total mass and energy of the system for the different methods as a function of the average mesh size. The mass and energy conservation errors, along with the average mesh size, are defined by Equations 4.35, 4.36, and 4.37, respectively.

$$\text{Mass Conservation Error} = \frac{\left|\sum_{i\in\Omega} \Delta V_i \rho_i - \sum_{i\in\Omega} \Delta V_i \rho_{i,0}\right|}{\sum_{i\in\Omega} \Delta V_i \rho_{i,0}} \tag{4.35}$$

$$\text{Energy Conservation Error} = \frac{\left|\sum_{i\in\Omega} \Delta V_i \rho_i E_i - \sum_{i\in\Omega} \Delta V_i \rho_{i,0} E_{i,0}\right|}{\sum_{i\in\Omega} \Delta V_i \rho_{i,0} E_{i,0}} \tag{4.36}$$

$$\text{Average Mesh Size } h = \sqrt{\frac{\sum_{i\in\Omega} \Delta V_i}{\text{Total Number of Cells}}} \tag{4.37}$$

where $\rho_{i,0}$ and $\rho_i$ are respectively the initial and final density of the cell $i$, and the same notation is used for the total energy $E$.

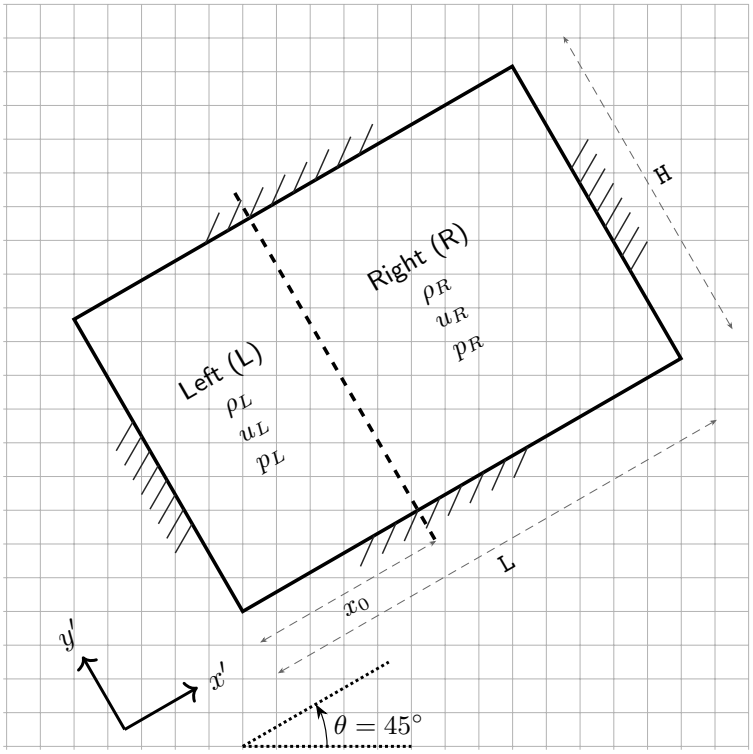
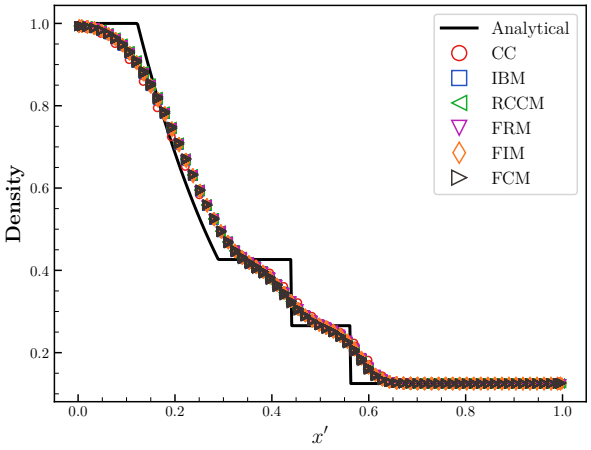
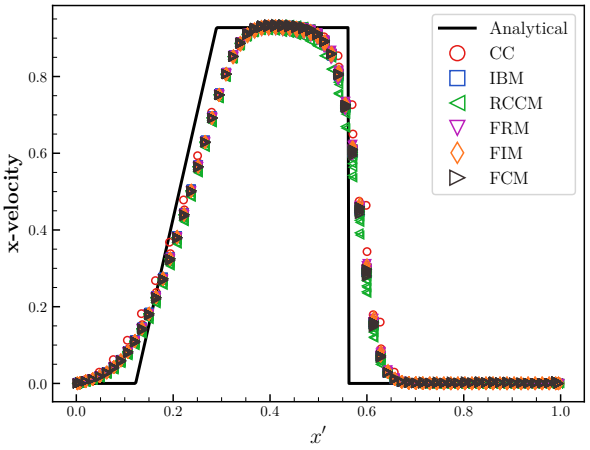



Figure 4.6 Geometry of the rotated shock tube test case.

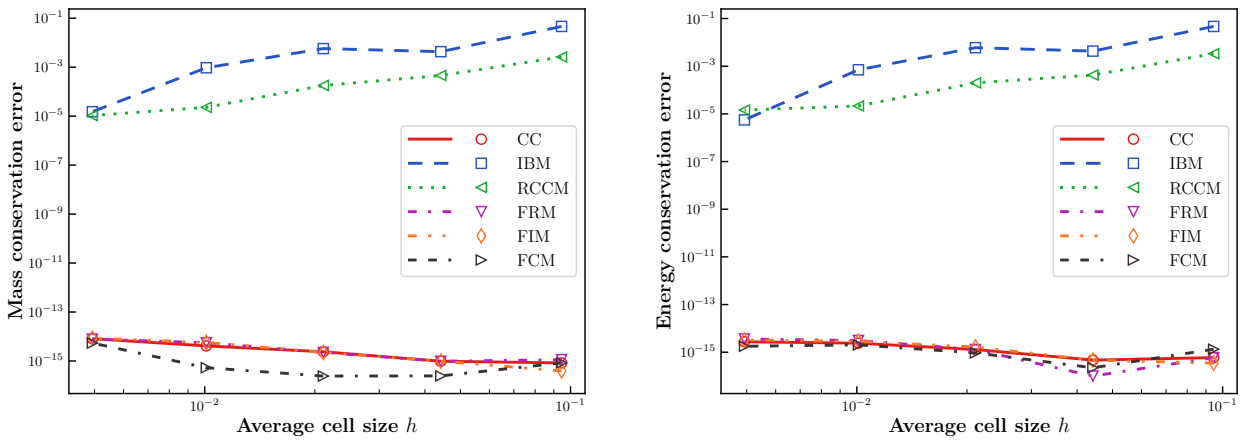



(a) Density profiles at the final time.




(b) Velocity profiles at the final time.

Figure 4.7 Comparison of the numerical solutions for the shock tube test case.




(a) Mass conservation error.

(b) Energy conservation error.

Figure 4.8 Comparison of the conservation errors for the shock tube test case.

The mass and energy conservation errors for the CC and conservative methods are consistently at machine precision, as expected. The non-conservative methods (IBM and RCCM) exhibit non-zero conservation errors that diminish as the mesh is refined. Since the RCCM method takes into account all cut-cells, its conservation errors are smaller than those of the IBM method, particularly for coarse meshes. As the mesh is refined, the size of the cut-cells ignored by the IBM method becomes negligible, resulting in similar mass and energy conservation errors for both methods. However, the conservation errors of the IBM and RCCM methods do not reach machine precision, even for very fine meshes, due to the reconstruction at the boundaries introducing some conservation errors. For the conservative methods, despite relying on a reconstruction at the boundary, the conservative correction step ensures mass and energy conservation errors at machine precision for all mesh sizes.

The next test case will demonstrate the impact of conservation properties on capturing critical flow features.

### 4.4.2 Transonic Flow Inside a Canal with a Bump

The second test case involves a transonic flow through a canal featuring a bump with a thickness of 10%. This test case extends also the one presented in reference [1]. The geometry along with the boundary conditions are illustrated in Figure 4.9.

Figure 4.10 shows the Mach number contours obtained with the CC method for a fine mesh (last mesh of Table 4.4). The contours of the other methods are indistinguishable from those of the CC method and are therefore not shown. The shock on the bump and other important

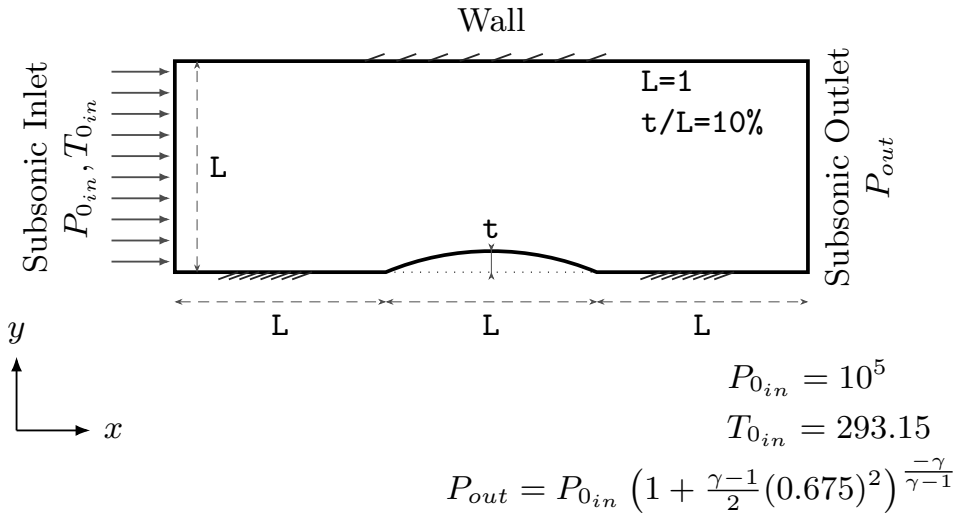



Figure 4.9 Geometry and boundary conditions for the transonic flow in a canal with a bump [1, 2].

flow features, such as the supersonic pocket, are well captured by all solvers for this mesh.

To evaluate the impact of the conservation properties on the numerical solutions, the mass flow rate and the pressure coefficient along the lower surface of the canal are presented for successive mesh refinements. The properties of the four meshes used are detailed in Table 4.4. All the meshes have between 0.2% to 1.0% cut-cells, and the canal is positioned on the grid such that the ratio of the area between an uncut cell and the smallest cut-cell ranges from 23 to 237.

Table 4.4 Meshes size and properties for the transonic flow over a bump test case.

| **Mesh** | **Grid Size** | **Number of Uncut Cells** | **Number of Cut-Cells** | **Ratio of the Cut-Cells Area [%]** | **Ratio of the outer cut-cells area [%]** | **Factor Max/Min of Cell Area** |
|---|---|---|---|---|---|---|
| Extra Coarse | $150 \times 50$ | 5680 | 364 | 0.950 | 2.853 | 22.807 |
| Coarse | $300 \times 100$ | 23722 | 738 | 0.922 | 0.477 | 110.878 |
| Medium | $450 \times 150$ | 54933 | 1128 | 0.177 | 0.678 | 80.298 |
| Fine | $600 \times 200$ | 93883 | 1472 | 0.305 | 1.464 | 236.641 |

At steady state, the mass flow rate through the canal is constant and is given by Equation 4.38 for a surface $x = c$ where $c$ is a constant.

$$\dot{m} = \int_0^L \rho u dy = \sum_{i|\ x_i = \mathrm{c}} \rho_i u_i \Delta y_i \tag{4.38}$$

In Equation 4.38, $\rho_i$ and $u_i$ are respectively the density and velocity of the cell $i$, thus

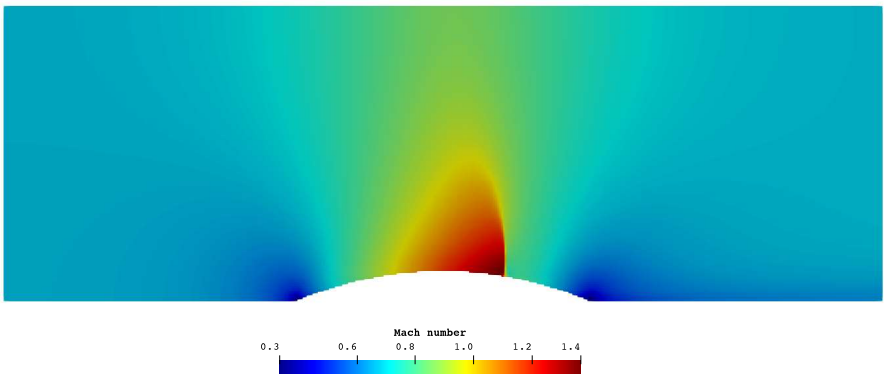



Figure 4.10 Mach number contours obtained with the CC method for the transonic flow over bump.

numerically, the mass flow rate is not guaranteed to be constant for different surfaces $x = c$, even if the solver is conservative, since the conserved variables $\mathbf{U}$ are used to compute the mass flow rate not the fluxes $\mathbf{F}$. By evaluating the mass flow rate for two surfaces $x = c_1$ and $x = c_2$ where $c_1 < c_2$, an error on the mass flow rate can be computed as shown in Equation 4.39.

$$\text{Mass Flow Rate Error} = \frac{|\dot{m}_{c_1} - \dot{m}_{c_2}|}{\dot{m}_{c_2}} \tag{4.39}$$

It is expected that the error on the mass flow rate will diminish as the mesh is refined and that the conservative methods will have a smaller error compared to the non-conservative methods. This is indeed observed in Figure 4.11, where the error in the mass flow rate between two surfaces near the inlet and the outlet of the canal is plotted as a function of the number of cells in the $y$ direction. Except for the second mesh, the mass flow rate errors of the conservative methods have the same order of magnitude, and they are smaller than the ones obtained with the non-conservative methods (IBM and RCCM). The error on the mass flow rate of the IBM method is larger than the RCCM method, confirming that the RCCM method is more conservative than the IBM method.

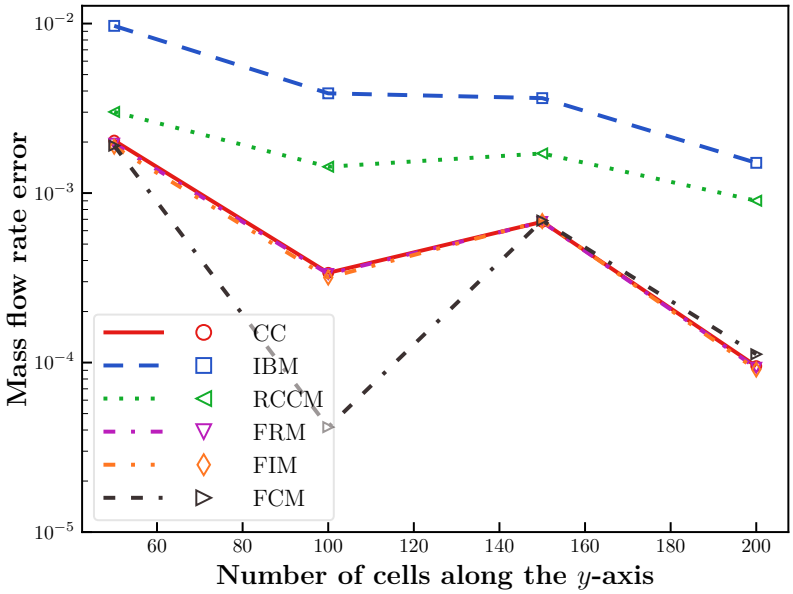
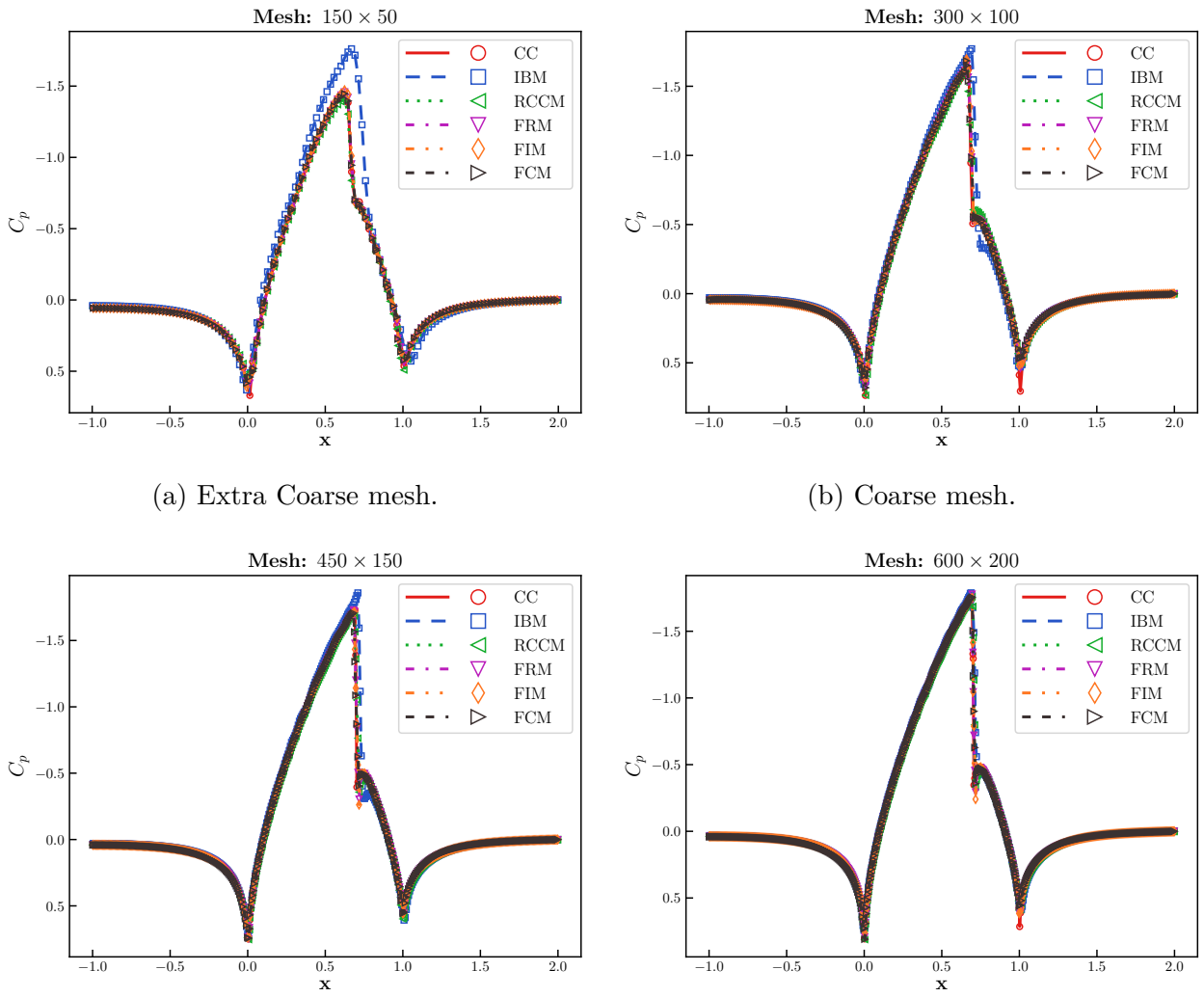



Figure 4.11 Mass flow rate error for the transonic flow over a bump test case.

Another way to assess the impact of the conservation properties on the numerical solutions is to examine the pressure coefficient along the lower surface of the canal. Figures 4.12a to 4.12d show the pressure coefficient along the lower surface of the canal for the four meshes. First, it can be observed that all the methods seem to converge to the same solution as the mesh is refined. Except for the IBM method, the pressure coefficient profiles of the CC method are very similar to those of the other methods, especially the conservative ones (FRM, FIM and FCM). The shock position on the lower surface of the canal obtained with the IBM method is a slightly shifted compared to the other methods, particularly for the coarse meshes. As the mesh is refined, the shock position of the IBM method converges to the shock position obtained with the other methods. The differences in the shock position can be attributed to the non-conservative nature of the IBM method [108, 118]. The shock position of the RCCM method is only slightly shifted compared to the conservative methods. This is understandable since, although the RCCM method is non-conservative, it accounts for all cut-cells, resulting in smaller conservation errors than the IBM method. Consequently, the shock position of the RCCM method is closer to that of the conservative methods. The solutions of the three conservative methods are almost identical to those of the CC method for all meshes.

The next section will evaluate the accuracy of the different methods using a 2D supersonic flow containing discontinuities.




(a) Extra Coarse mesh.

(b) Coarse mesh.

(c) Medium mesh.

(d) Fine mesh.

Figure 4.12 Pressure coefficient $C_p$ along the lower surface of the canal for the transonic flow over a bump test case.

### 4.4.3 Confluence of Two Supersonic Flows

Figure 4.13 illustrates the configuration of the confluence test case. Two supersonic flows coming from opposite angles are colliding at the center point of the left boundary. The parameters for this problem are given in Table 4.5. The analytical solution is characterized by three discontinuities: two shocks and a contact discontinuity, defined by their respective angles $\beta_1$, $\beta_2$, and $\beta_c$. These three discontinuities delimit four regions of constant properties. Using the theory of compressible flows [119], the positions of these discontinuities and the state of the flow in each region can be computed. The analytical solution corresponding to

the parameters in Table 4.5 is provided in Table 4.6.

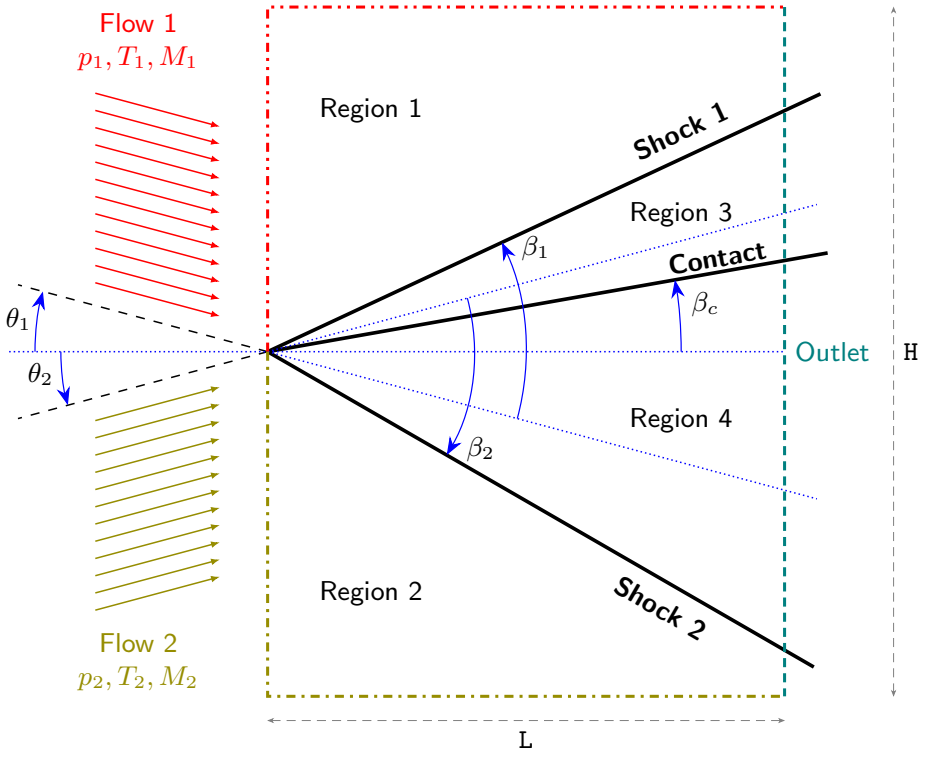



Figure 4.13 Geometry and boundary conditions of the confluence test case.

Table 4.5 Domain and initial conditions of the confluence test case.

| **Flow 1** | **Flow 2** | **Domain** |
|---|---|---|
| $[M_1,\ p_1,\ T_1,\ \theta_1]$ | $[M_2,\ p_2,\ T_2,\ \theta_2]$ | $[L,\ H]$ |
| $[1.8,\ 10^5,\ 300,\ 10°]$ | $[2.3,\ 1.5 \times 10^5,\ 400,\ 10°]$ | $[1,\ 2]$ |

To visualize the numerical solutions, a mesh $M_0$ aligned with the geometry is used, and its properties are listed in Table 4.7. The vertical alignment of the mesh with the geometry ensures a better cut near the outlet plane. Figure 4.14a shows the convergence of the density residuals of all solvers for the mesh $M_0$. It can be observed that the residuals of all the solvers have successfully converged to the machine precision. Figure 4.14b displays the Mach number cuts near the outlet plane for the different methods and the analytical solution. Similar to the shock tube test case, the numerical solutions obtained with the different methods are very similar. The shocks and contact discontinuity are captured by all solvers,

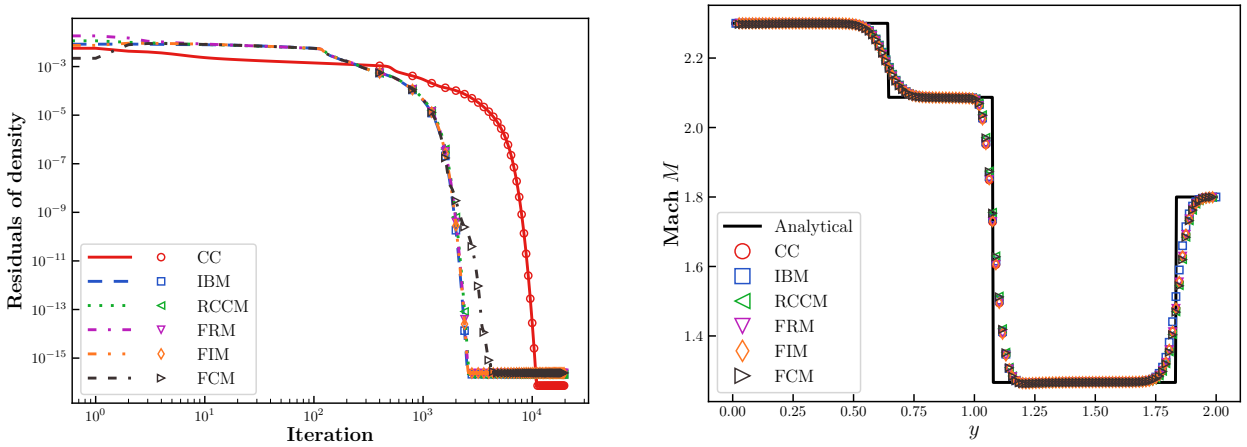
Table 4.6 Analytical solution for the confluence test case.

| **Discontinuities** $[\beta_1,\ \beta_2,\ \beta_c]$ | **Region 3** $[M_3,\ p_3,\ T_3]$ | **Region 4** $[M_4,\ p_4,\ T_4]$ |
|---|---|---|
| $[50.523°,\ 30.152°,\ 4.528°]$ | $[1.266,\ 2.086 \times 10^5,\ 374.365]$ | $[2.087,\ 2.086 \times 10^5,\ 439.958]$ |

albeit with some diffusion, which is expected due to the use of a first-order scheme. Overall, the reconstruction procedure and the conservative correction step do not introduce significant errors in the numerical solutions.

Table 4.7 Properties of the mesh used for visualizing the numerical solutions of the confluence test case.

| **Mesh** | **Grid size** | **Number of uncut cells** | **Number of cut-cells** | **Ratio of the cut-cells area [%]** | **Ratio of the outer cut-cells area [%]** | **Factor max/min of cell area** |
|---|---|---|---|---|---|---|
| $M_0$ | $80 \times 160$ | 10368 | 436 | 1.3465 | 0.956 | 11.243 |




(a) Convergence of the density residuals.

(b) Cut of the mach number near the outlet.

Figure 4.14 Comparison of the different methods for the confluence test case.

To assess the accuracy of the different methods, five meshes are used. These meshes are generated using the geometry shown in Figure 4.13, but rotated by 45° to introduce arbitrary reconstructed cells at the domain boundaries. The properties of these meshes are listed in Table 4.8.

Figure 4.15 shows the convergence of the density with the mesh refinement for the different

Table 4.8 Mesh sizes and properties for the convergence study of the confluence test case.

| **Mesh** | **Grid size** | **Number of uncut cells** | **Number of cut-cells** | **Ratio of the cut-cells area [%]** | **Ratio of the outer cut-cells area [%]** | **Factor max/min of cell area** |
|---|---|---|---|---|---|---|
| $M_1$ | $25 \times 25$ | 175 | 88 | 3.392 | 16.726 | 23.469 |
| $M_2$ | $50 \times 50$ | 812 | 178 | 1.102 | 8.387 | 10.302 |
| $M_3$ | $100 \times 100$ | 3422 | 359 | 1.147 | 4.538 | 11.947 |
| $M_4$ | $200 \times 200$ | 12656 | 683 | 0.512 | 2.359 | 130.641 |
| $M_5$ | $400 \times 400$ | 56525 | 1436 | 0.395 | 1.197 | 70.201 |

methods. The $L_1$ and $L_2$ error norms used to compute the convergence rate of a field $\phi$ are defined in Equations 4.40 and 4.41.

$$\epsilon_{L_1} = \sum_{i \in \Omega} \frac{\Delta V_i}{V} \left| \phi_i - \phi_{i,\text{exact}} \right| \tag{4.40}$$

$$\epsilon_{L_2} = \sqrt{\sum_{i \in \Omega} \frac{\Delta V_i}{V} \left( \phi_i - \phi_{i,\text{exact}} \right)^2} \tag{4.41}$$

Where $\phi_i$ and $\phi_{i,\text{exact}}$ are respectively the numerical and analytical values of the field $\phi$ at the cell $i$, and $V = \sum_{i \in \Omega} \Delta V_i$ denotes the total volume of the domain.

The convergence rates are computed using Equation 4.42.

$$\text{convergence rate} = \frac{\log\left(\frac{\epsilon_{h_2}}{\epsilon_{h_1}}\right)}{\log\left(\frac{h_2}{h_1}\right)} \tag{4.42}$$

Where $\epsilon_{h_1}$ and $\epsilon_{h_2}$ are the error norms obtained with the meshes with sizes $h_1$ and $h_2$, respectively. The mesh size $h$ is defined by Equation 4.37.

All methods yield similar results, with convergence rates around 0.3 and 0.6 for the $L_2$ and $L_1$ norms, respectively, as shown in Figure 4.15. Similar convergence rates are observed for other fields such as pressure and Mach number, as detailed in Table 4.9. The expected convergence rate for a first-order scheme in smooth flows is 1 for all norms. However, the presence of discontinuities in the solution explains why the convergence rate is not close to 1 [120].

The next section will employ a smooth solution to more accurately compute the convergence of the different methods.

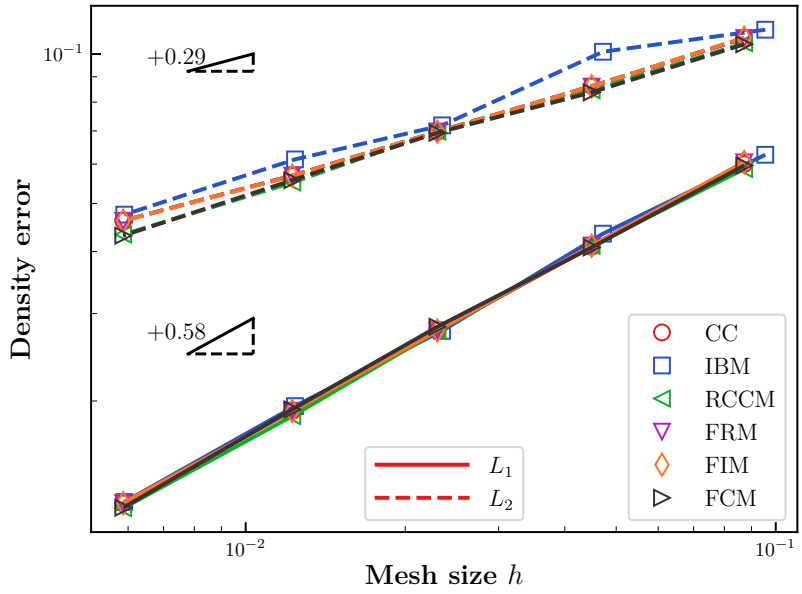



Figure 4.15 Convergence of the density with the mesh refinement for the confluence test case.

Table 4.9 Convergence rates of the different methods for the confluence test case.

| | $\rho$ | | $M$ | | $p$ | | $T$ | |
|---|---|---|---|---|---|---|---|---|
| | $L_1$ | $L_2$ | $L_1$ | $L_2$ | $L_1$ | $L_2$ | $L_1$ | $L_2$ |
| CC | 0.58 | 0.29 | 0.51 | 0.25 | 0.66 | 0.30 | 0.54 | 0.26 |
| IBM | 0.60 | 0.35 | 0.51 | 0.25 | 0.70 | 0.41 | 0.53 | 0.26 |
| RCCM | 0.58 | 0.33 | 0.52 | 0.25 | 0.71 | 0.45 | 0.52 | 0.23 |
| FRM | 0.58 | 0.28 | 0.51 | 0.25 | 0.66 | 0.30 | 0.54 | 0.26 |
| FIM | 0.58 | 0.28 | 0.51 | 0.25 | 0.66 | 0.30 | 0.54 | 0.26 |
| FCM | 0.62 | 0.35 | 0.54 | 0.25 | 0.76 | 0.47 | 0.56 | 0.24 |

### 4.4.4 Supersonic Vortex Flow between two Concentric Quarter Circles

The final test case examines a supersonic vortex flow between two concentric quarter circles, a test extensively used in the literature to evaluate the accuracy of numerical methods for inviscid compressible flows [44, 121, 122]. The geometry is shown in Figure 4.16, and the analytical solution is provided in Equation 4.43.

$$\begin{aligned} \rho &= \rho_{in}\left[1+\frac{\gamma-1}{2}M_{in}^2\left(1-\frac{r_{in}^2}{r^2}\right)\right]^{\frac{1}{\gamma-1}} \\ p &= \frac{\rho^\gamma}{\gamma} \qquad u = |\mathbf{v}|\cos\theta \qquad v = |\mathbf{v}|\sin\theta \\ r|\mathbf{v}| &= r_{in}|\mathbf{v}_{in}| \quad \text{where } |\mathbf{v}_{in}| = M_{in}\left(\rho_{in}^{\frac{\gamma-1}{2}}\right) \end{aligned} \tag{4.43}$$

where $\rho_{in}$ and $M_{in}$ are respectively the density and mach number at the inner radius $r_{in}$, and are chosen to be 1.0 and 2.25, respectively.

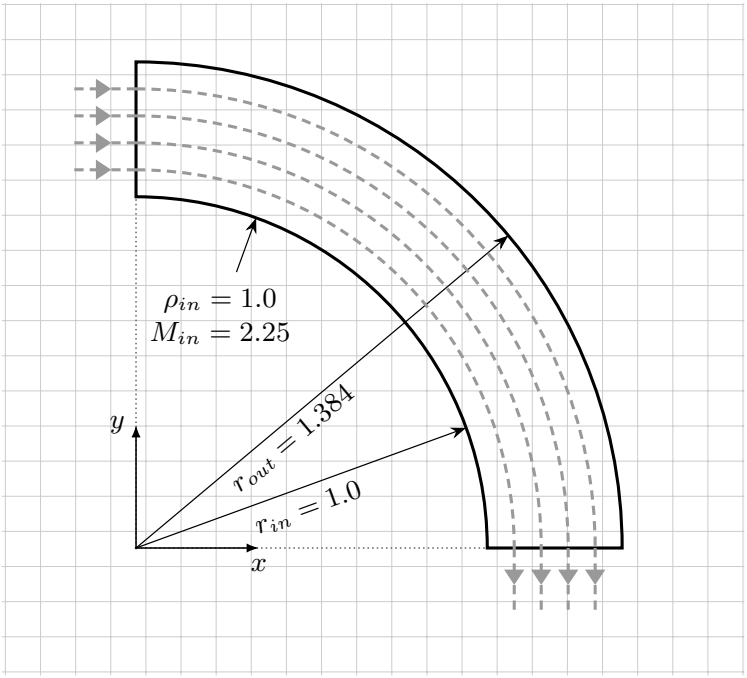


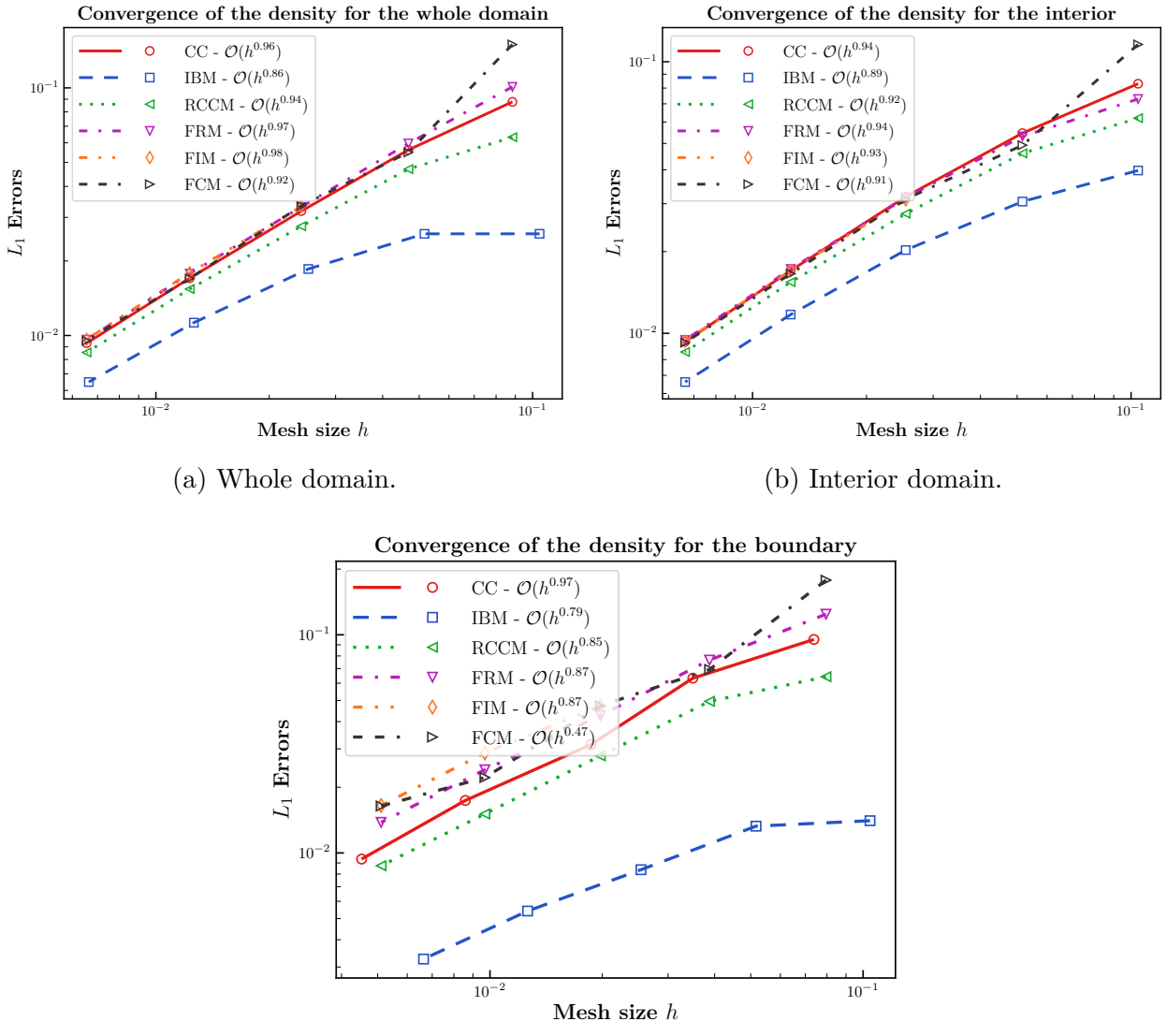

Figure 4.16 Geometry of the vortex test case.

To obtain the numerical solutions, the analytical solution is imposed as Dirichlet boundary condition on all the boundaries and used as initial condition as well. Five uniform grids, ranging from $16 \times 16$ to $256 \times 256$ cells, are employed. After the convergence of the residuals of all the methods to the machine precision, the error norms ($L_1$, $L_2$ and $L_\infty$) and convergence rates of the density and the pressure are computed using Equations 4.40, 4.41, 4.44 and 4.42, respectively.

$$\epsilon_{L_\infty} = \max_{i\in\Omega} |\phi_i - \phi_{i,\text{exact}}| \tag{4.44}$$

Also, to measure the impact of the reconstruction procedure and the conservative correction step on the convergence of the numerical solution, the convergence rates are computed on three set of cells: the entire domain (W), only the cells strictly inside the domain (I), and only the cells on the boundary of the domain (B).

Figures 4.17a to 4.17c illustrate the convergence of the density using the $L_1$ norm for the whole domain, the interior, and the boundary, respectively. Tables 4.10 and 4.11 present the convergence rates of the density and the pressure for the different methods across the different regions.




(a) Whole domain.

(b) Interior domain.

(c) Boundary domain.

Figure 4.17 Convergence of the density for the vortex test case using the $L_1$ norm.

Table 4.10 Convergence rates of the density for the different methods and regions.

| | $\mathbf{L_1}$ | | | $\mathbf{L_2}$ | | | $\mathbf{L_\infty}$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | W | I | B | W | I | B | W | I | B |
| CC | 0.96 | 0.94 | 0.97 | 0.96 | 0.94 | 0.95 | 0.91 | 0.89 | 0.88 |
| IBM | 0.86 | 0.89 | 0.79 | 0.86 | 0.88 | 0.77 | 0.75 | 0.75 | 0.69 |
| RCCM | 0.94 | 0.92 | 0.85 | 0.94 | 0.92 | 0.84 | 0.86 | 0.84 | 0.85 |
| FRM | 0.97 | 0.94 | 0.87 | 0.98 | 0.94 | 0.83 | 0.73 | 0.85 | 0.72 |
| FIM | 0.98 | 0.93 | 0.87 | 1.00 | 0.93 | 0.85 | 0.67 | 0.89 | 0.66 |
| FCM | 0.92 | 0.91 | 0.47 | 0.83 | 0.91 | 0.02 | −1.55 | 0.87 | −1.53 |

Table 4.11 Convergence rates of the pressure for the different methods and regions.

| | $\mathbf{L_1}$ | | | $\mathbf{L_2}$ | | | $\mathbf{L_\infty}$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | W | I | B | W | I | B | W | I | B |
| CC | 0.95 | 0.92 | 0.91 | 0.95 | 0.93 | 0.92 | 0.94 | 0.92 | 0.93 |
| IBM | 0.90 | 0.93 | 0.87 | 0.89 | 0.91 | 0.83 | 0.84 | 0.84 | 0.81 |
| RCCM | 0.94 | 0.91 | 0.90 | 0.94 | 0.92 | 0.90 | 0.96 | 0.94 | 0.93 |
| FRM | 0.98 | 0.93 | 0.88 | 1.01 | 0.94 | 0.89 | 0.91 | 0.91 | 0.90 |
| FIM | 0.98 | 0.92 | 0.83 | 0.99 | 0.93 | 0.80 | 0.64 | 0.92 | 0.63 |
| FCM | 0.95 | 0.88 | 0.83 | 0.99 | 0.90 | 0.88 | 0.61 | 0.90 | 0.60 |

Aside from a few exceptions, all methods exhibit convergence rates close to the expected rate of 1 for all three regions, consistent with the first-order scheme employed in this study. For the $L_1$ and $L_2$ norms, the convergence rates of the CC, RCCM, and conservative methods are nearly identical across the entire domain and its interior. This similarity arises because the cut-cells are very small, resulting in minimal $L_1$ and $L_2$ norm errors associated with them. However, as shown in Tables 4.10 and 4.11, the convergence rates degrade slightly for all three regions when considering the $L_\infty$ norm. The boundary of the domain is particularly affected by this degradation, especially for the conservative method. This degradation is partly due to the reconstruction procedure (used in RCCM and conservative methods) and partly due to the conservative correction step (specific to the conservative method), which can introduce discrepancies in the solution compared to the CC method, particularly near the domain boundaries. The FCM method appears to be the most affected near the domain boundary. For the IBM method, the convergence rates are somewhat lower compared to the

other methods, although its errors at the boundary are relatively smaller. The small errors at the boundary arise from the fact that the boundary cells employed in the IBM method are of high quality and comparable in size to the interior cells.

## 4.5 Conclusion

This paper presents three distinct methods designed to restore the conservation properties of the Immersed Boundary Method using a Cut-Cells approach. All methods are based on a semi-implicit reconstruction procedure that mitigates the small cell problem inherent in classical Cut-Cells methods. The first method (FRM) employs a flux redistribution technique to ensure the global conservation of all conserved quantities (mass, momentum, and energy) across the entire domain. The second (FIM) and third (FCM) methods utilize optimization procedures to guarantee the local conservation of these quantities within each cell. These methods have been rigorously tested on various test cases, demonstrating results comparable to the Cut-Cells method in terms of accuracy, convergence, and conservation. The numerical results demonstrate that all three conservative methods (FRM, FIM, and FCM) effectively restore the conservation properties while maintaining accuracy and stability. The shock tube test case shows that the conservative methods achieve mass and energy conservation errors at machine precision, unlike the non-conservative methods (IBM and RCCM). In the transonic flow over a bump test case, the conservative methods exhibit smaller mass flow rate errors, and they capture the shock wave more accurately than the non-conservative methods. Finally, the confluence of two supersonic flows and the supersonic vortex flow test cases confirm that the conservative methods achieve convergence rates close to the expected rate for all norms, and the results are consistent with the non-conservative methods.

In summary, the three methods (FRM, FIM, and FCM) share the common goal of restoring conservation properties in the numerical solutions by modifying the fluxes at the cut-cells boundaries, which is why they all give results that are almost identical to the Cut-Cells method. However, they differ in their approach to achieving conservation. The FRM method is computationally the most efficient and the simplest to implement, but it only guarantees global conservation and does not ensure local conservation within each cell. The FIM method is generally conservative and stable, but in rare cases, it may not be fully conservative. It also requires a more complex optimization procedure, making it more computationally expensive than the FRM method. The FCM method is always locally conservative but may be less stable than the FIM method due to the perturbation of reconstructed solutions. For a general case, the FRM method is recommended due to its simplicity and efficiency in ensuring global conservation. However, for more complex cases where local conservation is critical, the FIM

or FCM methods may be preferred despite their higher computational cost.

The three methods are all based on a flux formulation that ensures conservation properties in the numerical solutions, making them straightforward to implement in three-dimensional simulations. In the context of viscous flows, the proposed methods can also be extended to incorporate the additional terms related to the viscous fluxes. These viscous terms do not affect the implementation of the proposed methods, as all methods are based on a flux formulation that can be easily extended to include viscous terms. Also, the inclusion of viscous terms introduces some dissipation to the solution, which can enhance the stability of the numerical methods. However, the reconstruction procedure employed in the three methods may impact the resolution of the boundary layer, necessitating further studies to investigate its effects on viscous flow simulations. Additionally, the Cartesian grid used in the proposed methods may not be suitable for resolving the boundary layer in viscous flows. Alternative grid configurations, such as a hybrid grid combining a body-fitted grid near the immersed boundary with a Cartesian grid in the rest of the domain, may be required to accurately capture viscous effects. These considerations highlight the complexity of extending the proposed methods to viscous flows, necessitating further research and investigation.

Future work will focus on developing second-order schemes using these methods and testing them on more complex inviscid test cases to further assess their accuracy and robustness.

# CHAPTER 5 ARTICLE 2 : A SECOND ORDER ACCURATE CUT-CELLS METHOD FOR COMPRESSIBLE FLOWS USING SEMI-IMPLICIT LEAST-SQUARES RECONSTRUCTION FOR BOUNDARY CONDITIONS TREATMENT

El Hadji Abdou Aziz Ndiaye, Jean-Yves Trépanier, Renan De Holanda Sousa, Sébastien Leclaire, Manuscript under review in Computers and Mathematics with Applications,
Submission date: 23 January 2025

**Abstract**

This paper introduces a novel approach that uses a method called the Reconstructed Cut-Cells Method (RCCM) for addressing the small cells problem in the context of the Cut-Cells method applied to the Euler equations for inviscid compressible flows. The approach employs a second-order semi-implicit least-squares reconstruction to achieve stable and accurate solutions, even in regions with small cells, with the time step being limited only by the uncut cells. Additionally, an extension of the RCCM method, denoted as the Flux Redistribution Method (FRM), is presented to ensure global conservation. The paper is structured into three main parts: the presentation of the governing equations and their second-order finite volume discretization, a detailed description of the RCCM and FRM methods, and numerical results validating the efficacy of these methods through convergence studies and complex flow simulations. The method of manufactured solutions is used to rigorously assess the order of accuracy and the results demonstrate that the RCCM and FRM methods achieve second-order accuracy in the whole domain and at the boundaries taken separately. An application of the approach to a complex flow inside a simplified circuit breaker geometry is also presented, showcasing the method's effectiveness in simulating compressible flows in challenging computational domains.

Keywords : Compressible Flows; Cartesian Grids; Cut-Cells; Small Cells Problem; Second Order Accuracy; Least-Squares Reconstruction

## 5.1 Introduction

Most numerical methods for solving partial differential equations arising from fluid mechanics use a decomposition of the domain of interest into cells and then solve the equations on these cells. This decomposition process, known as mesh generation, can be quite complex

depending on the geometry of the domain and it can have a significant impacts on the quality of the numerical solution. For instance, a uniform Cartesian grid offers excellent discretization advantages, such as compact stencils and high-order accuracy, compared to unstructured grids. Unfortunately, Cartesian grids are not well-suited for non-rectangular geometries. Consequently, for complex geometries, other mesh generation techniques such as unstructured grids are often used. These other mesh generation techniques come with their own set of problems such as the difficulty to automate the mesh generation process and the time-consuming nature of this process.

One effective way to avoid the complexity of the mesh generation process is to force the use of a Cartesian grid and and modify the cells that intersect the domain boundary to conform to it. This approach is known as the *Cut-Cell* method. Regardless of the complexity of the geometry, the mesh generation process remains simple and straightforward.

The primary drawback of the Cut-Cells approach arises from the cut cells themselves located near the boundary. These cells often have irregular polygonal shapes and can be very small in size, leading to a deterioration in mesh quality and a significant loss of accuracy in the numerical solution. Small cells, in particular, pose a stability problem when using an explicit time integration scheme, a well-known issue in Cut-Cells literature referred to as the *small cells problem* [31].

Many techniques have been proposed to deal with the small cells problem. The most common one is the cell-merging technique [35, 37], which involves merging small cells with their neighboring cells to create larger cells. This approach eliminates small cells and the associated stability problems. However, the main disadvantage of this technique is the lack of a systematic way to merge cells, and the merging algorithms developed so far are not fully robust for complex 3D geometries [44, 59]. Another popular technique is the flux redistribution technique [49, 51, 52]. This technique consist of redistributing a fraction of the flux from the small cells to their neighbors such that the CFL condition based on the unrestricted time step is satisfied in the small cells region. The time update is then performed without any time step limitation but a loss of accuracy on the small cells region is observed. Specifically, with a second order scheme, order of accuracy is reduced to first order in the small cells region [52, 59]. The h-box method [57, 58] is another more complex technique that attempts to extend the domain of dependence of the small cells in order to avoid the stability problem. This method can avoid the small cells problem and preserve the order of accuracy simultaneously. However, it has been developed only for 2D geometries, and its extension to 3D has not yet been achieved. Other techniques have been proposed in the literature, but none have satisfactorily solved the small cells problem without any loss of accuracy in the small cells

region and without difficulty in extending the method to complex 3D applications [41,42,123].

In this paper, we propose a new approach to deal with the small cells problem in the context of the Cut-Cells method applied to the Euler equations for inviscid compressible flows. This approach employs a method called the *Reconstructed Cut-Cells Method* (RCCM), which uses a second order semi-implicit least-squares interpolation on the small cells region in order to have a stable and accurate solution. The method is inspired from the boundary condition treatment of sharp-interface immersed boundary methods [17, 76, 77, 81, 107]. A further extension of the RCCM method, called the *Flux Redistribution Method* (FRM), that allows to ensure global conservation of the conserved variables in the Euler equations is also presented. The main contributions of this paper is the demonstration of the RCCM and FRM methods effectiveness to handle the accuracy and stability issues associated with the small cells problem through convergence studies and complex flow simulations.

The paper is divided into three parts. In the first part, the governing equations and the second order finite volume discretization of the Euler equations are presented. In the second part, the RCCM method is described alongside the implementation details of the method. In the third and final part, numerical results for convergence study and solution of complex compressible flows are provided. The method of manufactured solutions [112,113,124] is used to rigorously assess the order of accuracy of the method and numerical results of a complex flow presenting many shocks patterns and flow discontinuities are provided for a simplified high-voltage circuit breaker geometry.

## 5.2 Numerical Method

In this section, the governing equations and their second order finite volume discretization will be presented.

### 5.2.1 Governing Equations

The two-dimensional Euler equations for inviscid compressible flows will be used throughout this paper and are given by Equation 5.1.

$$\frac{\partial}{\partial t}\int_{\Omega}\mathbf{U}d\Omega + \oint_{\partial\Omega}\mathbf{F}(\mathbf{U})dS = \int_{\Omega}\mathbf{Q}d\Omega \tag{5.1}$$

where $\mathbf{U}$, $\mathbf{F}$ and $\mathbf{Q}$ are respectively the vector of conservative variables, the flux vector and the source term vector. The conservative variables $\mathbf{U}$ and the flux vector $\mathbf{F}$ are given by the

relations defined in Equation 5.2.

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix} \qquad \mathbf{F} = \begin{bmatrix} \rho(\mathbf{v}\cdot\mathbf{n}) \\ \rho u(\mathbf{v}\cdot\mathbf{n}) + pn_x \\ \rho v(\mathbf{v}\cdot\mathbf{n}) + pn_y \\ \rho E(\mathbf{v}\cdot\mathbf{n}) + p(\mathbf{v}\cdot\mathbf{n}) \end{bmatrix} \tag{5.2}$$

In Equations 5.2, $\rho$ is the density, $\mathbf{v} = [u, v]$ the velocity vector, $E$ the total energy and $p$ the pressure. The perfect gas equation of state is used to close the system 5.1 and is given by the relation 5.3.

$$p = \rho(\gamma - 1)\left(E - \frac{1}{2}\|\mathbf{v}\|^2\right) \tag{5.3}$$

where $\gamma = c_p/c_v$ is the ratio of specific heats, $c_p$ and $c_v$ are the specific heats at constant pressure and volume respectively.

### 5.2.2 Second Order Finite Volume Method on Cut-Cells Grids

Figure 5.1 shows an example of a cut-cells grid where the cells at the boundary are clipped by the geometry. In order to simplify the presentation of the discretization, the cut-cells grid will be considered to be a set of cells $C_i$ of polygonal shapes separated by faces.

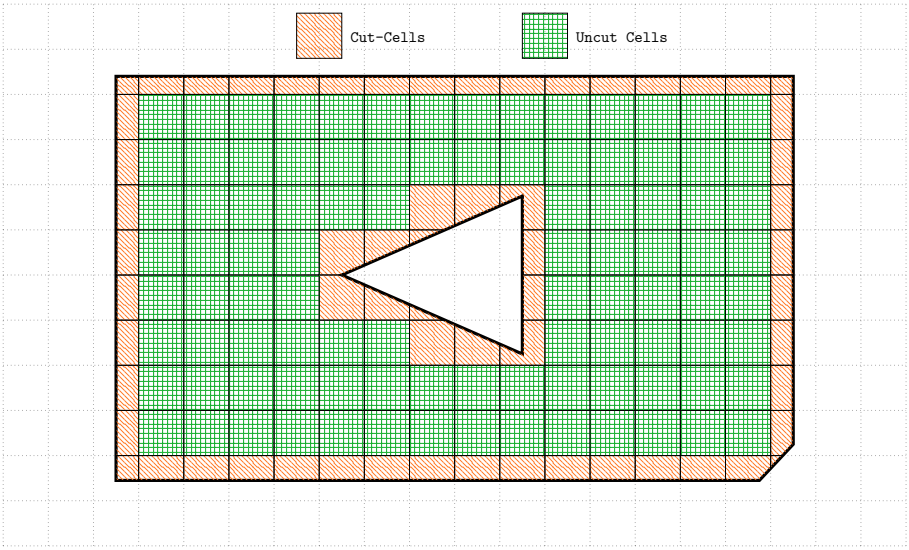



Figure 5.1 Example of a cut-cells grid.

For each cell $C_i$, $N(i)$ is the set of indices of the neighboring cells (cells that share a face with the cell $C_i$), and $A_{ij}$ is the area of the face between the cells $C_i$ and $C_j$. Hence, the

time-evolution of the conservative variables $\mathbf{U}_i$ of the cell $C_i$ is given by Equation 5.4.

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\Delta t^n}{\Delta V_i}\left(\sum_{j\in N(i)} \mathbf{F}_{i,j}^n \Delta A_{ij} + \mathbf{Q}_i \Delta V_i\right) \tag{5.4}$$

where $\Delta t^n$ is the time step, $\Delta V_i$ the volume of the cell $C_i$, $\mathbf{F}_{i,j}$ is the flux vector at the face $f_{ij}$ separating the cells $C_i$ and $C_j$ and $\mathbf{Q}_i$ is the source term vector of the cell $C_i$.

The flux vector $\mathbf{F}_{i,j}$ is computed using the approximate Riemann solver of Roe [115]. For a given face $f_{ij}$, the flux vector $\mathbf{F}_{i,j}$ is given as a function of the left and right states $\mathbf{U}_L$ and $\mathbf{U}_R$ by Equation 5.5.

$$\mathbf{F}_{i,j} = \frac{1}{2}\left(\mathbf{F}(\mathbf{U}_L) + \mathbf{F}(\mathbf{U}_R)\right) - \frac{1}{2}|\mathbf{R}_{ij}|(\mathbf{U}_R - \mathbf{U}_L) \tag{5.5}$$

where $|\mathbf{R}_{ij}|$ is the Roe matrix and is a function of the left and right states $\mathbf{U}_L$ and $\mathbf{U}_R$. The left and right states $\mathbf{U}_L$ and $\mathbf{U}_R$, given by Equations 5.6 and 5.7, are computed using a linear reconstruction of the conservative variables $\mathbf{U}_i$ at the face $f_{ij}$.

$$\mathbf{U}_L = \mathbf{U}_i + \nabla\mathbf{U}_i \cdot \mathbf{r}_{ij} \tag{5.6}$$

$$\mathbf{U}_R = \mathbf{U}_j + \nabla\mathbf{U}_j \cdot \mathbf{r}_{ji} \tag{5.7}$$

where $\mathbf{r}_{ij}$ and $\mathbf{r}_{ji}$ are the vectors from the centroids of the cell $C_i$ and $C_j$ to the centroid of the face $f_{ij}$ respectively. The vector $\nabla\mathbf{U}_i$ is the gradient of the conservative variables $\mathbf{U}_i$ and is computed using a linear least-squares interpolation of the conservative variables $\mathbf{U}_i$ in the cell $C_i$.

New extrema can be generated when using the direct reconstruction given by Equations 5.6 and 5.7 that can lead to the apparition of spurious oscillations in the numerical solution. These oscillations can introduce negative values of the density and the pressure which will lead to divergence of the numerical method. To avoid this problem, limiters such as the one proposed by Barth and Jespersen [125] and the one proposed by Venkatakrishnan [126] will be used. The general form of the limiter is given by Equation 5.8.

$$\mathbf{U}_L = \mathbf{U}_i + \psi_i \nabla\mathbf{U}_i \cdot \mathbf{r}_{ij} \tag{5.8}$$

where $\psi_i$ is the limiter function that will be computed using the Barth and Jespersen (BJ) or the Venkatakrishnan (VK) limiter which are given by Equations 5.9 and 5.10 respectively

[116, 125, 126].

$$\psi_i = \min_{j \in N(i)} \begin{cases} \min\left(1,\ \Delta_{i,M}/\Delta_{ij}\right) & \text{if } \Delta_{ij} > 0 \\ \min\left(1,\ \Delta_{i,m}/\Delta_{ij}\right) & \text{if } \Delta_{ij} < 0 \\ 1 & \text{otherwise} \end{cases} \tag{5.9}$$

$$\psi_i = \min_{j \in N(i)} \begin{cases} \frac{1}{\Delta_{ij}} \left((\Delta_{i,M}^2 + \epsilon^2)\Delta_{ij} + 2\Delta_{i,M}\Delta_{ij}^2\right) / \left(\epsilon^2 + 2\Delta_{ij}^2 + \Delta_{i,M}^2 + \Delta_{i,M}\Delta_{ij}\right) & \text{if } \Delta_{ij} > 0 \\ \frac{1}{\Delta_{ij}} \left((\Delta_{i,m}^2 + \epsilon^2)\Delta_{ij} + 2\Delta_{i,m}\Delta_{ij}^2\right) / \left(\epsilon^2 + 2\Delta_{ij}^2 + \Delta_{i,m}^2 + \Delta_{i,m}\Delta_{ij}\right) & \text{if } \Delta_{ij} < 0 \\ 1 & \text{otherwise} \end{cases} \tag{5.10}$$

where:

$$\begin{aligned} \mathbf{U}_M &= \max_{j \in N(i)} (\mathbf{U}_j, \mathbf{U}_i) \qquad \text{and} \qquad \mathbf{U}_m = \min_{j \in N(i)} (\mathbf{U}_j, \mathbf{U}_i) \\ \Delta_{i,M} &= \mathbf{U}_M - \mathbf{U}_i \quad , \quad \Delta_{i,m} = \mathbf{U}_m - \mathbf{U}_i \quad \text{and} \quad \Delta_{ij} = \nabla \mathbf{U}_i \cdot \mathbf{r}_{ij} \\ \epsilon^2 &= (K \Delta h_i)^3 \end{aligned}$$

The constant $K$ is a user-defined constant and $\Delta h$ is the characteristic length of the cell $C_i$. This completes the description of the evolution of the conservative variables $\mathbf{U}_i$ of the cell $C_i$.

For the time update, the third order Runge-Kutta method of Shu and Osher [127] will be used and is given by Equations 5.11.

$$\begin{aligned} \mathbf{U}^* &= \mathbf{U}^n - \frac{\Delta t^n}{\Delta V_i} \sum_{j \in N(i)} \left(\mathbf{F}_{i,j}^n \Delta A_{ij} + \mathbf{Q}_i \Delta V_i\right) \\ \mathbf{U}^{**} &= \frac{3}{4}\mathbf{U}^n + \frac{1}{4}\mathbf{U}^* - \frac{1}{4}\frac{\Delta t^n}{\Delta V_i} \sum_{j \in N(i)} \left(\mathbf{F}_{i,j}^* \Delta A_{ij} + \mathbf{Q}_i \Delta V_i\right) \\ \mathbf{U}^{n+1} &= \frac{1}{3}\mathbf{U}^n + \frac{2}{3}\mathbf{U}^{**} - \frac{2}{3}\frac{\Delta t^n}{\Delta V_i} \sum_{j \in N(i)} \left(\mathbf{F}_{i,j}^{**} \Delta A_{ij} + \mathbf{Q}_i \Delta V_i\right) \end{aligned} \tag{5.11}$$

The time step $\Delta t^n$ is computed using the CFL condition and is given by Equation 5.12.

$$\Delta t = \lambda \min_i \left(\frac{\Delta h_i}{\|\mathbf{v}_i\| + c_i}\right) \tag{5.12}$$

where $\lambda$ is the Courant number and $c_i$ is the maximum eigenvalue of the flux Jacobian matrix of the cell $C_i$. It is clear from Equation 5.12 that the time step $\Delta t$ is limited by the smallest cell of the grid. For a cut-cells grid, the smallest cell is usually very small and this can lead to a very small time step that can make the simulation unpractical. In this paper, the method that directly use this FVM discretization in every cells will be called the *Cut-Cells* (CC)

method. The CC method will be used as a reference to develop the RCCM method that will be presented in the following section.

## 5.3 Reconstructed Cut-Cells Method

In this section, the RCCM method will be described. It is divided into three parts. In the first part, the general idea of the method will be presented. After that, the reconstruction procedure that allows to obtain a stable and second order accurate solution will be described. Finally, an extension of the RCCM method called the *Flux Redistribution Method* (FRM), developed to ensure global conservation of the conserved variables in the Euler equations, will be presented [3].

### 5.3.1 Method Description

For an arbitrary geometry immersed in a Cartesian grid, the RCCM method classifies the cells of the grid into two categories: standard cells (`S-Cells`) and reconstructed cells (`R-Cells`). To identify the `R-Cells`, various criteria can be used, such as a threshold on the area of the cells. The only requirement is that the reconstructed cells must include all the small cells of the grid. The criteria used in this paper is to consider as `R-Cells`, all the cut-cells that have their center (of the uncut cell) outside the geometry. Figure 5.2 shows an example of a cut-cells grid used by the RCCM method, with the reconstructed cells highlighted in blue.

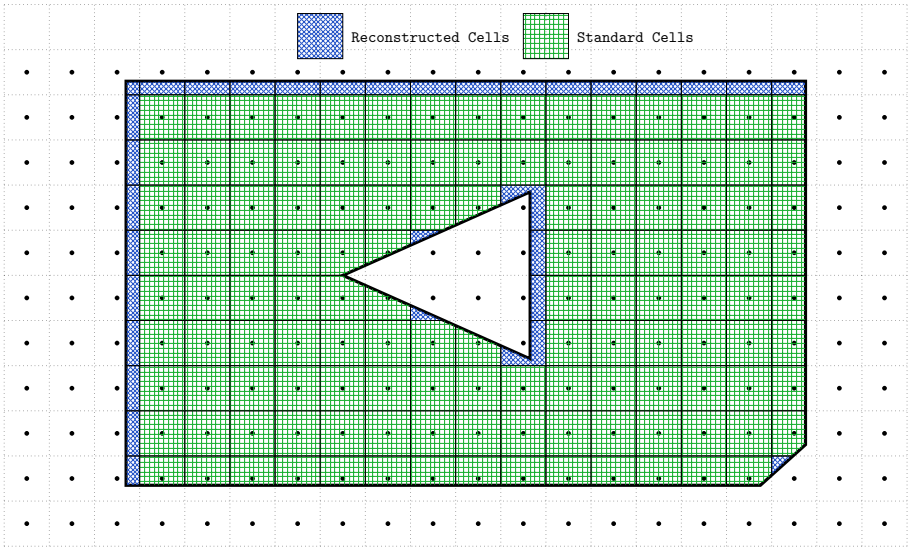



Figure 5.2 Example of a grid for the RCCM method.

The RCCM method functions as follow:

- At first, for the `S-Cells`, the standard FVM discretization is used as described in the previous section.

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\Delta t^n}{\Delta V_i} \sum_{j \in N(i)} \left( \mathbf{F}_{i,j}^n \Delta A_{ij} + \mathbf{Q}_i \Delta V_i \right) \qquad \forall\, i \in \texttt{S-Cells} \tag{5.13}$$

- At second, for the `R-Cells`, a second order semi-implicit least-squares reconstruction is used to compute the conservative variables.

$$\mathbf{U}_i^{n+1} = \texttt{reconstruction}(\mathbf{U}_j^{n+1}, \mathbf{U}_k^{n+1}, \ldots \,|\, j \in \texttt{R-Cells},\, k \in \texttt{S-Cells}) \quad \forall\, i \in \texttt{R-Cells} \tag{5.14}$$

For the FVM update of the `S-Cells`, the time step $\Delta t^n$ is computed using only the `S-Cells`, thus avoiding the restriction of the time step by the small cells. The use of a second order reconstruction on the `R-Cells` ensures a second-order accurate solution in the small cells region. Additionally, the implicit nature of the reconstruction and the avoidance of the FVM update on the `R-Cells` ensure a stable numerical solution.

### 5.3.2 Reconstruction Procedure

For each `R-Cells` $C_I$, the objective is to find the primitive variables $\mathbf{W}_I = [\rho_I, u_I, v_I, p_I]$. These primitive variables are then used to compute the conservative variables $\mathbf{U}_I$ using Equation 5.15.

$$\mathbf{U}_I = \left[ \rho_I, \quad \rho_I u_I, \quad \rho_I v_I, \quad \frac{p_I}{\gamma - 1} + \frac{1}{2} \rho_I \left( u_I^2 + v_I^2 \right) \right]^T \tag{5.15}$$

By considering $\phi_I$ to represent any of the primitive variables $\rho_I, u_I, v_I, p_I$ around the centroid of the cell $C_I$, it is supposed that $\phi_I$ can be approximated by a second order polynomial function $\tilde{\phi}_I$ centered at the centroid of the cell $C_I$ as shown in Equation 5.16.

$$\tilde{\phi}_I(x,y) = a_{\phi_I} + b_{\phi_I}(x - x_I) + c_{\phi_I}(y - y_I) + d_{\phi_I}(x - x_I)(y - y_I) + e_{\phi_I}(x - x_I)^2 + f_{\phi_I}(y - y_I)^2 \tag{5.16}$$

The task is to find the optimal values for the coefficients $a_{\phi_I}, b_{\phi_I}, c_{\phi_I}, d_{\phi_I}, e_{\phi_I}$ and $f_{\phi_I}$ using a least-squares approach. From Equation 5.16, it is evident that the value of the primitive variable $\phi_I$ at the centroid of the cell $C_I$ is directly given by the coefficient $a_{\phi_I}$. Therefore, for each `R-Cell` and each primitive variable, the coefficient $a_{\phi_I}$ is the primary value that needs to be computed.

To determine $a_{\phi_I}$ and the other coefficients, the idea is to use points $(x_{I_i}, y_{I_i})$ around the centroid of the cell $C_I$ where information on the primitive variables field $\phi$ are known. These points form the stencil of the least-squares reconstruction. The stencil points can either be the centroids of the neighboring cells or the centroids of the faces of the cell $C_I$ where boundary conditions are imposed.

The following describes the treatment of the stencil points and the assembly of the local least-squares system $A_{\phi_I}\mathbf{x}_{\phi_I} = \mathbf{b}_{\phi_I}$ for each `R-Cell` where $\mathbf{x}_{\phi_I} = (a_{\phi_I}, b_{\phi_I}, c_{\phi_I}, d_{\phi_I}, e_{\phi_I}, f_{\phi_I})^T$ represents the unknown of the system. It will be separated into two cases: the first case corresponds to the stencil points that are the centroids of the neighboring cells, and the second case corresponds to the stencil points that are the centroids of the faces of the cell $C_I$ where boundary conditions are imposed.

### Case 1: Centroids of the Neighboring Cells

If $(x_{I_i}, y_{I_i})$ is the centroid of a neighboring cell $C_{I_i}$ of the cell $C_I$, two scenarios can be considered:

- If the cell $C_{I_i}$ is a `S-Cell`, the value of the primitive variable $\phi_{I_i}$ is known from the FVM update.
- If the cell $C_{I_i}$ is an `R-Cell`, the value of the primitive variable $\phi_{I_i}$ is not known at the time of the reconstruction.

The second scenario implies the resolution of a global linear system of equations that involves all the `R-Cells` of the grid. The assembly of the global system will be described later in this section. In either scenario, the corresponding line in the local least-squares system $A_{\phi_I}\mathbf{x}_{\phi_I} = \mathbf{b}_{\phi_I}$ is given by Equation 5.17.

$$\begin{pmatrix} 1, & \Delta x_{I,I_i}, & \Delta y_{I,I_i}, & \Delta x_{I,I_i}\Delta y_{I,I_i}, & \Delta x_{I,I_i}^2, & \Delta y_{I,I_i}^2 \end{pmatrix} \begin{pmatrix} a_{\phi_I} \\ b_{\phi_I} \\ c_{\phi_I} \\ d_{\phi_I} \\ e_{\phi_I} \\ f_{\phi_I} \end{pmatrix} = \phi_{I_i} \tag{5.17}$$

where $\Delta x_{I,I_i} = x_{I_i} - x_I$ and $\Delta y_{I,I_i} = y_{I_i} - y_I$.

**Case 2: Boundary Points**

Depending on the boundary conditions, the information on the field $\phi$ can be of two types:

- Dirichlet boundary conditions: the value of the field $\phi$ is known and is used directly in the least-squares system; or
- Neumann boundary conditions: the value of the gradient of the field along the normal to the boundary $\nabla\phi \cdot \mathbf{n}$ is known and is used in the least-squares system

For example, if the boundary point corresponds to a supersonic inlet, all the primitive variables $(\rho, u, v, p)$ are given and are then used as Dirichlet boundary conditions. In contrast, if the boundary point corresponds to a supersonic outlet, the gradient of the primitive variables $(\nabla\rho \cdot \mathbf{n}, \nabla u \cdot \mathbf{n}, \nabla v \cdot \mathbf{n}, \nabla p \cdot \mathbf{n})$ are known (all equal to zero) and are then used as Neumann boundary conditions.

For a Dirichlet boundary condition, where the value $\phi_{I_i}$ of the field $\phi$ at the point $(x_{I_i}, y_{I_i})$ is known, the treatment is the same as for the centroids for a neighboring `S-Cell`.

For a Neumann boundary condition, where the value $\nabla\phi_{I_i} \cdot \mathbf{n}$ of the derivative along the normal of the field $\phi$ at the point $(x_{I_i}, y_{I_i})$ is known, Equation 5.18 is used to find the gradient of the field $\phi$ along the normal at the boundary point.

$$\begin{aligned}
\nabla\phi_{I_i} \cdot \mathbf{n} \approx \nabla\tilde{\phi}_I \cdot \mathbf{n}\Big|_{(x_{I_i}, y_{I_i})} &= \left(\frac{\partial\tilde{\phi}_I}{\partial x} n_x + \frac{\partial\tilde{\phi}_I}{\partial y} n_y\right)\Bigg|_{(x_{I_i}, y_{I_i})} \\
\left.\frac{\partial\tilde{\phi}_I}{\partial x}\right|_{(x_{I_i}, y_{I_i})} &= b_{\phi_I} + d_{\phi_I}(y_{I_i} - y_I) + 2e_{\phi_I}(x_{I_i} - x_I) \\
\left.\frac{\partial\tilde{\phi}_I}{\partial y}\right|_{(x_{I_i}, y_{I_i})} &= c_{\phi_I} + d_{\phi_I}(x_{I_i} - x_I) + 2f_{\phi_I}(y_{I_i} - y_I)
\end{aligned} \tag{5.18}$$

where $\mathbf{n} = [n_x, n_y]$ is the outward normal to the boundary at the boundary point $(x_{I_i}, y_{I_i})$. The corresponding line in the local least-squares system $A_{\phi_I}\mathbf{x}_{\phi_I} = \mathbf{b}_{\phi_I}$ is then given by Equation 5.19 for a Neumann boundary condition.

$$\begin{pmatrix} 0, & n_x, & n_y, & n_x\Delta y_{I,I_i} + n_y\Delta x_{I,I_i}, & 2n_x\Delta x_{I,I_i}, & 2n_y\Delta y_{I,I_i} \end{pmatrix} \begin{pmatrix} a_{\phi_I} \\ b_{\phi_I} \\ c_{\phi_I} \\ d_{\phi_I} \\ e_{\phi_I} \\ f_{\phi_I} \end{pmatrix} = \begin{pmatrix} \nabla\phi_{I_i} \cdot \mathbf{n} \end{pmatrix} \tag{5.19}$$

For practical applications, the boundary conditions are given physically (e.g., supersonic inlet, solid wall, etc.). Therefore, a transformation of the physical boundary conditions to the mathematical boundary conditions (Dirichlet or Neumann) is required. Table 5.1 shows the transformation of some physical boundary conditions used in this paper to the mathematical boundary conditions.

Table 5.1 Transformation from physical to mathematical boundary conditions.

| Physical boundary condition | Pressure $p$ | Density $\rho$ | Velocity $\mathbf{v}$ |
|---|---|---|---|
| Supersonic inlet | $p = p_\infty$ (D) | $\rho = \rho_\infty$ (D) | $\mathbf{v} = \mathbf{v}_\infty$ (D) |
| Supersonic outlet | $\nabla p \cdot \mathbf{n} = 0$ (N) | $\nabla \rho \cdot \mathbf{n} = 0$ (N) | $\nabla \mathbf{v} \cdot \mathbf{n} = 0$ (N) |
| Solid wall | $\nabla p \cdot \mathbf{n} = 0$ (N) | $\nabla \rho \cdot \mathbf{n} = 0$ (N) | $\nabla \mathbf{v} \cdot \mathbf{n} = 0$ (N) |
| Subsonic inlet | $\nabla p \cdot \mathbf{n} = 0$ (N) | $\rho = \rho(p, p_0, T_0, \alpha)$ (D) | $\mathbf{v} = \mathbf{v}(p, p_0, T_0, \alpha)$ (D) |
| Subsonic outlet | $p = p_{\text{out}}$ (D) | $\nabla \rho \cdot \mathbf{n} = 0$ (N) | $\nabla \mathbf{v} \cdot \mathbf{n} = 0$ (N) |

Note that for a subsonic inlet, one variable (the pressure) needs to be reconstructed first, and then the value of the other variables are computed using the value of the reconstructed pressure and the physical boundary values (total pressure $p_0$, total temperature $T_0$ and angle of attack $\alpha$).

This completes the description of the treatment of the points of the reconstruction stencil. Next, the assembly of the global system will be described.

### Assembly of the Global Least-Squares System

For each `R-Cells`, the local least-squares system $A_{\phi_I}\mathbf{x}_{\phi_I} = \mathbf{b}_{\phi_I}$ is formulated as shown in Equation 5.20.

$$\begin{pmatrix} 1, & \Delta x_{I,I_1}, & \Delta y_{I,I_1}, & \Delta x_{I,I_1}\Delta y_{I,I_1}, & \Delta x^2_{I,I_1}, & \Delta y^2_{I,I_1} \\ 1, & \Delta x_{I,I_2}, & \Delta y_{I,I_2}, & \Delta x_{I,I_2}\Delta y_{I,I_2}, & \Delta x^2_{I,I_2}, & \Delta y^2_{I,I_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1, & \Delta x_{I,I_n}, & \Delta y_{I,I_n}, & \Delta x_{I,I_n}\Delta y_{I,I_n}, & \Delta x^2_{I,I_n}, & \Delta y^2_{I,I_n} \\ 0, & n_{x_{n+1}}, & n_{y_{n+1}}, & n_{x_{n+1}}\Delta y_{I,I_{n+1}} + n_{y_{n+1}}\Delta x_{I,I_{n+1}}, & 2n_{x_{n+1}}\Delta x_{I,I_{n+1}}, & 2n_{y_{n+1}}\Delta y_{I,I_{n+1}} \\ 0, & n_{x_{n+2}}, & n_{y_{n+2}}, & n_{x_{n+2}}\Delta y_{I,I_{n+2}} + n_{y_{n+2}}\Delta x_{I,I_{n+2}}, & 2n_{x_{n+2}}\Delta x_{I,I_{n+2}}, & 2n_{y_{n+2}}\Delta y_{I,I_{n+2}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0, & n_{x_m}, & n_{y_m}, & n_{x_m}\Delta y_{I,I_m} + n_{y_m}\Delta x_{I,I_m}, & 2n_{x_m}\Delta x_{I,I_m}, & 2n_{y_m}\Delta y_{I,I_m} \end{pmatrix} \begin{pmatrix} a_{\phi_I} \\ b_{\phi_I} \\ c_{\phi_I} \\ d_{\phi_I} \\ e_{\phi_I} \\ f_{\phi_I} \end{pmatrix} = \begin{pmatrix} \phi_{I_1} \\ \phi_{I_2} \\ \vdots \\ \phi_{I_n} \\ \nabla\phi_{I_{n+1}} \cdot \mathbf{n} \\ \nabla\phi_{I_{n+2}} \cdot \mathbf{n} \\ \vdots \\ \nabla\phi_{I_m} \cdot \mathbf{n} \end{pmatrix} \tag{5.20}$$

Where $m$ represents the total number of points in the stencil for cell $C_I$, while $n$ is the count of those points not subject to Neumann boundary conditions. The stencil itself is composed of the centroids of its direct neighbors, as well as the centroids of those neighbors' neighbors

and multiple nearby boundary points. The total number of points in the stencil, $m$, varies for each `R-Cell` but is always greater than or equal to 6. Hence a least-squares approach given by Equation 5.21 is used to find the values for the coefficients $a_{\phi_I}, b_{\phi_I}, c_{\phi_I}, d_{\phi_I}, e_{\phi_I}$ and $f_{\phi_I}$.

$$\mathbf{x}_{\phi_I} = (A_{\phi_I}^T A_{\phi_I})^{-1} A_{\phi_I}^T \mathbf{b}_{\phi_I} = K_{\phi_I} \mathbf{b}_{\phi_I} \tag{5.21}$$

Where $K_{\phi_I} = (A_{\phi_I}^T A_{\phi_I})^{-1} A_{\phi_I}^T$ is a small $6 \times m$ matrix.

In Equation 5.21 that gives the solution of the local least-squares system $A_{\phi_I}\mathbf{x}_{\phi_I} = \mathbf{b}_{\phi_I}$, all values are known at the time of the reconstruction except for some values of the primitive variables $\phi_{I_j}$ that correspond to other `R-Cells`. Hence an implicit coupling of all the `R-Cells` is required in order to find the desired values of the primitive variables $\phi_I$ at each `R-Cells`.

Since the matrix $A_{\phi_I}$ is fixed and depends only on the coordinates of the stencil points, the system 5.20 can be pre-solved for each `R-Cells` at the pre-processing step of the simulation to obtain the matrix $K_{\phi_I}$. The values of the primitive variables $\phi_I$ can then be computed using Equation 5.22.

$$\phi_I = a_{\phi_I} = \sum_{i=1}^{n} K_{\phi_I}^{1,i} \phi_{I_i} + \sum_{i=n+1}^{m} K_{\phi_I}^{1,i} \nabla\phi_{I_i} \cdot \mathbf{n} \tag{5.22}$$

where $K_{\phi_I}^{1,i}$ is the element in the $i^{th}$ column and the first row of the matrix $K_{\phi_I}$.

By separating the known and unknown values of the primitive variables $\phi$, Equation 5.22 can be rewritten as shown in Equation 5.23.

$$\sum_{J=1}^{N} K^{I,J}\phi_J = \sum_{I_i \notin V} K_{\phi_I}^{1,i} \phi_{I_i} + \sum_{i=n+1}^{m} K_{\phi_I}^{1,i} \nabla\phi_{I_i} \cdot \mathbf{n} = b^I \tag{5.23}$$

Where:

- $N$ is the total number of `R-Cells` of the grid
- $V$ is the set of the neighboring `R-Cells` of the cell $C_I$
- The coefficients $K^{I,J}$ are the elements of the global matrix $K$ that couples all the `R-Cells` of the grid and are given by Equation 5.24.

$$K^{I,J} = \begin{cases} 1 & \text{if } J = I \\ -K_{\phi_I}^{1,i} & \text{if } J = I_i \in V \\ 0 & \text{otherwise} \end{cases} \tag{5.24}$$

Equation 5.23 corresponds to a row of the global system $K\mathbf{x} = \mathbf{b}$ where $\mathbf{x}$ is the vector of the primitive variables $\phi$ of all the `R-Cells` of the grid ($\mathbf{x} = [\phi_1, \phi_2, \ldots, \phi_N]^T$) Additionally, since the elements of the matrix $K$ depend only on the coordinates of the stencil points, it can be pre-assembled and decomposed (using a LU decomposition for example) at the pre-processing step of the simulation. Hence at each time step, the right-hand side $\mathbf{b} = [b^1, b^2, \ldots, b^N]^T$ of the global system is computed, and the system is efficiently solved using a simple forward and backward substitution. The vector $\mathbf{b}$ need to be computed at each time step since it depends on the values of the primitive variables $\phi$ at the newly updated `S-Cells`.

The described reconstruction procedure can be straightforwardly extended in three dimensions or to any order of accuracy by increasing the number of coefficients in the local reconstruction 5.16. Also, in two or three dimensions, the number of reconstructed cells $N$ will always be much smaller than the total number of standard cells of the grid (which are of the order of $N^2$ and $N^{3/2}$ respectively), thus making the time cost of the reconstruction procedure negligible compared to the FVM update of the standard cells.

This completes the description of the reconstruction procedure. Next, the Flux Redistribution Method (FRM), extension of the RCCM method, will be presented.

### 5.3.3 The Flux Redistribution Method

The RCCM method is not inherently conservative since the discretization of the conservative variables $\mathbf{U}_i$ of the `R-Cells` is performed using a least-squares reconstruction, which cannot be expressed as a conservative update given by Equation 5.4. This lack of conservation can lead to errors in capturing shocks and discontinuities in the flow [108]. To address this issue, a method called the Flux Redistribution Method (FRM) was proposed in a previous paper [3] dealing with conservation issues in the Immersed Boundary Method. The FRM method is a straightforward application of the flux redistribution technique proposed by Colella and *al.* in [52]. The following lines describe the FRM method.

At each time step, two solutions $\mathbf{U}_{\text{CC}}^{n+1}$ and $\mathbf{U}_{\text{RCCM}}^{n+1}$ are computed using the CC and the RCCM methods, respectively. In the computation of these two solutions, the same time step $\Delta t^{RCCM}$ corresponding to the RCCM method is used. The solution $\mathbf{U}_{\text{CC}}^{n+1}$ is thus unstable since the CFL condition may not be satisfied on the `R-Cells`. However, it is conservative since an FVM conservative update is used on all the cells of the grid. On the other hand, the solution $\mathbf{U}_{\text{RCCM}}^{n+1}$ is stable but not conservative since a reconstruction is used on the `R-Cells`. The idea of the FRM method is to compute a new solution $\mathbf{U}_{\text{FRM}}^{n+1}$ that is both conservative and stable by using the two solutions $\mathbf{U}_{\text{CC}}^{n+1}$ and $\mathbf{U}_{\text{RCCM}}^{n+1}$. The new solution $\mathbf{U}_{\text{FRM}}^{n+1}$ is computed

such that the global conservation relation 5.25 is satisfied.

$$\sum_{i \in C} \mathbf{U}^{n+1}_{\mathrm{FRM},i} \Delta V_i = \sum_{i \in C} \mathbf{U}^{n+1}_{\mathrm{CC},i} \Delta V_i \tag{5.25}$$

To achieve this, the mass deficit $\Delta M_i$ between the two solutions $\mathbf{U}^{n+1}_{\mathrm{CC},i}$ and $\mathbf{U}^{n+1}_{\mathrm{RCCM},i}$ for each cell $C_i$ given by Equation 5.26 is redistributed to the neighboring cells.

$$\Delta M_i = \mathbf{U}^{n+1}_{\mathrm{CC},i} \Delta V_i - \mathbf{U}^{n+1}_{\mathrm{RCCM},i} \Delta V_i \tag{5.26}$$

Note that the mass deficit $\Delta M_i$ is zero for the `S-Cells` since the two solutions $\mathbf{U}^{n+1}_{\mathrm{CC},i}$ and $\mathbf{U}^{n+1}_{\mathrm{RCCM},i}$ are the same for these cells. The method of redistributing the mass deficit $\Delta M_i$ is described as follows:

- For each Cell $C_i$, Initialize the solution $\mathbf{U}^{n+1}_{\mathrm{FRM},i} = \mathbf{U}^{n+1}_{\mathrm{RCCM},i}$
- For each `R-Cells` $C_i$, Compute the mass deficit $\Delta M_i$ using Equation 5.26
- For each `R-Cells` $C_i$, redistribute the mass deficit $\Delta M_i$ to itself and its neighboring cells using Equation 5.27

$$\mathbf{U}^{n+1}_{\mathrm{FRM},j} \leftarrow \mathbf{U}^{n+1}_{\mathrm{FRM},j} + \frac{\Delta V_j}{\sum_{k \in \{i, N(i)\}} \Delta V_k} \Delta M_i \quad \forall\, j \in \{i, N(i)\} \tag{5.27}$$

In Equation 5.27, the redistribution is weighted by the volume of the cells so that the small cells receive less mass than the large cells allowing them to keep their stability. By this construction, the final solution $\mathbf{U}^{n+1}_{\mathrm{FRM}}$ will be globally conservative and stable. This completes the description of the FRM method. In the next section, numerical results will be presented to verify the accuracy and the stability of the RCCM and FRM methods.

## 5.4 Numerical Results

This section presents four test cases to evaluate the accuracy and the stability of the RCCM and FRM methods. The first two test cases are steady-state problems where the analytical solution is known and are used to evaluate the accuracy of the RCCM and FRM methods. The last two test cases are unsteady problems for more complex flows and/or geometries and since no analytical solution is available, the results are compared to the results of from an Unstructured Finite Volume Method (UFVM) solver and from COMSOL Multiphysics. In the first test case, the method of manufactured solutions [112, 113, 124] is used to formally evaluate the accuracy of the RCCM and FRM methods using a smooth solution. In the

second test case, the confluence of two supersonic flows is considered to evaluate the ability of the RCCM and FRM methods to capture shocks and discontinuities. The third test case is a shock tube kind of problem where a Mach 2 shock wave is reflected on a cylinder, which is a complex unsteady problem involving shock waves and reflections. Finally, the fourth test case is similar to the third test case but with a more complex geometry and is used to evaluate the ability of the RCCM and FRM methods to handle complex geometries encountered in industrial applications.

### 5.4.1 Convergence Study using the Method of Manufactured Solutions

This section presents the results of a convergence study using the method of manufactured solutions (MMS) to evaluate the accuracy of the RCCM and FRM methods. It is divided into three subsections: the description of the geometry and the manufactured solution, the results of the convergence study, and the evaluation of the effect of the cut-cells size and location on the accuracy of the RCCM and FRM methods.

#### Geometry and Manufactured Solution

Figure 5.3 illustrates the geometry used for the convergence study: a square with side length $L = 1$.

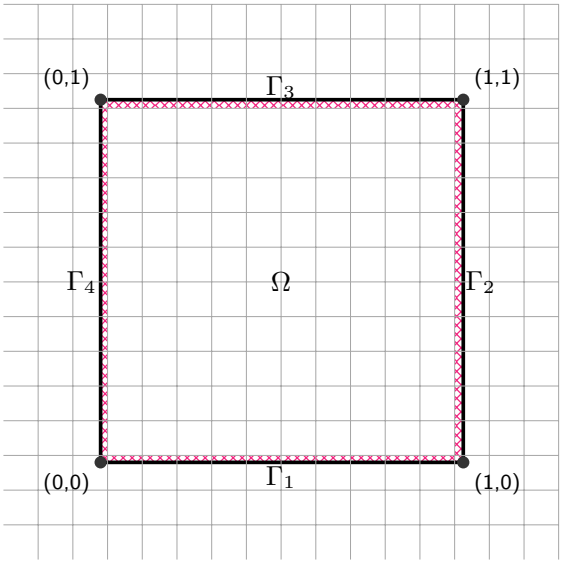



Figure 5.3 Geometry used for the convergence study.

Five grids with successive refinement levels will be used for the convergence study. The initial

grid is chosen such that with the successive refinements, the sum of the areas of the cut-cells highlighted in red in Figure 5.3 will remain constant. This constraint is imposed to ensure that the positions of the cut-cells are not modified during the grid refinement and the size of each cut-cell is uniformly reduced. Later in this section, this constraint will be relaxed to evaluate the effect of the cut-cells size and location on the accuracy of the RCCM and FRM methods.

The manufactured solution used for the convergence study is derived from [128] and is given by Equations 5.28. The fluid properties are $\gamma = 1.4$ and $R = 1.0$. The coefficients of the manufactured solution are provided in Table 5.2 and are chosen to ensure subsonic flow throughout the domain.

$$\begin{aligned}
\rho^{\text{MMS}}(x,y) &= \rho_0 + \rho_x \sin(a_{\rho_x}\pi\frac{x}{L}) + \rho_y \cos(a_{\rho_y}\pi\frac{y}{L}) + \rho_{xy}\cos(a_{\rho_{xy}}\pi\frac{x}{L})\cos(a_{\rho_{xy}}\pi\frac{y}{L}) \\
u^{\text{MMS}}(x,y) &= u_0 + u_x \sin(a_{u_x}\pi\frac{x}{L}) + u_y \cos(a_{u_y}\pi\frac{y}{L}) + u_{xy}\cos(a_{u_{xy}}\pi\frac{x}{L})\cos(a_{u_{xy}}\pi\frac{y}{L}) \\
v^{\text{MMS}}(x,y) &= v_0 + v_x \cos(a_{v_x}\pi\frac{x}{L}) + v_y \sin(a_{v_y}\pi\frac{y}{L}) + v_{xy}\cos(a_{v_{xy}}\pi\frac{x}{L})\cos(a_{v_{xy}}\pi\frac{y}{L}) \\
p^{\text{MMS}}(x,y) &= p_0 + p_x \cos(a_{p_x}\pi\frac{x}{L}) + p_y \sin(a_{p_y}\pi\frac{y}{L}) + p_{xy}\cos(a_{p_{xy}}\pi\frac{x}{L})\cos(a_{p_{xy}}\pi\frac{y}{L})
\end{aligned} \tag{5.28}$$

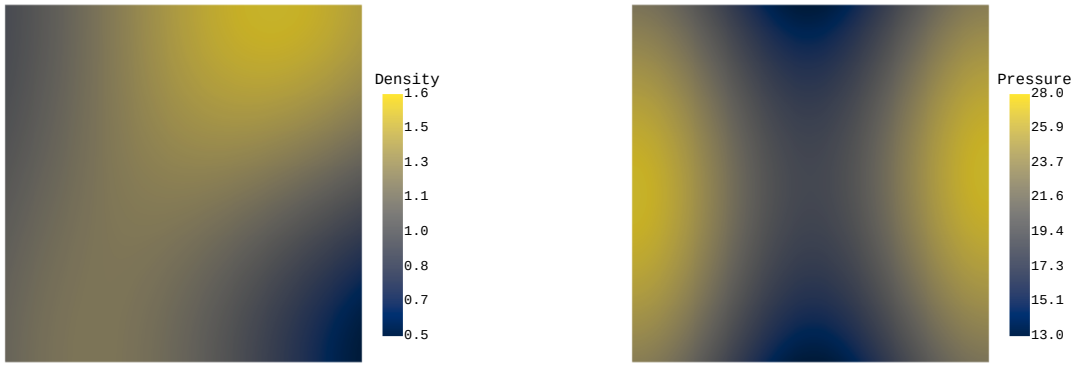
Table 5.2 Coefficients of the manufactured solution.

| **Variable** $\phi$ | $\phi_0$ | $\phi_x$ | $\phi_y$ | $\phi_{xy}$ | $a_{\phi_x}$ | $a_{\phi_y}$ | $a_{\phi_{xy}}$ |
|---|---|---|---|---|---|---|---|
| $\rho$ | 1.0 | 0.3 | $-0.2$ | 0.3 | 1.0 | 1.0 | 1.0 |
| $u$ | 1.0 | 0.3 | 0.3 | 0.3 | 3.0 | 1.0 | 1.0 |
| $v$ | 1.0 | 0.3 | 0.3 | 0.3 | 1.0 | 1.0 | 1.0 |
| $P$ | 18.0 | 5.0 | 5.0 | 0.5 | 2.0 | 1.0 | 1.0 |

Figure 5.4 shows the contours of the primitive variables $\rho^{\text{MMS}}$ and $p^{\text{MMS}}$ of the manufactured solution. The solution is smooth and exhibits significant variations within the computational domain.

By solving the Euler Equations 5.29 with the source terms defined by Equation 5.30, it becomes evident that the MMS solution is the exact solution, making it suitable for assessing the accuracy of the RCCM and FRM methods.

$$\frac{\partial}{\partial t}\int_\Omega \mathbf{U} d\Omega + \oint_{\partial\Omega} \mathbf{F}(\mathbf{U}) dS = \int_\Omega \mathbf{S}^{\text{MMS}} d\Omega \tag{5.29}$$




(a) Density $\rho^{\mathrm{MMS}}$.

(b) Pressure $p^{\mathrm{MMS}}$.

Figure 5.4 Contours of the density and pressure of the manufactured solution.

$$\mathbf{S}^{\mathrm{MMS}} = \begin{bmatrix} \partial_x \left(\rho^{\mathrm{MMS}} u^{\mathrm{MMS}}\right) + \partial_y \left(\rho^{\mathrm{MMS}} v^{\mathrm{MMS}}\right) \\ \partial_x \left(\rho^{\mathrm{MMS}} u^{\mathrm{MMS}} u^{\mathrm{MMS}} + p^{\mathrm{MMS}}\right) + \partial_y \left(\rho^{\mathrm{MMS}} u^{\mathrm{MMS}} v^{\mathrm{MMS}}\right) \\ \partial_x \left(\rho^{\mathrm{MMS}} v^{\mathrm{MMS}} u^{\mathrm{MMS}}\right) + \partial_y \left(\rho^{\mathrm{MMS}} v^{\mathrm{MMS}} v^{\mathrm{MMS}} + p^{\mathrm{MMS}}\right) \\ \partial_x \left(\rho^{\mathrm{MMS}} H^{\mathrm{MMS}} u^{\mathrm{MMS}}\right) + \partial_y \left(\rho^{\mathrm{MMS}} H^{\mathrm{MMS}} v^{\mathrm{MMS}}\right) \end{bmatrix} \tag{5.30}$$

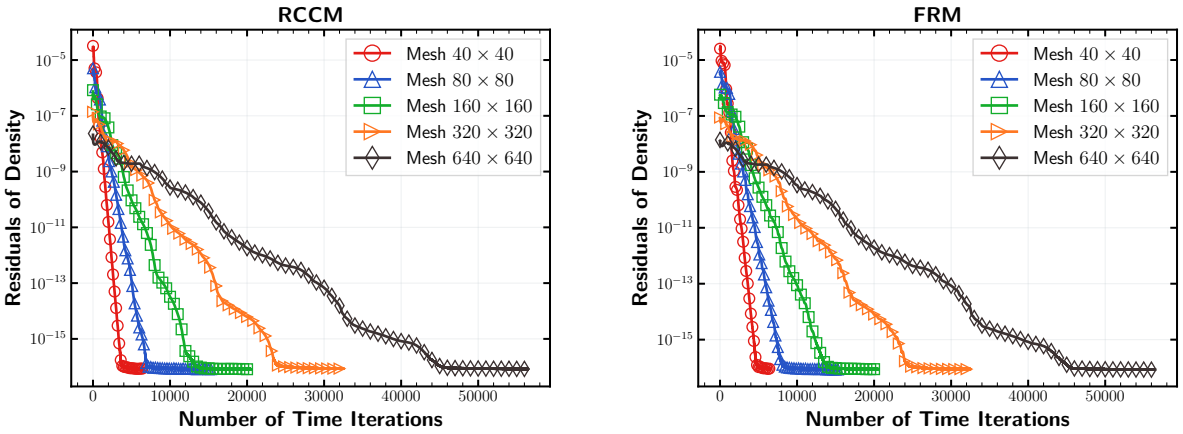
Using the RCCM and FRM methods without any limiter and the MMS solution as initial and Dirichlet boundary conditions on all four boundaries $(\Gamma_1, \ldots, \Gamma_4)$ of the domain, the convergence study is performed.

### Convergence Study

Figure 5.5 shows the convergence of density residual for the five grids. All residuals successfully converge to machine precision for both methods.

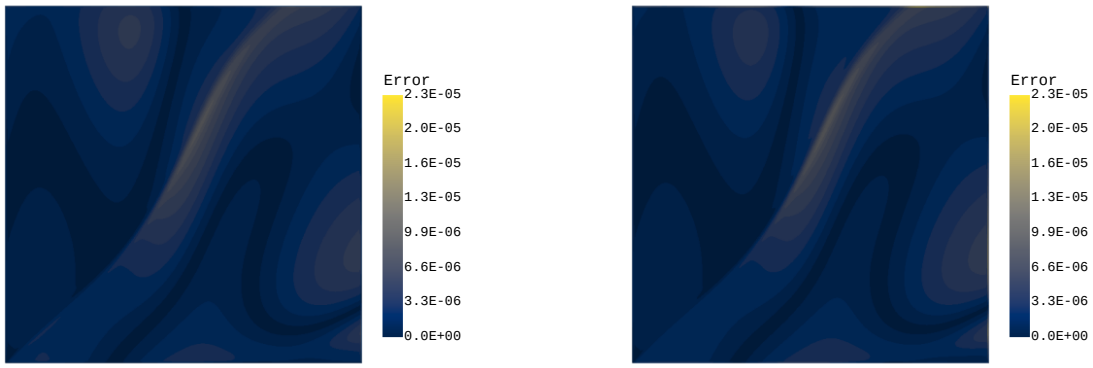
Figure 5.6 show the absolute error of the density for the finest grid using the RCCM and FRM methods. The errors do not appear to be concentrated at the boundary and are smoothly distributed throughout the domain.

Equations 5.31 allow for the computation of the $L_1$ and $L_\infty$ error norms of a variable $\phi$




(a) RCCM method. (b) FRM method.

Figure 5.5 Convergence of density residual for the MMS test case.




(a) RCCM density error $|\rho^{RCCM} - \rho^{MMS}|$. (b) FRM density error $|\rho^{FRM} - \rho^{MMS}|$.

Figure 5.6 Contours of the density error of the solution obtained with the last grid using the RCCM and FRM methods.

alongside the convergence rate $p$.

$$\begin{aligned} \epsilon_{L_1} &= \frac{1}{V}\sum_{i=1}^{N} \Delta V_i \left|\phi_i^{\text{numerical}} - \phi_i^{\text{MMS}}\right| \\ \epsilon_{L_\infty} &= \max_{1\leq i\leq N} \left|\phi_i^{\text{numerical}} - \phi_i^{\text{MMS}}\right| \\ p &= \log\left(\frac{\epsilon_{h_2}}{\epsilon_{h_1}}\right) / \log\left(\frac{h_2}{h_1}\right) \end{aligned} \tag{5.31}$$

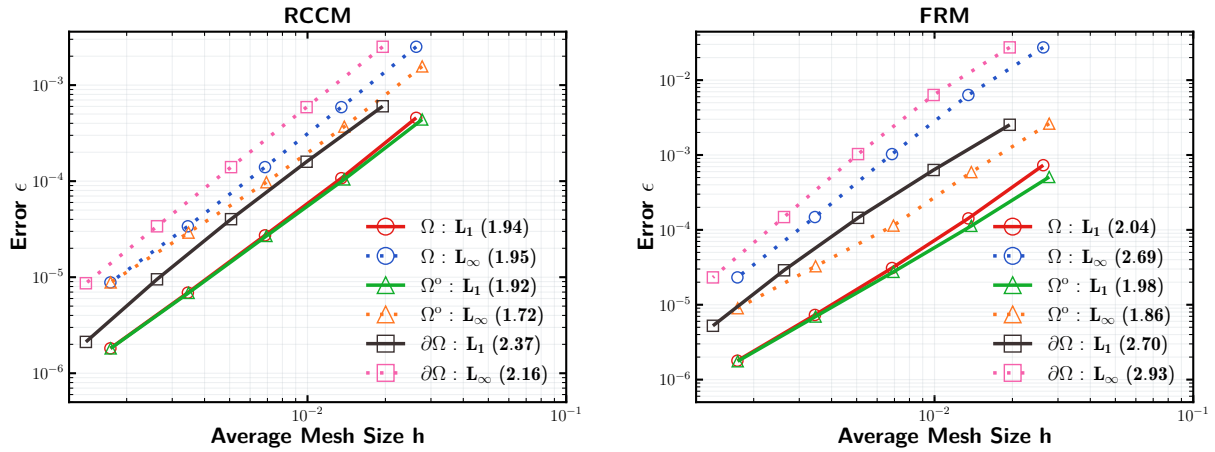
Where:

- $N$ is the number of cells
- $\Delta V_i$ is the volume of the cell $i$
- $V$ is the volume of the domain: $V = \sum_{i=1}^{N} \Delta V_i$
- $\phi_i^{\text{numerical}}$ is the numerical value of the variable $\phi$ at the cell $i$
- $\phi_i^{\text{exact}}$ is the exact value of the variable $\phi$ at the cell $i$
- $h$ is the characteristic grid size: $h = \sqrt{V/N}$
- $\epsilon_{h_1}$ and $\epsilon_{h_2}$ are the error norms ($L_1$ or $L_\infty$) at two different grid sizes $h_1$ and $h_2$

Figure 5.7 shows the convergence of the $L_1$ and $L_\infty$ error norms of the density for the five grids. The errors are computed in three regions: the whole domain ($\Omega$), the interior of the domain ($\Omega^o$) and the boundary of the domain ($\partial\Omega$).

The $L_1$ and $L_\infty$ error norms of the density converge with second-order accuracy for both the RCCM and FRM methods across the entire domain and within the interior and boundary of the domain taken separately as shown in Figure 5.7. For the $L_\infty$ error norm, since it is more sensitive, the order of convergence can be higher or slightly lower than 2.0 but remains on average close to 2.0. Comparing the two methods, the two methods exhibit similar convergence rates for the density across the three regions of the domain but the RCCM presents slightly lower errors than the FRM method. This is due to the fact that the FRM method is more diffusive than the RCCM method since it performs an additional redistribution step of the flux to ensure conservation.

Tables 5.3 and 5.4 summarize the different results of the convergence study for the two methods. The accuracy of the RCCM and FRM methods remains consistent for the other primitive variables $u$, $v$ and $p$.




(a) RCCM method.

(b) FRM method.

Figure 5.7 Convergence of the $L_1$ and $L_\infty$ error norms of the density for the MMS test case.

Table 5.3 Results of the convergence study for the MMS test case using the RCCM method.

| | RCCM | | | | | |
|---|---|---|---|---|---|---|
| | **Whole** | | **Interior** | | **Boundary** | |
| | $L_1$ | $L_\infty$ | $L_1$ | $L_\infty$ | $L_1$ | $L_\infty$ |
| $\rho$ | 1.94 | 1.95 | 1.92 | 1.72 | 2.37 | 2.16 |
| $u$ | 2.03 | 2.17 | 2.00 | 1.92 | 2.61 | 2.47 |
| $v$ | 1.89 | 1.95 | 1.88 | 1.81 | 2.27 | 2.13 |
| $p$ | 1.93 | 1.87 | 1.92 | 1.86 | 2.13 | 2.04 |

Table 5.4 Results of the convergence study for the MMS test case using the FRM method.

| | FRM | | | | | |
|---|---|---|---|---|---|---|
| | **Whole** | | **Interior** | | **Boundary** | |
| | $L_1$ | $L_\infty$ | $L_1$ | $L_\infty$ | $L_1$ | $L_\infty$ |
| $\rho$ | 2.04 | 2.69 | 1.98 | 1.86 | 2.70 | 2.93 |
| $u$ | 2.04 | 2.29 | 2.02 | 2.23 | 2.54 | 2.50 |
| $v$ | 2.03 | 2.31 | 1.99 | 1.92 | 2.57 | 2.52 |
| $p$ | 2.01 | 2.22 | 1.99 | 2.01 | 2.28 | 2.42 |

### Convergence Study with Arbitrarily Located Cut-Cells

The convergence study is repeated with the same manufactured solution and the same grid sizes but with the modified geometry shown in Figure 5.8 and Table 5.5 provides the properties of the resulting grids.

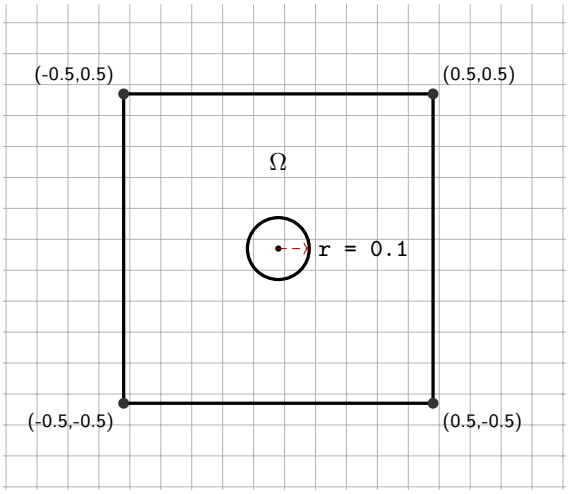


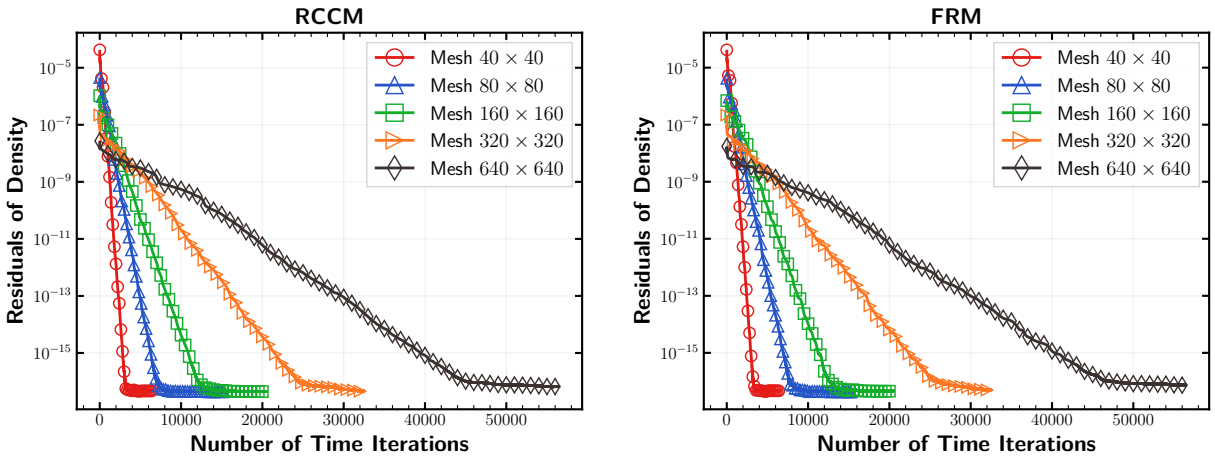

Figure 5.8 Modified geometry used for the convergence study.

Table 5.5 Properties of the grids used for the MMS convergence study with arbitrarily located cut-cells.

| **Grid** | **Grid size** | **Number of Uncut Cells** | **Number of Cut-Cells** | **Reconstructed Cells Area Ratio [%]** | **Factor max/min of Cells Area** |
|---|---|---|---|---|---|
| 1 | $40 \times 40$ | 1066 | 165 | 0.38 | 33.61 |
| 2 | $80 \times 80$ | 4920 | 343 | 1.62 | 101.40 |
| 3 | $160 \times 160$ | 18117 | 654 | 0.58 | 383.16 |
| 4 | $320 \times 320$ | 79096 | 1340 | 0.11 | 1252.82 |
| 5 | $640 \times 640$ | 317925 | 2567 | 0.25 | 1221.02 |

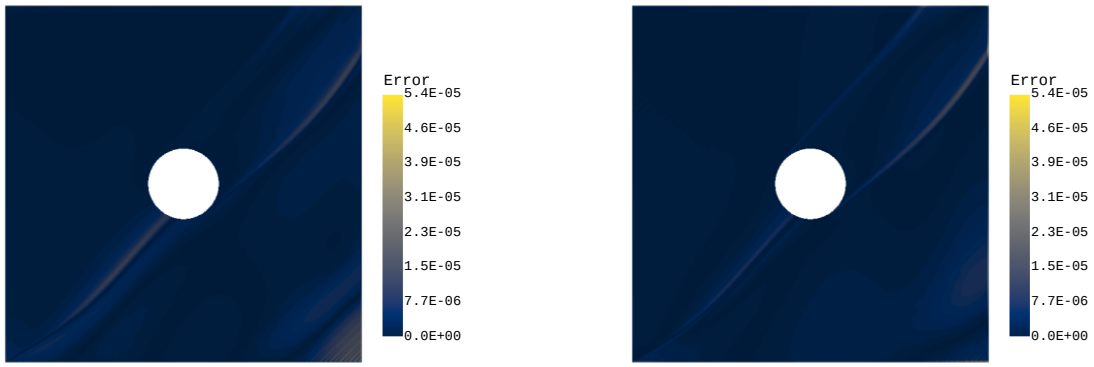
Figures 5.9 shows the convergence of the density residual for the five grids. As in the previous case, all residuals converge to machine precision for both methods.

Similarly, Figures 5.10 show the absolute error of the density for the finest grid using the RCCM and FRM methods. The errors are also smoothly distributed throughout the domain and do not appear to be concentrated at the exterior nor at the interior boundaries of the domain.




(a) RCCM method. (b) FRM method.

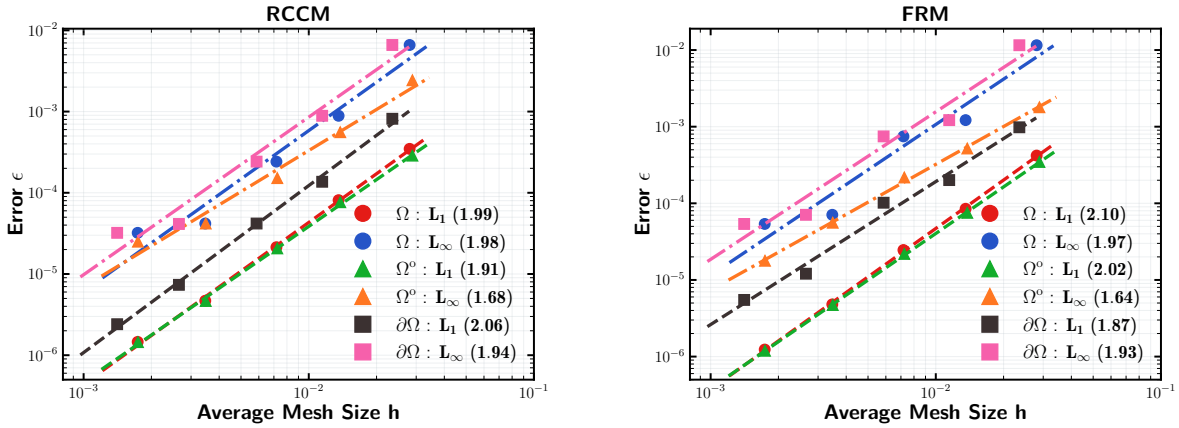
Figure 5.9 Convergence of density residual for the MMS test case with arbitrarily located cut-cells.




(a) RCCM density error $|\rho^{RCCM} - \rho^{MMS}|$. (b) FRM density error $|\rho^{FRM} - \rho^{MMS}|$.

Figure 5.10 Contours of the density error of the solution obtained with the last grid using the RCCM and FRM methods with arbitrarily located cut-cells.

Finally, Figures 5.11 show the convergence of the $L_1$ and $L_\infty$ error norms of the density for the five grids computed in the same three regions of the domain.




(a) RCCM method.

(b) FRM method.

Figure 5.11 Convergence of the $L_1$ and $L_\infty$ error norms of the density for the MMS test case with arbitrarily located cut-cells.

In the previous case, the orders of convergence were computed using only the last two grids. Since for this geometry, the cut-cells are arbitrarily located on the boundary and can change in size and location between two grids, the convergence is not as smooth and the orders are better computed using an average of multiple grids. Hence, in this case, the orders of convergence are computed using a linear regression with all the grids. With this metric, the resulting orders of convergence are approximately 2.0 for the $L_1$ and $L_\infty$ error norms for both methods and all three regions of the domain. For the $L_\infty$ error norm, the convergence is more sensitive to the location and size of the cut-cells and the order of convergence can be slightly lower than 2.0 in some regions. The orders of convergence for the other primitive variables $u$, $v$ and $p$ are similar to those obtained for the density.

### 5.4.2 Confluence of Two Supersonic Flows

Figure 5.12 illustrates the test case, which involves the collision of two supersonic flows, resulting in the formation of four regions of constant state. These regions are delineated by two shocks and one contact discontinuity.

The properties of the mesh used for this test case are detailed in Table 5.6. Note that the grid is rotated by 45 degrees to ensure that cut-cells may arbitrarily appear at the domain boundaries.

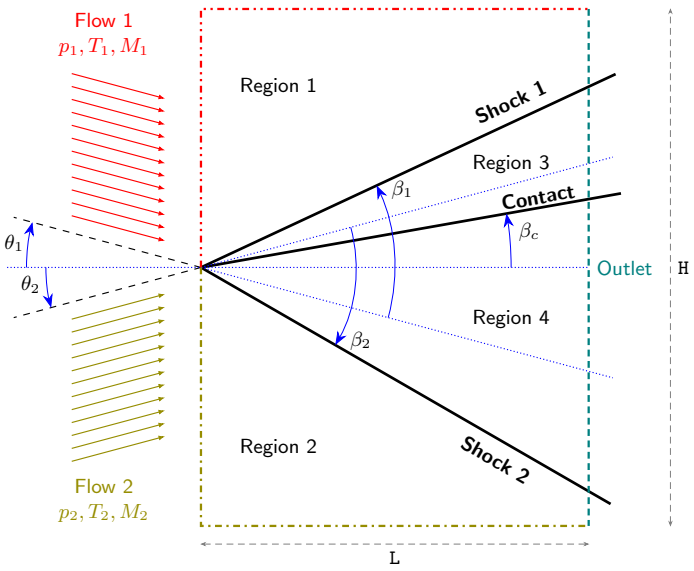



Figure 5.12 Geometry of the confluence of two supersonic flows [3].

Table 5.6 Properties of the grid used for the confluence of two supersonic flows.

| **Grid size** | **Number of Uncut Cells** | **Number of Cut-Cells** | **Reconstructed Cells Area Ratio [%]** | **Factor max/min of Cells Area** |
|---|---|---|---|---|
| $300 \times 300$ | 31199 | 1019 | 0.30 | 32.80 |

Given the presence of discontinuities in the flow, limiters are activated to prevent the generation of spurious oscillations or solver divergence. Figure 5.13 shows the convergence of the density residuals for both the RCCM and FRM methods.

It is observed that the residuals computed using the Barth-Jespersen (BJ) limiter stagnate and do not converge to machine precision. In contrast, the residuals computed using the Venkatakrishnan (VK) limiter converge to machine precision. This difference is attributed to the smoother nature of the VK limiter, which allows the residuals to reach machine precision. No significant difference is observed in the convergence of the residuals between the RCCM and FRM methods.

Figure 5.14 presents the Mach number contours for both methods using the BJ and VK limiters.

The results of the RCCM and FRM methods are in good agreement with the analytical solution. Both methods accurately capture the shocks and the contact discontinuity. However,

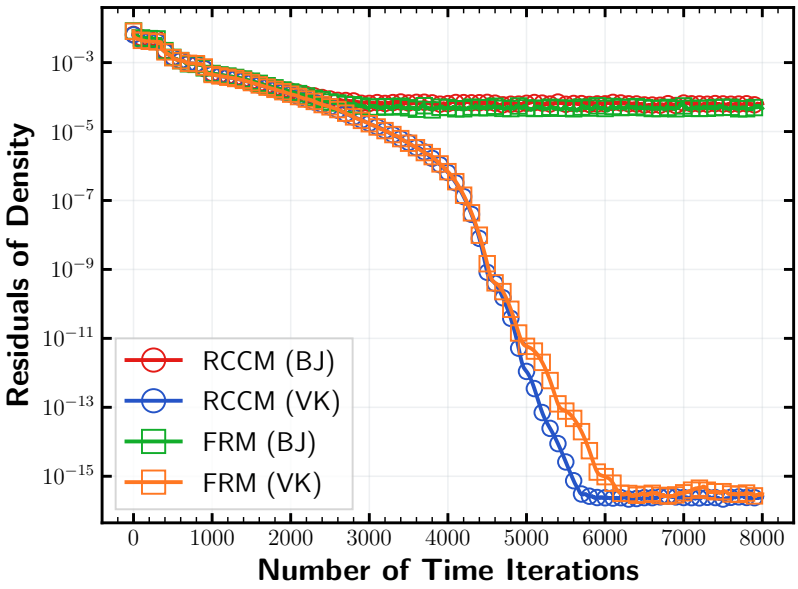
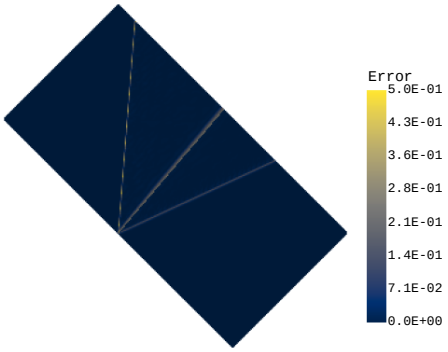



Figure 5.13 Convergence of the residuals of the density for the confluence of two supersonic flows.

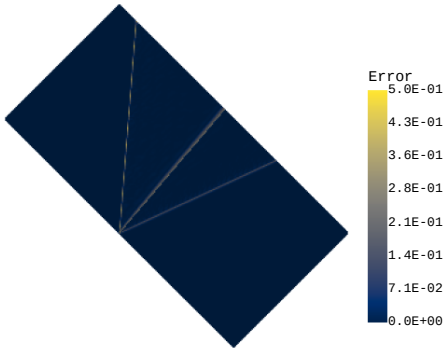
errors are more concentrated near the discontinuities, particularly the contact discontinuity. This can be seen more clearly in Figure 5.15 which shows the cut of the Mach number along the outlet boundary of the domain. The concentration of errors near the discontinuities can be attributed to the activation of limiters, which introduce some diffusion into the solution. It can be observed that the VK limiter even though it allows the residuals to converge to machine precision, introduces some oscillations in the solution near the discontinuities which are less pronounced with the BJ limiter. No significant difference is observed between the results of the RCCM and FRM methods which is consistent with the results obtained in reference [3]. Indeed, since the mesh used for this test case is sufficiently fine, the loss in conservation introduced by the RCCM method is negligible and thus the results of the two methods are similar. The following section will present a more complex test case to further evaluate the RCCM and FRM methods: the diffraction of a shock wave over a cylinder.

### 5.4.3 Diffraction of a Shock Wave over a Cylinder

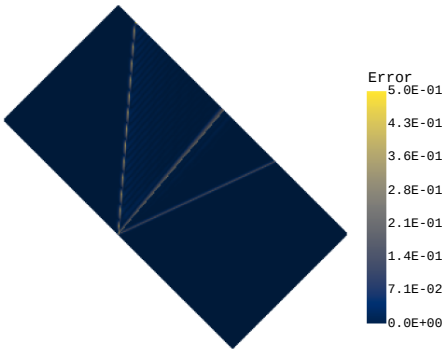
This test case correspond to a shock wave at Mach 2.0 diffracting over a cylinder of radius $r = 0.15$ [44]. The geometry is shown in Figure 5.16. The flow is initially at rest (state B) and the shock wave is positioned at $x_0 = 0.2$ and propagates from left to right. Equation 5.32 describes the initial conditions and boundary conditions of the problem and the properties



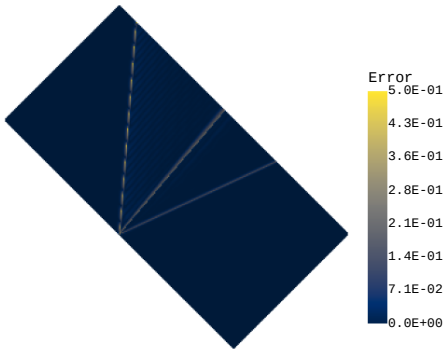

(a) RCCM (BJ).




(b) FRM (BJ).

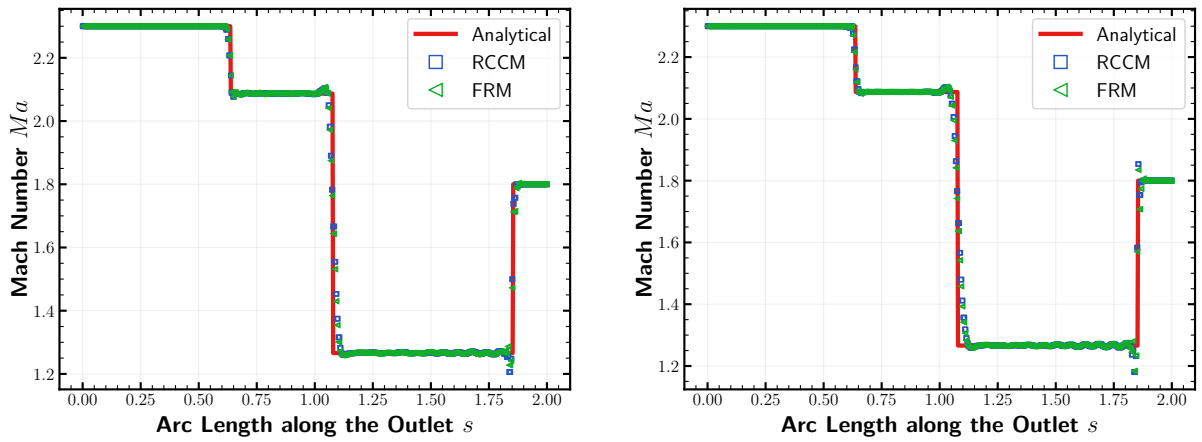



(c) RCCM (VK).




(d) FRM (VK).

Figure 5.14 Mach number error contours of the confluence of two supersonic flows using BJ and VK limiters.




(a) BJ limiter.

(b) VK limiter.

Figure 5.15 Cut of the Mach number along the outlet boundary of the domain for the confluence of two supersonic flows test case.

of the mesh used are given in Table 5.7.

$$\text{Initial conditions:} \begin{cases} \text{Pre-Shock State (B):} & p = 1.0, u = 0.0, v = 0.0, \rho = \gamma = 1.4 \\ \text{Post-Shock State (A):} & p = 4.5, u = 1.25, v = 0.0, \rho = 56/15 \end{cases}$$
$$\text{Boundary conditions:} \begin{cases} \text{Inlet } \Gamma_1\text{:} & \text{Dirichlet using the post-shock state (A)} \\ \text{Other boundaries:} & \text{Wall boundary condition} \end{cases} \tag{5.32}$$

Table 5.7 Properties of the grid used for the diffraction of a shock wave over a cylinder test case.

| **Grid size** | **Number of Uncut Cells** | **Number of Cut-Cells** | **Average Mesh Size** | **Factor max/min of Cells Area** |
|---|---|---|---|---|
| $366 \times 366$ | 102783 | 1738 | 0.00298 | 2271.60 |

Figure 5.17 shows the evolution of the density contours for the RCCM and FRM methods at different times. When the incident shock wave reaches the cylinder, a regular shock reflection pattern is observed along with the formation of a Mach stem and a slip line. A triple point is then formed at the intersection of the slip line and the Mach stem. The two methods capture these expected features accurately, with no significant difference between the results of the RCCM and FRM methods.

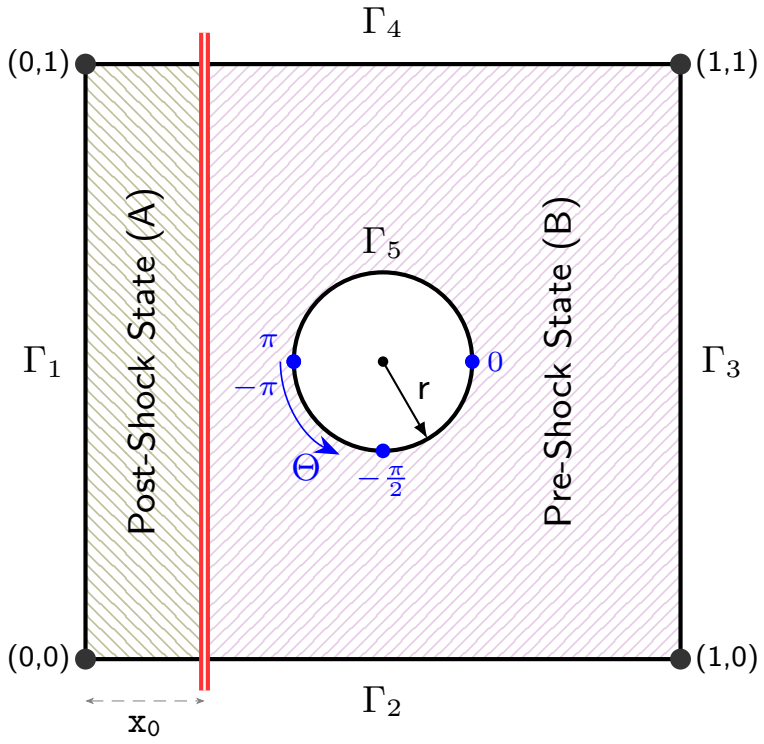



Figure 5.16 Geometry of the diffraction of a shock wave over a cylinder.

Figure 5.18 shows the cut of the density around the cylinder for the RCCM and FRM methods. The results of the two methods are compared with results obtained using the UFVM and COMSOL solvers. The UFVM and COMSOL solvers use triangular meshes with equivalent average cell size (0.00294 and 0.00164 respectively) as the Cartesian mesh used in the RCCM and FRM methods. It can be seen that the results provided by all four solvers are in good agreement. The RCCM and FRM methods, even though they use irregular cut-cells near the cylinder, provide results that are smooth and consistent with those obtained using the UFVM and COMSOL solvers.

The next section will present the results of the RCCM and FRM methods for a similar unsteady flow over a more complex geometry inspired by a circuit breaker.

### 5.4.4 Complex Unsteady Flow on a Simplified Circuit Breaker Geometry

The final test case involves a complex unsteady flow over a geometry inspired by a high-voltage circuit breaker. Figure 5.19 shows the geometry, which features a series of regions

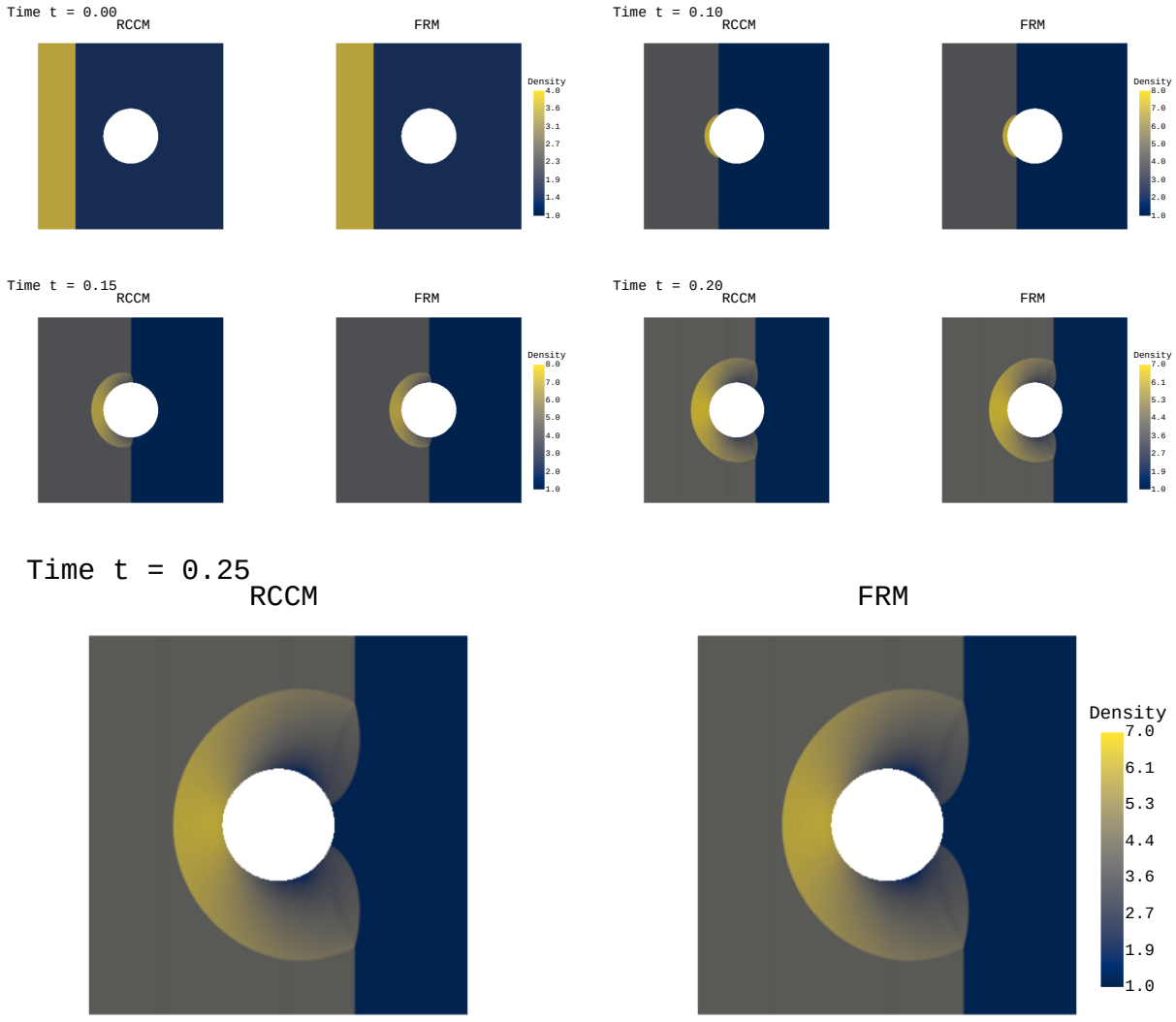



Figure 5.17 Density contours of the diffraction of a shock wave over a cylinder test case.

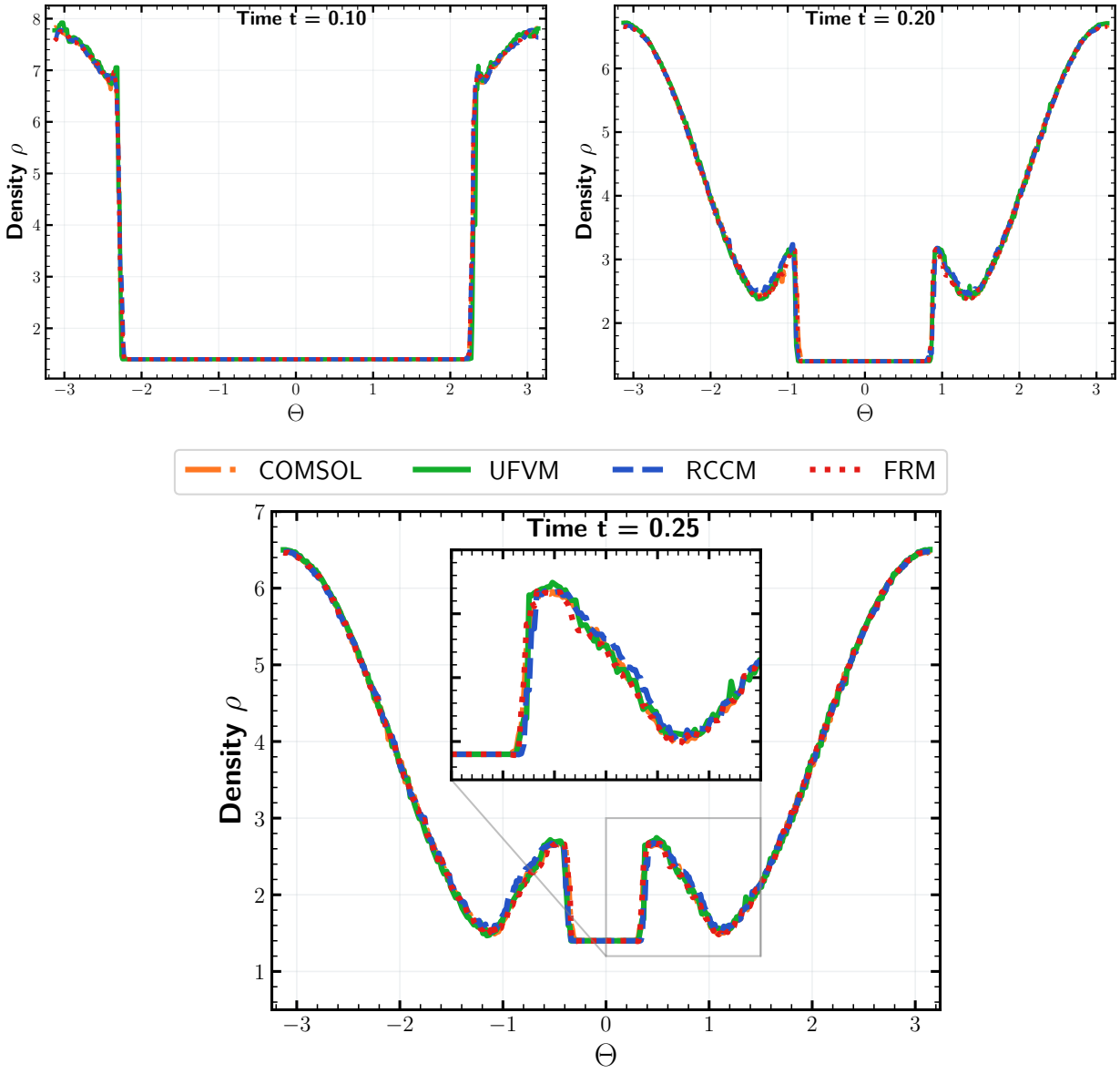



Figure 5.18 Cut of the density around the cylinder for the diffraction of a shock wave over a cylinder test case.

with different scales and forms, including sharp corners and narrow gaps. The choice of this geometry is motivated by the need to evaluate the RCCM and FRM methods in a challenging and realistic scenario. This test case combines the complexity of the geometry with the unsteadiness of the flow, making it a demanding benchmark for the methods.

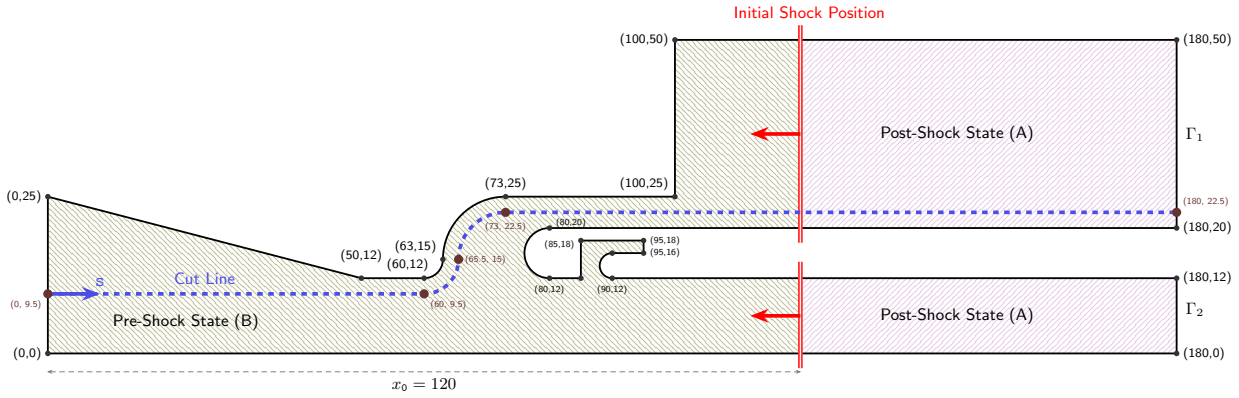



Figure 5.19 Geometry of the simplified circuit breaker test case.

Two shock waves at Mach 2.0 both are initially positioned at $x_0$ in the upper and lower regions of the domain and propagate from right to left. The flow is unsteady, and the shock wave interacts with the geometry and themselves, leading to complex flow patterns. The problem is parameterized as follows:

- Initial conditions:
  - Pre-Shock State (B): $p = 1.0$, $u = 0.0$, $v = 0.0$, $\rho = \gamma = 1.4$
  - Post-Shock State (A): $p = 4.5$, $u = -1.25$, $v = 0.0$, $\rho = 56/15$
- Boundary conditions:
  - Inlets $\Gamma_1$ and $\Gamma_2$: Dirichlet boundary condition with the post-shock state (A)
  - Other boundaries: Wall boundary condition
- Solvers parameters:
  - Limiter: Barth-Jespersen (BJ)
  - CFL number: 0.8
  - Final time: 50.0

Table 5.8 Properties of the grid used for the simplified circuit breaker test case.

| **Grid size** | **Number of Uncut Cells** | **Number of Cut-Cells** | **Average Mesh Size** | **Factor max/min of Cells Area** |
|---|---|---|---|---|
| $1800 \times 500$ | 436025 | 5696 | 0.10845 | 2427.67 |

The properties of the mesh used with the RCCM and FRM methods for this test case are given in Table 5.8.

Figures 5.20 and 5.21 show the evolution of the density contours for the RCCM method at different times. It can be observed that the flow is highly unsteady and complex, with the many flow patterns and structures forming and interacting with the geometry over time such as the shock and reflected shock waves, mach stem, slip lines, and vortices.

Figure 5.22 compares the density contours obtained with the RCCM, FRM and UFVM methods at the final time $t = 50.0$. The UFVM method use a body-fitted triangular mesh with an equivalent average cell size of 0.0681 (compared to the 0.10845 of the RCCM and FRM grid). The results show that the RCCM and FRM methods capture the main flow features and structures and are in good agreement with the UFVM method. The only noticeable difference is the presence of some numerical diffusion in the RCCM and FRM methods around the vortices.

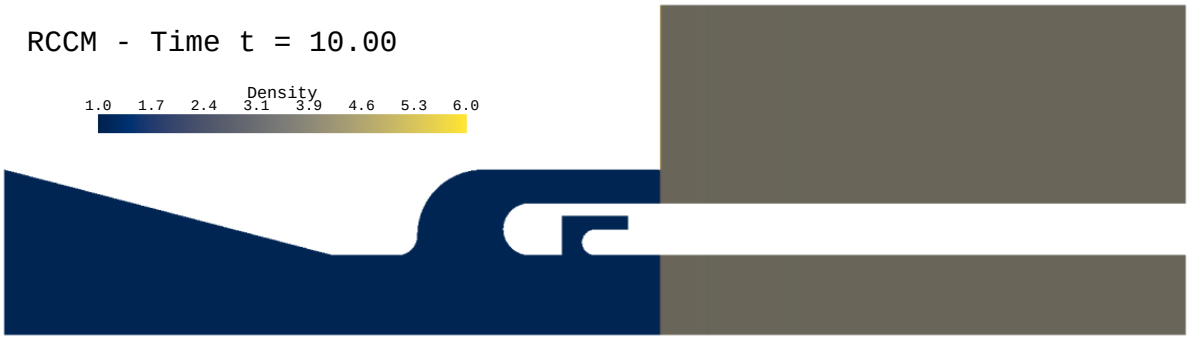
To better compare and evaluate the accuracy of the RCCM and FRM methods, Figure 5.23 shows the cut of the density along the median line of the domain drawn in blue in Figure 5.19. The results show that the three methods gives almost exactly the same results at early times. However, as the flow evolves, small differences start to appear between the methods but globally, the three methods remain very close.

To conclude this test case, a comparison of the conservation of mass and energy between the RCCM and FRM methods will be presented. From the conservative discretization form of the Euler Equations 5.4, the total mass and energy in the domain can be computed at each time step using the following equations:

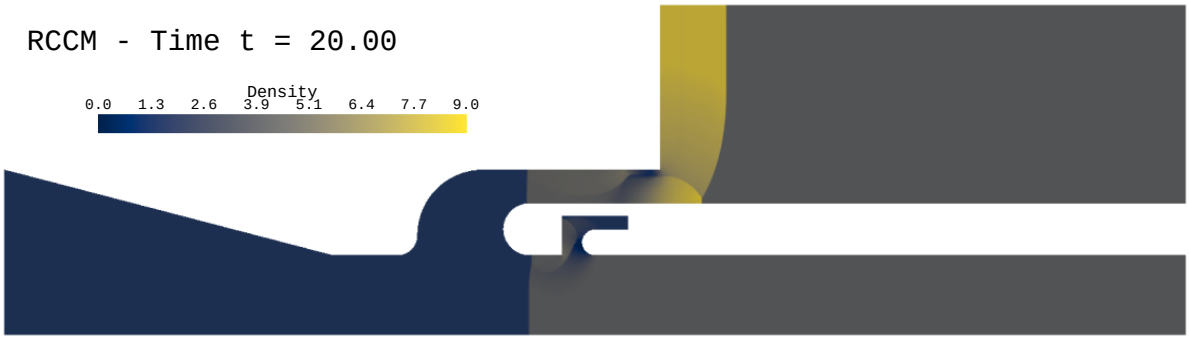
$$\begin{aligned}
\text{Mass:} \quad & M_T^{n+1} = \sum_{i=1}^{N} \rho_i^{n+1} \Delta V_i = \sum_{i=1}^{N} \rho_i^n \Delta V_i - \Delta t^n \oint_{\partial\Omega} \rho(\mathbf{v} \cdot \mathbf{n}) dS \\
& M_T^{n+1} = M_T^n - \Delta t^n \oint_{\partial\Omega} \rho \mathbf{v} \cdot \mathbf{n} dS \\
\text{Energy:} \quad & E_T^{n+1} = \sum_{i=1}^{N} (\rho E)_i^{n+1} \Delta V_i = E_T^n - \Delta t^n \oint_{\partial\Omega} (\rho E + p)(\mathbf{v} \cdot \mathbf{n}) dS
\end{aligned} \tag{5.33}$$


(a) $t = 0.0$.

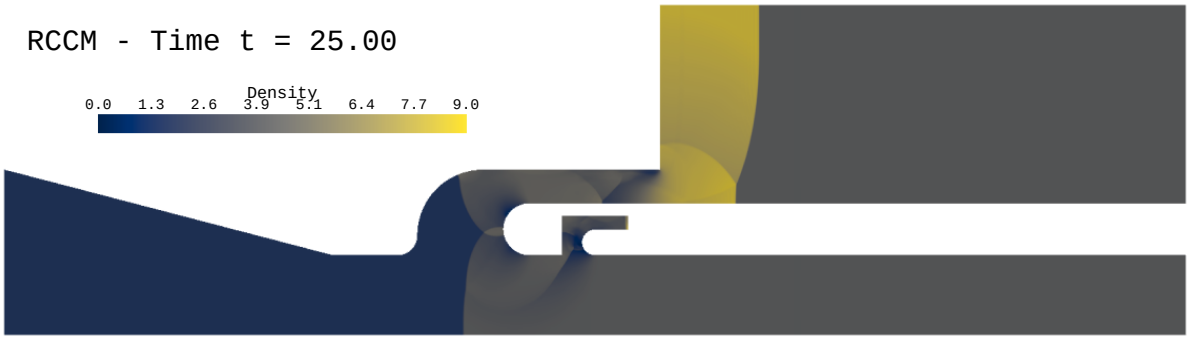

(b) $t = 5.0$.




(c) $t = 10.0$.

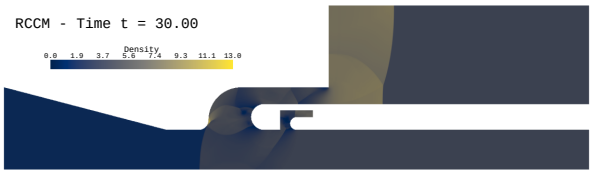



(d) $t = 20.0$.



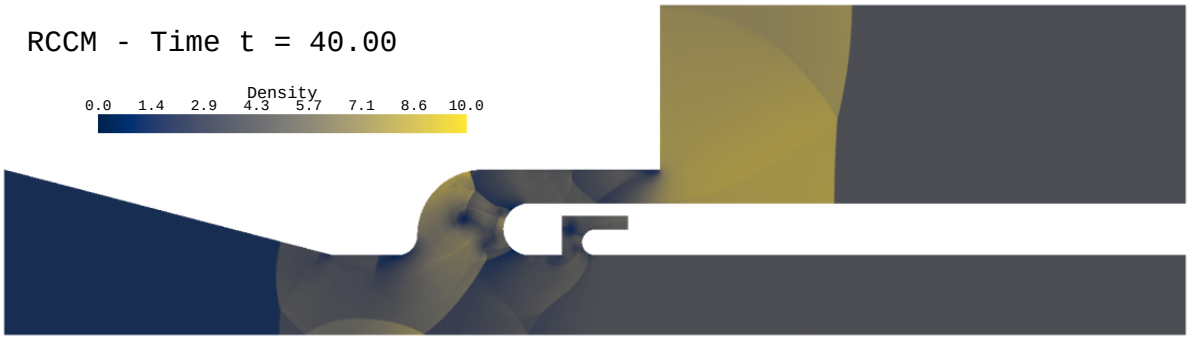

(e) $t = 25.0$.

Figure 5.20 Density contours for the simplified circuit breaker test case using the RCCM method at different times (1/2).

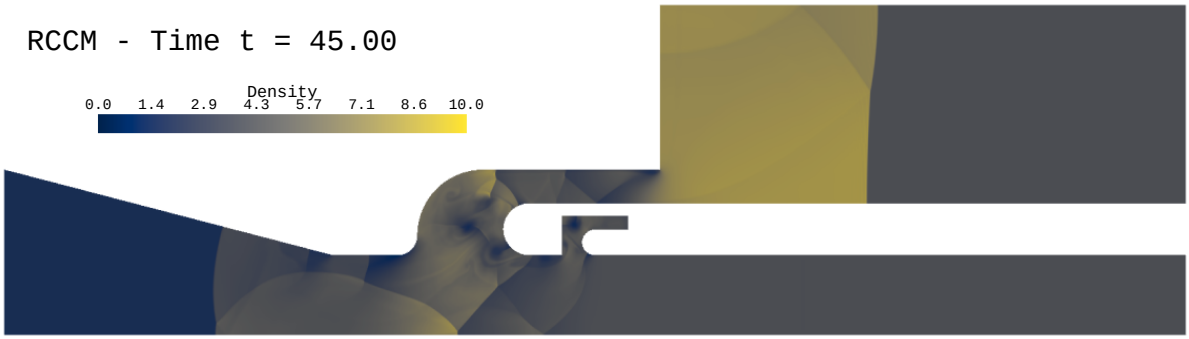



(a) $t = 30.0$.

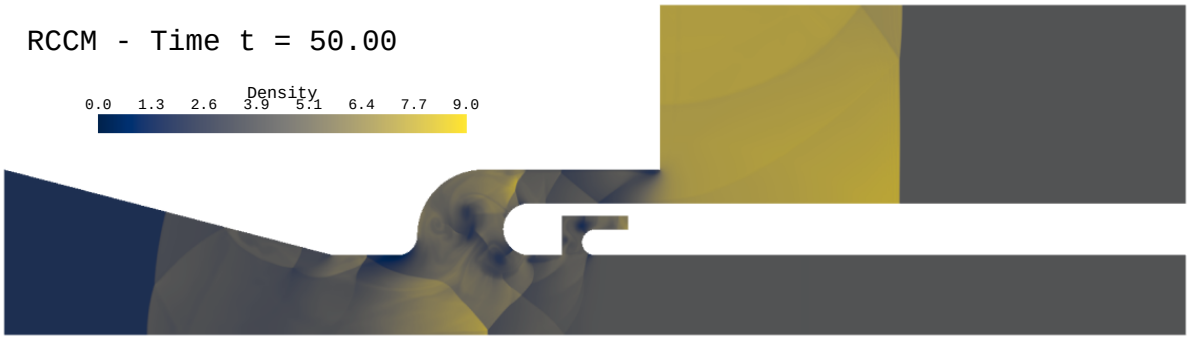

(b) $t = 35.0$.




(c) $t = 40.0$.




(d) $t = 45.0$.




(e) $t = 50.0$.

Figure 5.21 Density contours for the simplified circuit breaker test case using the RCCM method at different times (2/2).

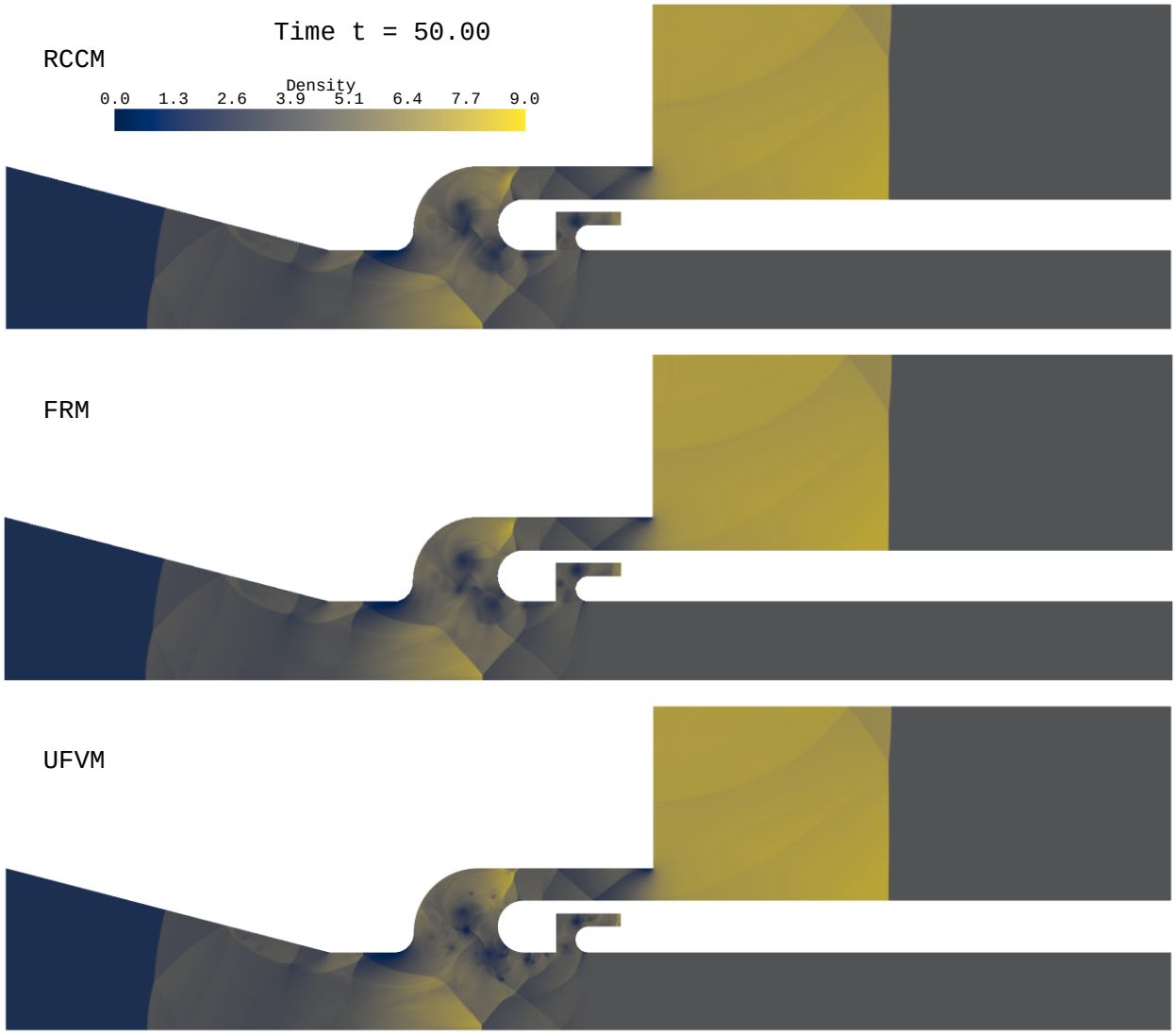



Figure 5.22 Density contours for the simplified circuit breaker test case using the RCCM, FRM and UFVM methods at $t = 50.0$.

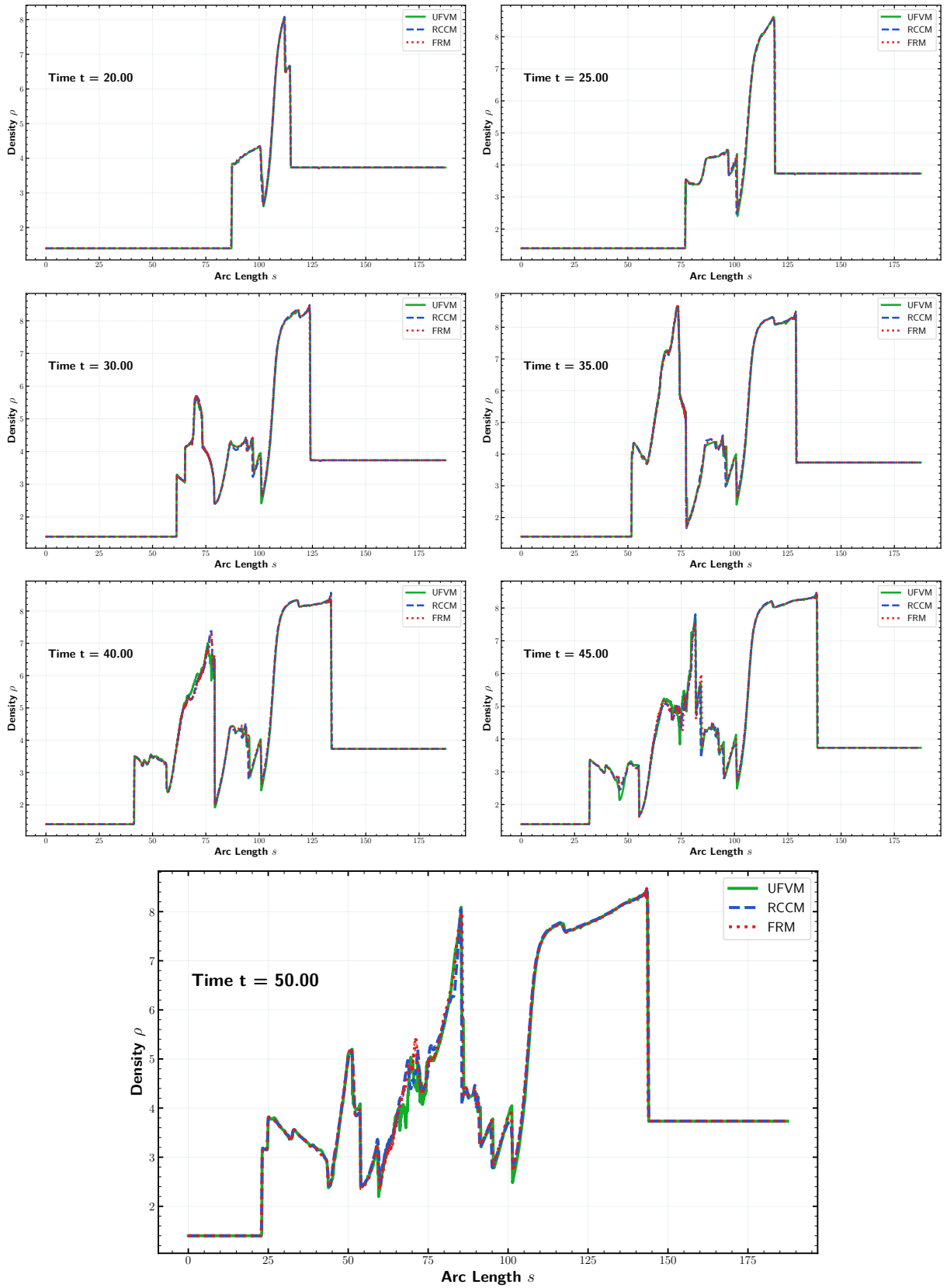



Figure 5.23 Cut of the density along the median line of the domain for the simplified circuit breaker test case using the RCCM, FRM and UFVM methods.

Since, for a Wall boundary condition, the normal velocity is zero, the mass and energy fluxes at the boundaries are constant and depend only on the Post-Shock state (A). Thus, by denoting the constant fluxes $M_{\partial\Omega} = \oint_{\partial\Omega} \rho \mathbf{v} \cdot \mathbf{n} dS$ and $E_{\partial\Omega} = \oint_{\partial\Omega} (\rho E + p)(\mathbf{v} \cdot \mathbf{n}) dS$, the conservation of mass and energy can be written as:

$$\begin{aligned} \text{Mass:} \quad & M_T^n = M_T^0 - t^n M_{\partial\Omega} \\ \text{Energy:} \quad & E_T^{n+1} = E_T^0 - t^n E_{\partial\Omega} \end{aligned} \tag{5.34}$$

Hence, the conservation of mass and energy can be evaluated by computing the relative errors of the mass and energy at each time step using the following equations:

$$\begin{aligned} \text{Mass error:} \quad & \epsilon_M^n = \frac{M_T^{n,*} - M_T^n}{M_T^0} \\ \text{Energy error:} \quad & \epsilon_E^n = \frac{E_T^{n,*} - E_T^n}{E_T^0} \end{aligned} \tag{5.35}$$

where $M_T^{n,*}$ and $E_T^{n,*}$ are the total mass and energy computed using the RCCM and FRM methods at time $t^n$. Figure 5.24 shows the evolution of the relative errors of the mass and energy for the RCCM and FRM methods. The results show that the FRM method is fully conservative and maintains the mass and energy conservation errors close to machine precision. Whereas, the RCCM method introduces some small errors in the conservation of mass and energy. These errors slightly increase over time but remain small (less than 1%) and do not affect the overall accuracy of the solution. A sudden jump in the mass and energy errors at $t = 10.0$ can be observed which is due to the collision of the shock in the upper region of the circuit breaker geometry with the vertical wall.

## 5.5 Conclusion

In this paper, we have introduced an approach based on the Reconstructed Cut-Cells Method (RCCM) and its extension, the Flux Redistribution Method (FRM), as an innovative solution to the small cells problem in the context of the Cut-Cells method applied to the Euler equations for inviscid compressible flows. The RCCM method employs a second-order semi-implicit least-squares reconstruction to achieve stable and accurate solutions, even in regions with small cells without any time step restriction based on the cut-cells size. The FRM method further enhances the RCCM by ensuring global conservation and maintaining second-order accuracy in the presence of small cells. Our numerical results demonstrate the effectiveness of the RCCM and FRM methods through rigorous convergence studies and complex flow simulations. The method of manufactured solutions was employed to assess the

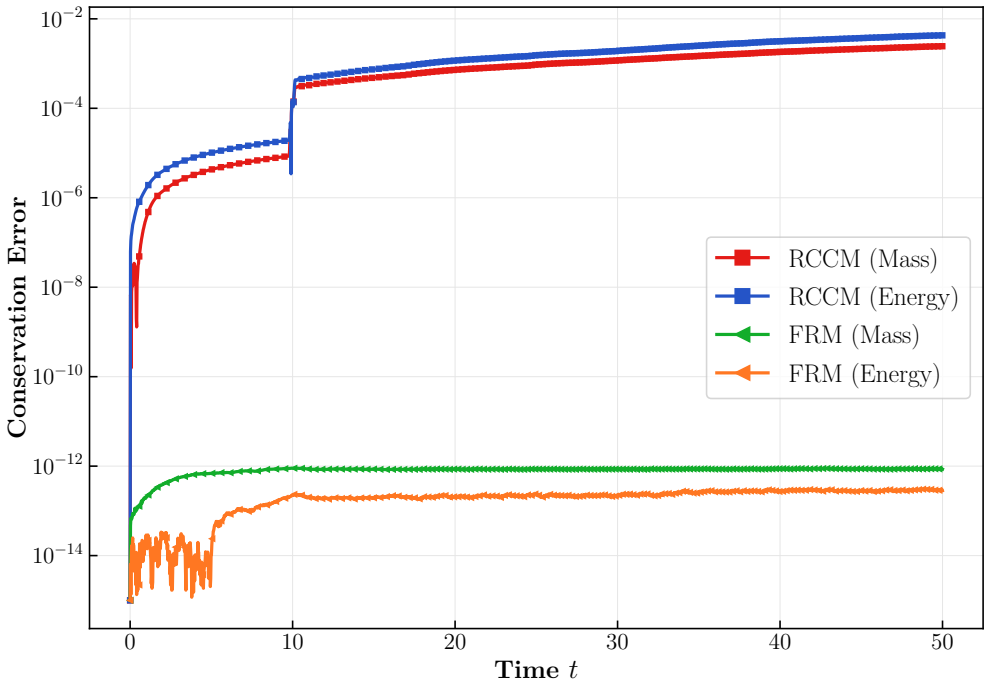



Figure 5.24 Evolution of the relative errors of the mass and energy for the simplified circuit breaker test case using the RCCM and FRM methods.

order of accuracy, confirming that both the RCCM and FRM methods achieve second-order accuracy in the whole domain and in the boundary taken separately. The RCCM and FRM methods provide a robust framework for simulating compressible flows in complex geometries, overcoming the limitations of traditional Cut-Cells methods. By addressing the small cells problem without sacrificing accuracy or stability, these methods offer a promising alternative for a wide range of applications in computational fluid dynamics. Future work will focus on extending these methods to three-dimensional geometries and for the simulation of moving boundaries problems.

# CHAPTER 6 THREE-DIMENSIONAL EULER SOLVER USING A CARTESIAN IBM METHOD

## Introduction

This chapter presents an implementation of the Immersed Boundary Method (IBM) for solving the Euler equations in three dimensions. In previous chapters, cut-cells methods developed in two dimensions has demonstrated both accuracy and effectiveness in addressing the conservation issues associated with the IBM method. The objective now is to extend these methods to three dimensions. The initial step toward this goal involves extending the basic IBM reconstruction to three dimensions, which forms the foundation of these methods. Subsequently, this IBM method will be integrated with a cut-cells grid to develop accurate and conservative methods for solving the Euler equations in three dimensions. This chapter only treats the basic IBM reconstruction in three dimensions and demonstrates its accuracy and convergence properties. The chapter is organized into two main parts: the first part details the numerical method, and the second part presents numerical results obtained using this method.

## 6.1 Numerical Methods

The IBM method with a Finite Volume Method (FVM) discretization is employed to solve the Euler equations in three dimensions. The method utilizes a Cartesian grid where cells located inside the computational domain are discretized using the FVM method, while cells located on the boundary are reconstructed using an interpolation method. This section is organized into three subsections. The first subsection presents the governing equations and mesh generation. The second subsection describes the FVM discretization of the governing equations. The third subsection details the reconstruction of boundary cells.

### 6.1.1 Governing Equations and Mesh Generation

The next three paragraphs present the governing equations, the discrete topology used to represent the computational domain, and the mesh generation process.

### Governing Equations

The governing equations are given by the Euler equations in three dimensions as shown in Equation 6.1.

$$\frac{\partial}{\partial t}\int_{\Omega}\mathbf{U}dV + \oint_{\partial\Omega}\mathbf{F}(\mathbf{U})dS = \int_{\Omega}\mathbf{Q}dV \tag{6.1}$$

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix}, \qquad \mathbf{F}(\mathbf{U}) = \begin{pmatrix} \rho(\mathbf{u}\cdot\mathbf{n}) \\ \rho u(\mathbf{u}\cdot\mathbf{n}) + pn_x \\ \rho v(\mathbf{u}\cdot\mathbf{n}) + pn_y \\ \rho w(\mathbf{u}\cdot\mathbf{n}) + pn_z \\ (\rho E + p)(\mathbf{u}\cdot\mathbf{n}) \end{pmatrix}$$

where $\Omega$ and $\partial\Omega$ represents the computational domain and its boundary, $\mathbf{U}, \mathbf{F}$ and $\mathbf{Q}$ are the vectors of conservative variables, fluxes and source terms respectively, $\mathbf{n}$ is the outward unit normal vector to the boundary, $\mathbf{u} = (u, v, w)$ represents the velocity vector, $\rho$ is the density, $E$ is the total energy, and $p$ is the pressure. The perfect gas equation of state, given by Equation 6.2, is used to close the system of equations.

$$p = (\gamma - 1)\rho\left[E - \frac{1}{2}\left(u^2 + v^2 + w^2\right)\right] \tag{6.2}$$

where $\gamma$ is the ratio of specific heats.

### Discrete Topology

The computational domain in which the equations are solved is defined using a boundary representation (b-rep) approach with a hierarchical data structure called *discrete topology*. Detailed information about the discrete topology is provided in Appendix A. Essentially, the discrete topology is a data structure that enables representation of a geometry through its boundary using a surface triangulation. The triangles are organized into topological entities called `Face`, `Shell` and `Volume` that provide a consistent representation of the geometry. The `Volume` is the main entity representing a computational domain. It contains a list of `Shell`s, which are closed surfaces that represent the boundary of the volume. If a `Volume` contains more than one `Shell`, the first `Shell` is considered to be the outer boundary of the volume, while the others are considered inner boundaries. The `Shell` entity contains a list of `Face`s with additional information about the orientation of the faces. The `Face` entity is simply a set of triangles that simplifies the construction of the `Shell` entity and is also used in the solver where boundary conditions are imposed at the `Face` level. Figure 6.1 shows

an example of a discrete topology composed of a single `Volume` with two `Shell`s. The outer `Shell` is a cube composed of six `Face`s, and the inner `Shell` is a sphere composed of one `Face`. The triangles are colored according to the face they belong to.

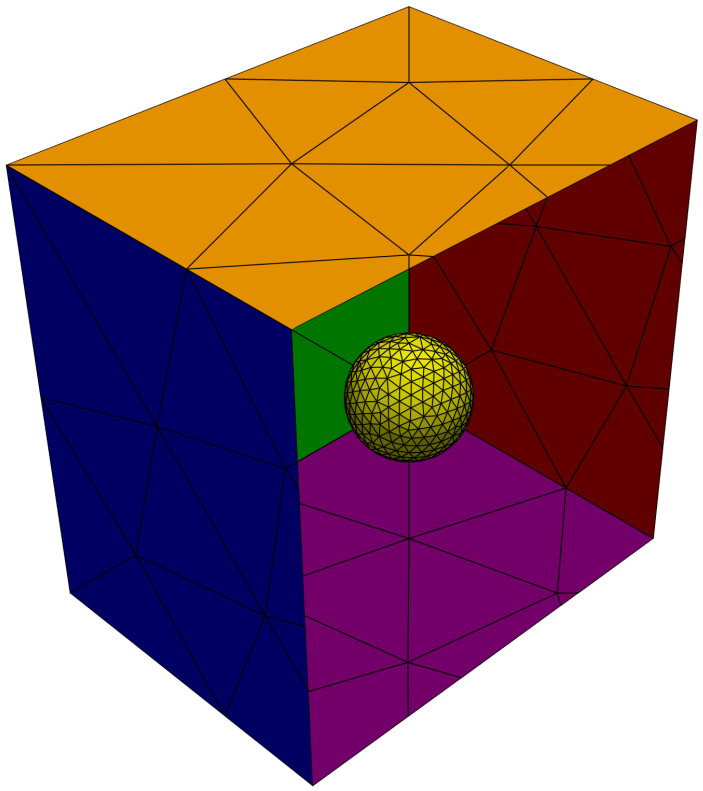


Figure 6.1 Illustration of a discrete topology composed of a single volume with two shells.

### Mesh Generation

The mesh is generated by immersing the discrete topology into a Cartesian grid. The Cartesian grid is initialized by specifying the number of cells in each of the three directions (i.e., grid size $N_x \times N_y \times N_z$) and the location of the grid lines. The grid line positions can be defined in various ways. For example, they can be defined by the minimum coordinates in the three directions and the cell size in each direction ($\Delta x$, $\Delta y$, $\Delta z$). In this work, the grid lines are defined using the bounding box of the discrete topology and the percentage by which to extend the grid lines in the six cardinal directions. For example, if the bounding box of the discrete topology in the $x$ direction is $[x^{\mathrm{t}}_{\mathrm{min}}, x^{\mathrm{t}}_{\mathrm{max}}]$, the maximum and minimum coordinates

of the grid lines in the $x$ direction (denoted as $x^{\mathrm{g}}_{\mathrm{min}}$ and $x^{\mathrm{g}}_{\mathrm{max}}$ respectively) are defined as:

$$x^{\mathrm{g}}_{\mathrm{min}} = x^{\mathrm{t}}_{\mathrm{min}} - \epsilon^{x}_{\mathrm{min}}(x^{\mathrm{t}}_{\mathrm{max}} - x^{\mathrm{t}}_{\mathrm{min}}), \quad x^{\mathrm{g}}_{\mathrm{max}} = x^{\mathrm{t}}_{\mathrm{max}} + \epsilon^{x}_{\mathrm{max}}(x^{\mathrm{t}}_{\mathrm{max}} - x^{\mathrm{t}}_{\mathrm{min}}) \tag{6.3}$$

where $\epsilon^{x}_{\mathrm{min}}$ and $\epsilon^{x}_{\mathrm{max}}$ are the percentages by which to extend the grid lines in the $x$ direction. The same procedure is applied to the other two directions. By using the bounding box of the grid and the grid size, the positions of the grid lines can be determined. This approach allows to easily generate a Cartesian grid around any discrete topology without the need to manually position the grid lines in the three-dimensional space. Once the grid is generated and positioned, the list of cells that are intersected by the discrete topology is computed, and each intersected cell stores the list of triangles that intersect it. An optional step in the mesh generation process is to refine grid cells locally using geometric criteria. Many criteria can be used for grid cell refinement, such as curvature, surface proximity, etc. The grid refinement maintains a constraint of 2:1 level ratio between two neighboring cells to allow for a smoother cell size transition. This means that if a cell needs to be refined, all of its coarser neighbors must be recursively refined before the refinement of the cell can be performed. If an intersected cell is refined, the list of triangles that intersect it is transferred to some of its eight children cells. The final step of grid generation is the tagging operation, which classifies cells into three main categories: `Inside`, `Outside` and `Boundary`. The boundary cells correspond overall to the cells that are intersected by the discrete topology, and their discretization must account for the boundary conditions. The inside cells are the cells that are completely located within the computational domain and are discretized with the FVM method. The outside cells are the cells that are completely located outside the discrete topology and are ignored by the solver. The list of tagged cells, along with the list of triangles that intersect each boundary cell, constitutes the mesh in which the governing equations are discretized. Figure 6.2 illustrates a mesh obtained with the topology shown in Figure 6.1. A simple refinement criterion based on the minimum refinement level of the intersected cells is used to refine the grid. A full description of the grid generation process is given in Appendix B.

After the mesh is generated, the final step is to discretize the governing equations using the IBM method. The discretization can be decomposed into two main steps: FVM discretization of the inside cells and reconstruction of the boundary cells. The next two sections are dedicated to the presentation of these two steps.

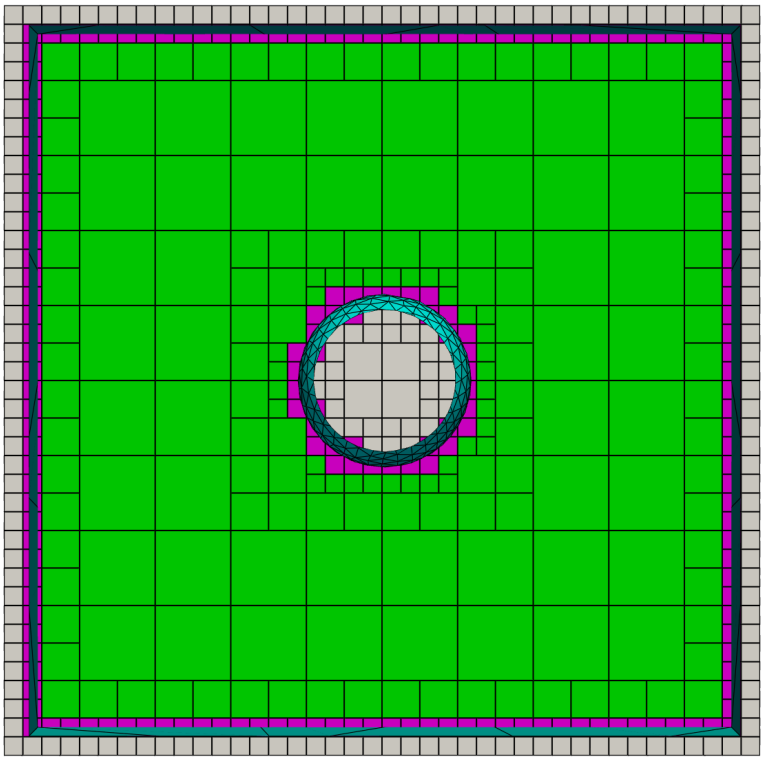


Figure 6.2 Example of a mesh generated from a simple discrete topology. The mesh is colored by the cell type: `Inside` (green), `Outside` (gray) and `Boundary` (magenta).

### 6.1.2 Finite Volume Discretization of the Inside Cells

Equation 6.4 shows the FVM discretization of the governing equations for an inside cell $C_i$.

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\Delta t^n}{\Delta V_i} \left[ \sum_{f \in \partial C_i} \mathbf{F}_f^n A_f + \mathbf{Q}_i^n \Delta V_i \right] \tag{6.4}$$

where $\mathbf{U}_i^n$ is the vector of conservative variables at time $t^n$ in cell $C_i$, $\Delta t^n$ is the time step, $\Delta V_i$ is the volume of cell $C_i$, $\partial C_i$ denotes the set of faces of cell $C_i$, $A_f$ is the area of face $f$, $\mathbf{F}_f^n$ represents the numerical flux vector at face $f$, and $\mathbf{Q}_i^n$ is the source term in cell $C_i$. The numerical flux vector can be computed using various methods such as Lax-Friedrichs [114], AUSM [129] or HLLC [130], etc. In the numerical results presented in this chapter, the approximate Riemann solver of Roe [115] is the scheme employed to compute the numerical flux vector. In the Roe method, the numerical flux is computed using two states, $\mathbf{U}_f^L$ and $\mathbf{U}_f^R$, defined at the left and right sides of the face $f$ (relative to the normal vector $\mathbf{n}$ of the

face). The flux vector is then calculated as:

$$\mathbf{F}_f = \frac{1}{2}\left[\mathbf{F}(\mathbf{U}_f^L) + \mathbf{F}(\mathbf{U}_f^R) - |\tilde{\mathbf{R}}|\left(\mathbf{U}_f^R - \mathbf{U}_f^L\right)\right] \tag{6.5}$$

where $\tilde{\mathbf{R}}$ is the Roe matrix defined as the Jacobian of the flux vector $\mathbf{F}$ at the Roe average state $\tilde{\mathbf{U}}(\mathbf{U}_f^L, \mathbf{U}_f^R)$. The left and right states are computed using linear least-squares reconstruction method with a limited slope computed using the Barth-Jespersen limiter [125]. For implementation purposes, although the mesh is semi-structured and neighboring cell information can be easily retrieved, it is simpler to treat the entire mesh as a set of unstructured cells and use a face-based data structure to compute the fluxes and update the cell states. This face-based approach eliminates redundant computation of the computationally expensive Roe fluxes and slope limiters for each face of the inside cells. Finally, the third-order Runge-Kutta method of Shu-Osher [127] is employed in conjunction with the FVM update of Equation 6.4 to advance the solution in time.

### 6.1.3 Reconstruction of the Boundary Cells

Once the inside cells are updated using the FVM method, the next step is to update the boundary cells solution using a reconstruction procedure that can be expressed in the form of Equation 6.6.

$$\mathbf{U}_i^{n+1} = \text{Reconstruction}(\mathbf{U}_i^{n+1}, \mathbf{U}_j^{n+1}, \text{BC}) \tag{6.6}$$

where indices $i$ and $j$ refer to the boundary cells and the inside cells respectively, and BC represents the boundary conditions imposed on the faces of the discrete topology. The following paragraphs present the boundary cells definition and reconstruction methodology.

#### Boundary Cell Definition

In contrast to previous chapters where boundary cells (or reconstructed cells) were defined as cells adjacent to the boundary (i.e., having at least one neighbor outside the solution domain) whose geometric centers lie inside the computational domain, this chapter adopts a slightly different definition that yields improved numerical results. Here, boundary cells are defined as all cells intersected by the discrete topology. This choice is inspired by the RCCM method, where the cut-cells themselves served as the reconstruction cells, using their centroids as reconstruction centers. However, in the standard IBM approach, detailed cut-cells geometric information is not computed or readily available. Additionally, the geometric center of a Cartesian cell intersected by the boundary may lie outside the computational domain, making it unsuitable as a reconstruction center.

To determine appropriate reconstruction centers for boundary cells, two cases are considered. The first case is when the boundary cell has at least one inside cell as neighbor and the second case is when the boundary cell has no inside cells as neighbor. In the first case, the reconstructed cell will directly interact with the FVM update of the inside cell requiring the reconstruction center to be located inside the computational domain. However, in the second case, the reconstruction center can be placed anywhere within the boundary cell, and the cell can be viewed as a ghost-cell for the boundary cells reconstruction. Consequently, in this case, the reconstruction center is chosen as the geometric center of the boundary cell. For the first case, the reconstruction center is determined using Algorithm 1, which is illustrated in Figure 6.3.

**Algorithm 1** Reconstruction center definition for boundary cells

Define $C$ as the geometric center of the boundary cell.
**if** $C$ is located inside the computational domain **then**
    **return** $C$ as the reconstruction center.
**else**
    Define $P_\infty$ as the center of a face shared with an inside cell.
    Define $d$ as the distance between $C$ and the shared face.
    Define $P_0$ as the nearest point to $P_\infty$ with distance $d$ to the shared face.
    $i \leftarrow 0$
    **while** $P_i$ is not located inside the computational domain **do**
        $i \leftarrow i + 1$
        Define $P_i$ as the middle point between $P_{i-1}$ and $P_\infty$.
    **end while**
    **return** $P_i$ as the reconstruction center.
**end if**

### Reconstruction Methodology

Following the definition of boundary cells, their reconstruction is performed by extending the two-dimensional reconstruction approach presented in previous chapters to three dimensions. For each boundary cell $C$, primitive variables $\phi$ (where $\phi = \rho, u, v, w,$ or $p$) are approximated using a polynomial $\tilde{\phi}_C$ of degree $k$ centered at the reconstruction center $(x_C, y_C, z_C)$ and defined as:

$$\tilde{\phi}_C = a_{\phi,0} + a_{\phi,1}(x - x_C) + a_{\phi,2}(y - y_C) + a_{\phi,3}(z - z_C) + \cdots \tag{6.7}$$

To compute the coefficients $a_{\phi,i}$, the polynomial is evaluated at several points $(x_i, y_i, z_i)$ located near boundary cell $C$ where information about the primitive variable $\phi$ is available. These points form the *reconstruction stencil* and can be of three types: inside cell centers, boundary cell centers (reconstruction centers), and points located on the boundary of the

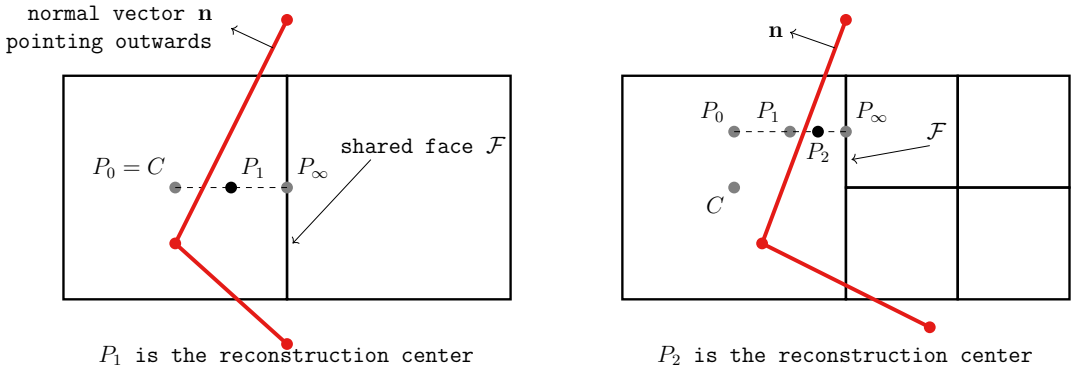



Figure 6.3 Illustration of Algorithm 1 for determining reconstruction centers for boundary cells whose geometric centers lie outside the computational domain. Left: The geometric center $C$ is outside, but the midpoint $P_1$ between $C$ and the shared face center $P_\infty$ is inside. $P_1$ is selected as the reconstruction center. Right: Both $C$ and $P_1$ are outside the computational domain. The midpoint $P_2$ between $P_1$ and $P_\infty$ lies inside and is chosen as the reconstruction center.

discrete topology. Each point in the reconstruction stencil generates a linear equation relating the polynomial coefficients to the primitive variable $\phi$ at the point $(x_i, y_i, z_i)$. These equations form a least-squares system that can be written in matrix form as shown in Equation 6.8.

$$K_{\phi,C}\mathbf{x}_{\phi,C} = \mathbf{b}_{\phi,i} \tag{6.8}$$

where $\mathbf{x}_{\phi,C}$ is the vector of coefficients of the polynomial $\tilde{\phi}_C$, and $K_{\phi,C}$ is a matrix that depends only on the coordinates of the points in the reconstruction stencil. Vector $\mathbf{b}_{\phi,i}$ is a vector that contains the primitive variable information at the points $(x_i, y_i, z_i)$. If a stencil point corresponds to an inside cell center, the primitive variable $\phi$ is known from the FVM update. If the point is located on the boundary of the discrete topology, the primitive variable $\phi$ (Dirichlet) or its gradient $\nabla\phi\cdot\mathbf{n}$ (Neumann) is known from the boundary condition imposed on the discrete topology face. However, if the point is a boundary cell center, no information about the primitive variable $\phi$ is known at the time of reconstruction. Consequently, coupling all boundary cells is necessary to compute the coefficients of the reconstructed polynomial for each boundary cell. If it was guaranteed that the reconstruction stencil of each boundary cell contained no other boundary cells, the reconstruction could be performed explicitly in a single step. However, ensuring this constraint is difficult for general three-dimensional geometries where boundary cells might be densely packed in some regions with insufficient inside cells to fill the reconstruction stencil. Therefore, a global reconstruction is used to reconstruct the

boundary cells, which can be written in matrix form as shown in Equation 6.9.

$$K_\phi \mathbf{x}_\phi = \mathbf{b}_\phi \tag{6.9}$$

where $\mathbf{x}_\phi$ is the vector containing the coefficients $a_{\phi,0}$ of the polynomial $\tilde{\phi}_C$ for all the boundary cells $C$ in the computational domain. Note that coefficient $a_{\phi,0}$ equals the value of primitive variable $\phi$ at the reconstruction center of the boundary cell $C$. Matrix $K_\phi$ is a square matrix that depends only on the coordinates of the points in the reconstruction stencils of all boundary cells, and vector $\mathbf{b}_\phi$ contains known values of the primitive variable information at points in the reconstruction stencils of all boundary cells at reconstruction time. For non-moving geometries, the global matrix $K_\phi$ remains constant throughout the simulation and can be computed and decomposed at the beginning. During each time step, only the right-hand side vector $\mathbf{b}_\phi$ needs to be updated to compute the boundary cells states.

**Reconstruction Stencil**

To conclude this section, details about the reconstruction stencil is provided. For a linear reconstruction ($k = 1$), only four coefficients are needed to compute the polynomial $\tilde{\phi}_C$. Therefore, the six direct neighbors of the boundary cell $C$ are sufficient to form a well-defined local least-squares system of equations. In some cases, some of the direct neighbors of the boundary cell $C$ may lie outside the computational domain, and in that case, these cells are excluded from the reconstruction stencil. Additionally, for a hierarchical mesh, a cell may have four neighbors in a given direction, in which case all neighbors in that direction are included in the reconstruction stencil. The reconstruction stencil is completed by adding boundary points located on the boundary of the discrete topology, selecting the closest points on each triangle that intersect the boundary cell $C$ or its direct neighbors. The obvious choice of selecting triangle centroids is avoided because triangles can be very large, placing centroids far from boundary cell $C$. Conversely, when triangles are small, the number of points in the reconstruction stencil can become excessive, potentially causing the least-squares system to become ill-conditioned. To address this issue, an automatic weighting approach based on the number of points in the reconstruction stencil is implemented, which is nearly equivalent to a sampling of a limited number of boundary points from all admissible boundary points in the stencil. For quadratic reconstruction ($k = 2$), ten coefficients are needed to compute the polynomial $\tilde{\phi}_C$. The reconstruction stencil is formed by combining the linear reconstruction stencil (called the *first layer*) with the direct neighbors of cells in the first layer and their associated boundary points (called the *second layer*). Various user-defined weighting coefficients are applied to adjust the influence of each category of

points (inside cells, boundary cells, boundary points) and each layer (first or second) in the reconstruction stencil. The default weighting coefficients favor points in the first layer due to their proximity to the reconstruction center. Figure 6.4 shows an example of a reconstruction stencil for a quadratic reconstruction.

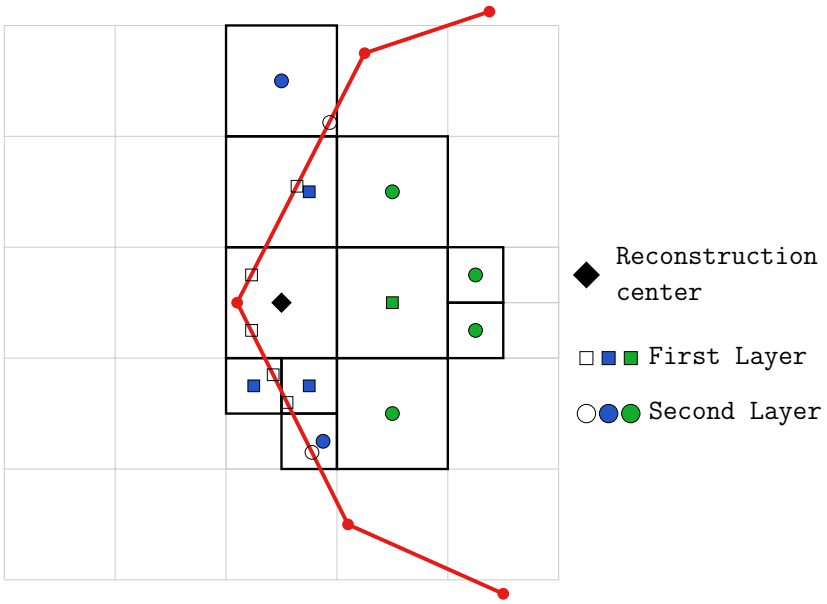



Figure 6.4 Reconstruction stencil for a quadratic reconstruction.

## 6.2 Numerical Results

This section presents four test cases that demonstrate the accuracy and effectiveness of the three-dimensional IBM method for solving the Euler equations. The first test case verifies the global reconstruction approach used in the IBM method and analyzes its convergence properties. This case also examines the convergence properties of the complete Euler solver. The second test case involves a shock tube problem with three types of symmetry (axial, cylindrical, and spherical). Analytical solutions are used to verify the numerical results for the capture of shock waves, contact discontinuities, and expansion waves. The third test case examines the diffraction of a Mach 1.49 shock over a cone, which presents complex flow features. The final test case investigates a complex unsteady flow inside a simplified three-dimensional high-voltage circuit breaker.

### 6.2.1 Reconstruction Verification and Convergence Analysis

This test case is divided into two parts: reconstruction verification and convergence analysis of the IBM Euler solver. In the reconstruction verification part, only the impact (more precisely the accuracy) of the reconstruction of the boundary cells is analyzed while the solution of the inside cells are ignored. The second part examines the convergence properties of the complete IBM solver for the Euler equations, where solutions for both inside and boundary cells are computed.

The test is performed on a simple geometry consisting of a cube with length $L = 1$ centered at the origin. The cube is rotated by an angle $\theta = 9°$ around the $z$ and $y$-axis successively. These rotations ensure that the IBM grid lines are not aligned with the geometry, providing a more generic test condition. Table 6.1 presents the properties of the meshes used in this test case. Each mesh is generated using a uniform Cartesian grid with the specified grid size. Additionally, local grid refinement is performed on cells intersected by the $+z$ boundary face (1 refinement level) and by a virtual sphere of radius $R = 0.2$ centered at the origin (2 refinement levels). This local grid refinement ensures that cells in both the boundary region and the interior of the cube have multiple refinement levels, allowing the inclusion of the hierarchical aspect of the mesh in the verification and convergence analysis. Figure 6.5 shows a cut view of the second mesh of the test case.

Table 6.1 Properties of the meshes used for the reconstruction verification and convergence analysis.

| Mesh | Grid Size | Number of Solved Cells | Number of Inside Cells | Number of Boundary Cells | Inside Cells Average Size | Boundary Cells Average Size |
|---|---|---|---|---|---|---|
| 1 | $20 \times 20 \times 20$ | 12 899 | 10 264 | 2 635 | 4.256E-2 | 5.466E-2 |
| 2 | $40 \times 40 \times 40$ | 65 669 | 55 413 | 10 256 | 2.529E-2 | 2.761E-2 |
| 3 | $80 \times 80 \times 80$ | 378 225 | 337 804 | 40 421 | 1.410E-2 | 1.386E-2 |
| 4 | $160 \times 160 \times 160$ | 2 442 293 | 2 281 108 | 161 185 | 7.528E-3 | 6.949E-3 |

#### Reconstruction Verification

The reconstruction verification is performed in three steps. First, the solution at the inside cells is initialized with a given function $\phi_{\text{exact}}$. Boundary conditions $\text{BC}_\phi$ compatible with the function $\phi_{\text{exact}}$ are imposed on the boundary of the cube. For example, with Neumann boundary conditions, the boundary values are given by $\text{BC}_\phi = \nabla\phi_{\text{exact}} \cdot \mathbf{n}$. The second step consists of reconstructing the boundary cells using the global reconstruction method presented in the previous section. Finally, the reconstructed values are compared to the

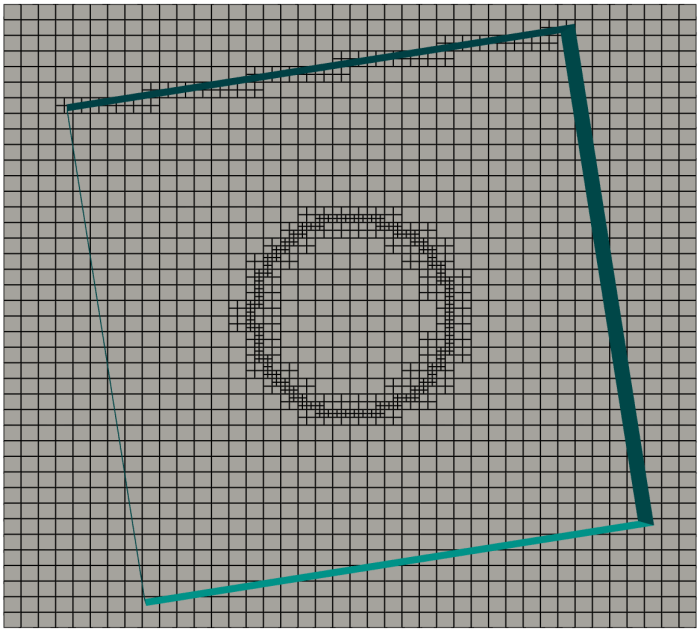


Figure 6.5 Illustration of the second mesh used for the reconstruction verification.

exact values of the function $\phi_{\text{exact}}$ at the reconstruction centers of the boundary cells.

Three functions given by Equation 6.10 corresponding to a linear, quadratic and sinusoidal (called MMS as *Method of Manufactured Solutions* [111, 113]) functions are employed for the reconstruction verification.

$$\begin{aligned}
\phi_{\text{LIN}}(x,y,z) &= a_0 + a_1 x + a_2 y + a_3 z, \\
\phi_{\text{QUAD}}(x,y,z) &= b_0 + b_1 x + b_2 y + b_3 z + b_4 x^2 + b_5 y^2 + b_6 z^2 + b_7 xy + b_8 xz + b_9 yz, \\
\phi_{\text{MMS}}(x,y,z) &= c_0 + c_1 \cos(\pi x) + c_2 \sin(\pi y) + c_3 \cos(\pi z) + c_4 \sin(\pi x)\, cos(\pi y) \sin(\pi z)
\end{aligned} \tag{6.10}$$

where $a_i$, $b_i$ and $c_i$ are non-zero coefficients chosen between 1/2 and 2 in absolute value. For the boundary conditions, two cases are considered: a Dirichlet case where all boundary conditions are of Dirichlet type and a Neumann case where all boundary conditions are of Neumann type. Table 6.2 shows the reconstruction errors using the $L_\infty$ norm for the three functions and for both boundary condition types. The second mesh from Table 6.1 was used for the reconstruction verification. As expected, the linear reconstruction reproduces the exact values of the linear function at machine precision for both Dirichlet and Neumann boundary conditions. Similarly, the quadratic reconstruction reproduces the exact values of

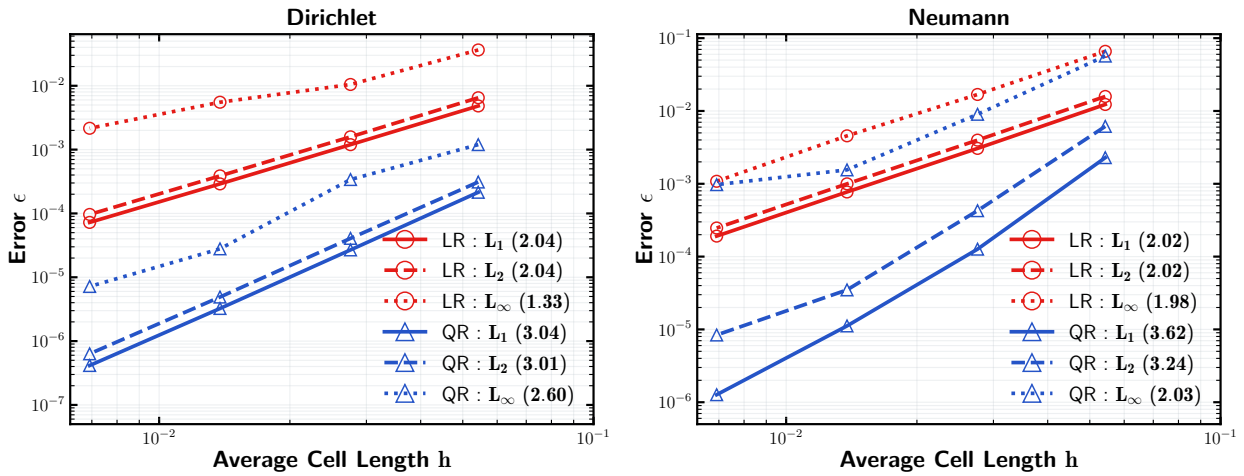
Table 6.2 Reconstruction errors using the $L_\infty$ norm for the three functions and for the two cases of boundary conditions.

| | **Linear Reconstruction** | | **Quadratic Reconstruction** | |
|---|---|---|---|---|
| | Dirichlet | Neumann | Dirichlet | Neumann |
| Linear $\phi_{\text{lin}}$ | 3.1086e−15 | 3.9293e−12 | 4.4409e−15 | 4.3321e−11 |
| Quadratic $\phi_{\text{quad}}$ | 1.6158e−02 | 2.5808e−03 | 5.1070e−15 | 1.4034e−10 |
| MMS $\phi_{\text{MMS}}$ | 1.0479e−02 | 1.6851e−02 | 3.3968e−04 | 9.0046e−03 |

both linear and quadratic functions at machine precision for both boundary condition types. For the MMS function, neither the linear nor the quadratic reconstruction can reproduce the exact values at machine precision. In the Euler solver, this means that the reconstruction process will introduce some numerical errors even if the solution in the inside cells were exact. Nevertheless, these reconstruction errors are very small (smaller for the quadratic reconstruction and for Dirichlet boundary conditions) and will decrease with grid refinement. This is confirmed by the convergence analysis shown in Figure 6.6, which illustrates the convergence of the $L_1$, $L_2$, and $L_\infty$ norms of the reconstruction errors associated with the MMS function with respect to the average grid size $h$. The reconstruction errors converge at a rate of approximately second order for linear reconstruction and third order for quadratic reconstruction, regardless of the boundary condition type. However, the errors are slightly smaller for Dirichlet boundary conditions than for Neumann boundary conditions.

### Convergence Analysis of the IBM Euler Solver

For the convergence analysis of the IBM Euler solver, the whole iteration process is performed and the errors for the inside and boundary cells are computed. The procedure is similar to the one used for the reconstruction verification. Five MMS (sinusoidal) functions similar to the previous MMS function and representing the density, the three components of the velocity and the pressure respectively are used to initialize the solution in the inside cells and in the boundary cells. The coefficients of the functions are chosen such that the flow is subsonic throughout the computational domain. Next, the IBM method is used to solve the Euler equations in the inside cells and the boundary cells with Dirichlet boundary conditions compatible with the initial conditions. The solver uses source terms computed from the MMS functions such that the MMS functions are the exact solution of the Euler equations and limiters are deactivated for this test case since the solution is smooth. Finally, the errors are computed using the $L_1$ and $L_\infty$ norms for the inside and boundary cells when the




(a) Dirichlet (b) Neumann

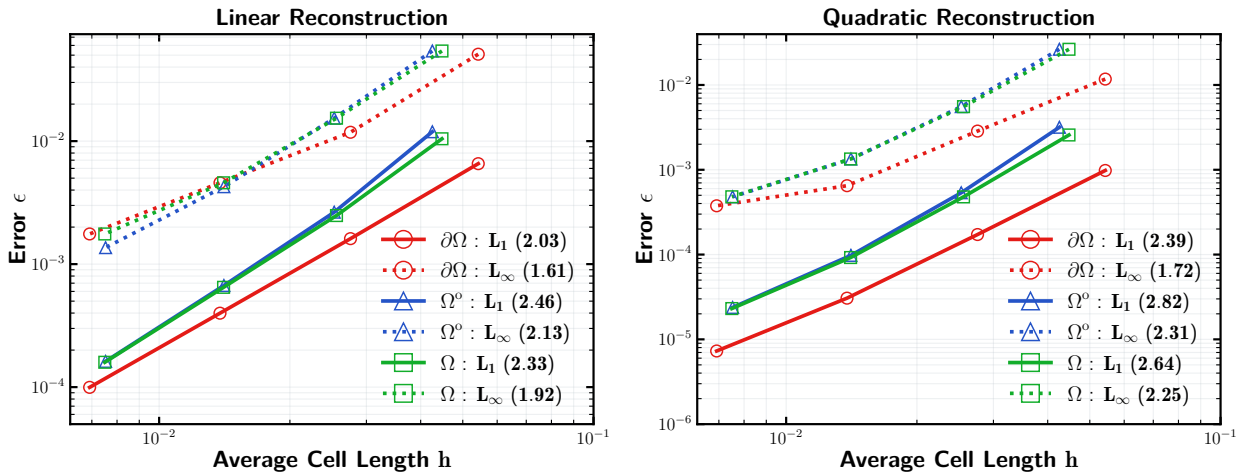
Figure 6.6 Convergence of the reconstruction errors using the $L_1$, $L_2$ and $L_\infty$ norms with respect to the average grid size $h$. LR = Linear Reconstruction, QR = Quadratic Reconstruction.

residuals of the Euler equations reach machine precision. Figure 6.7 shows the convergence of the errors using both linear and quadratic reconstruction. The errors are separated by regions: $\partial\Omega$ for boundary cells and $\Omega^o$ for inside cells and $\Omega$ for the entire computational domain. The errors converge at a high order (greater than 2 on average) for all three domains with the errors for the quadratic reconstruction being an order of magnitude smaller than the errors for the linear reconstruction.

This test case demonstrates that both the reconstruction process and the IBM Euler solver function correctly and converge at high order. The next test case examines a physical problem involving various features of compressible flows.

### 6.2.2 Shock Tube Problem

The shock tube problem is solved for three geometries: cube (rectangular cuboid), cylinder and sphere as shown in Figure 6.8. Table 6.3 shows the mesh and topology properties for each geometry. Figure 6.9 displays density contour plots for the three geometries at the final time $t = 0.25$. The numerical results seem to be in good agreement with the expected solutions with each geometry clearly showing its characteristic symmetry pattern. From the left state to the right state, rarefaction waves followed by a contact discontinuity and a shock wave are captured in each geometry. The amplitude and location of these waves vary slightly between geometries. For instance, the shock wave in the cubic geometry is slightly stronger




(a) Linear Reconstruction

(b) Quadratic Reconstruction

Figure 6.7 Convergence of the three-dimensional IBM Euler solver using manufactured solutions. The errors are separated by regions: $\partial\Omega$ for the boundary cells and $\Omega^o$ for the inside cells and $\Omega$ for the whole computational domain.

Table 6.3 Parameters of the shock tube problem for the three geometries.

| **Geometry** | **Grid Size** | **Number of Solved Cells** | **Number of Boundary Cells** | **Average Cell Size** | **Number of Triangles** |
|---|---|---|---|---|---|
| Cube | $320 \times 128 \times 64$ | 1 173 360 | 86 672 | 3.2814e−3 | 16 |
| Cylinder | $240 \times 240 \times 48$ | 1 774 124 | 115 949 | 7.1888e−3 | 175 536 |
| Sphere | $160 \times 160 \times 160$ | 1 781 243 | 104 508 | 1.3432e−2 | 298 802 |

than that in the spherical geometry.

Fortunately, analytical solutions are available for the shock tube problem, providing a means to verify the numerical results. Due to the problem's symmetry, the three-dimensional shock tube problem can be reduced to a one-dimensional problem [109] given by Equation 6.11.

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial \tilde{x}} = \mathbf{Q}(\mathbf{U}) \tag{6.11}$$

with

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho\tilde{u} \\ \rho E \end{pmatrix}, \quad \mathbf{F}(\mathbf{U}) = \begin{pmatrix} \rho\tilde{u} \\ \rho\tilde{u}^2 + p \\ (\rho E + p)\tilde{u} \end{pmatrix}, \quad \mathbf{Q}(\mathbf{U}) = -(\alpha - 1) \begin{pmatrix} \rho\tilde{u}/\tilde{x} \\ \rho\tilde{u}^2/\tilde{x} \\ (\rho E + p)\tilde{u}/\tilde{x} \end{pmatrix} \tag{6.12}$$

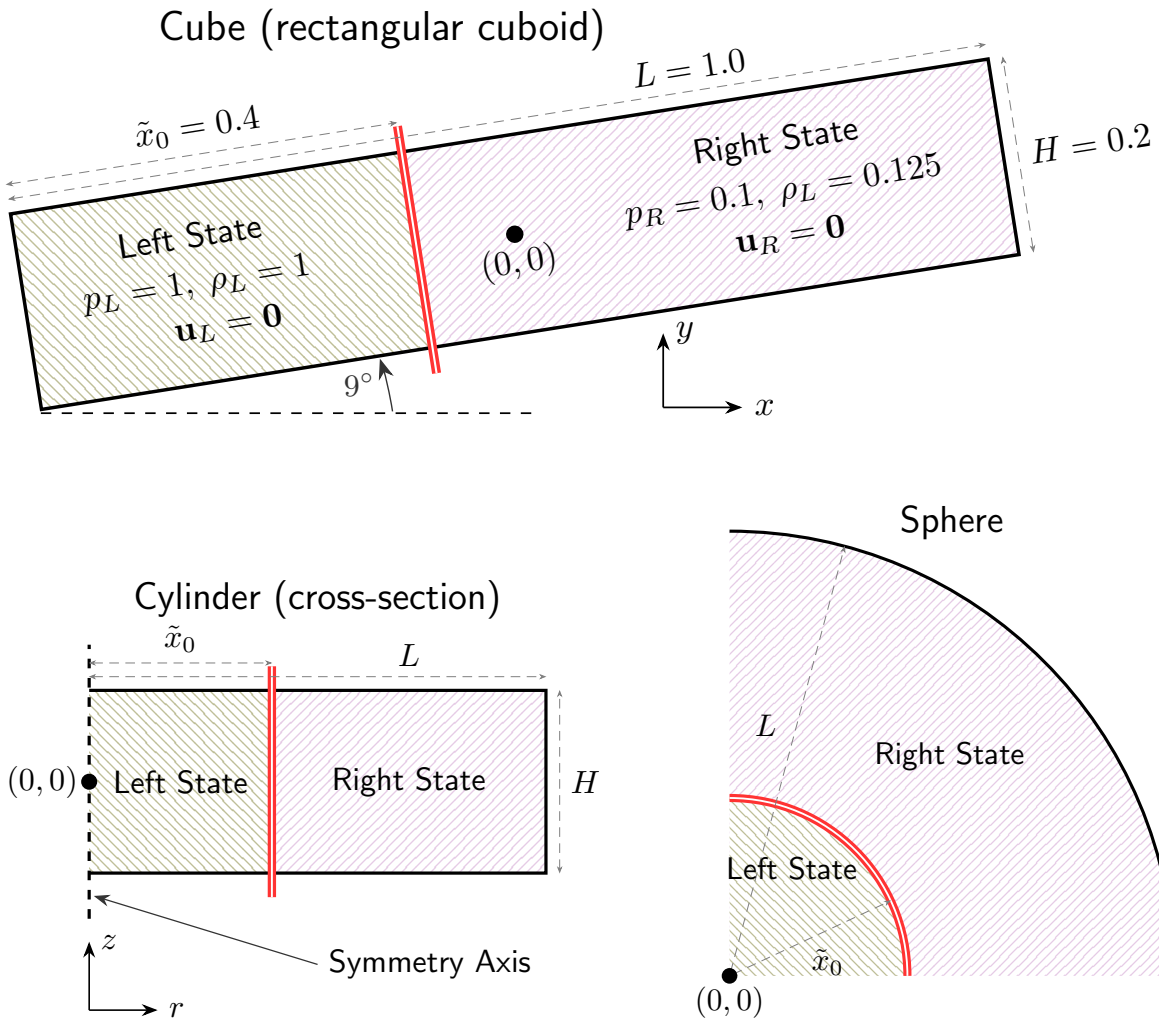
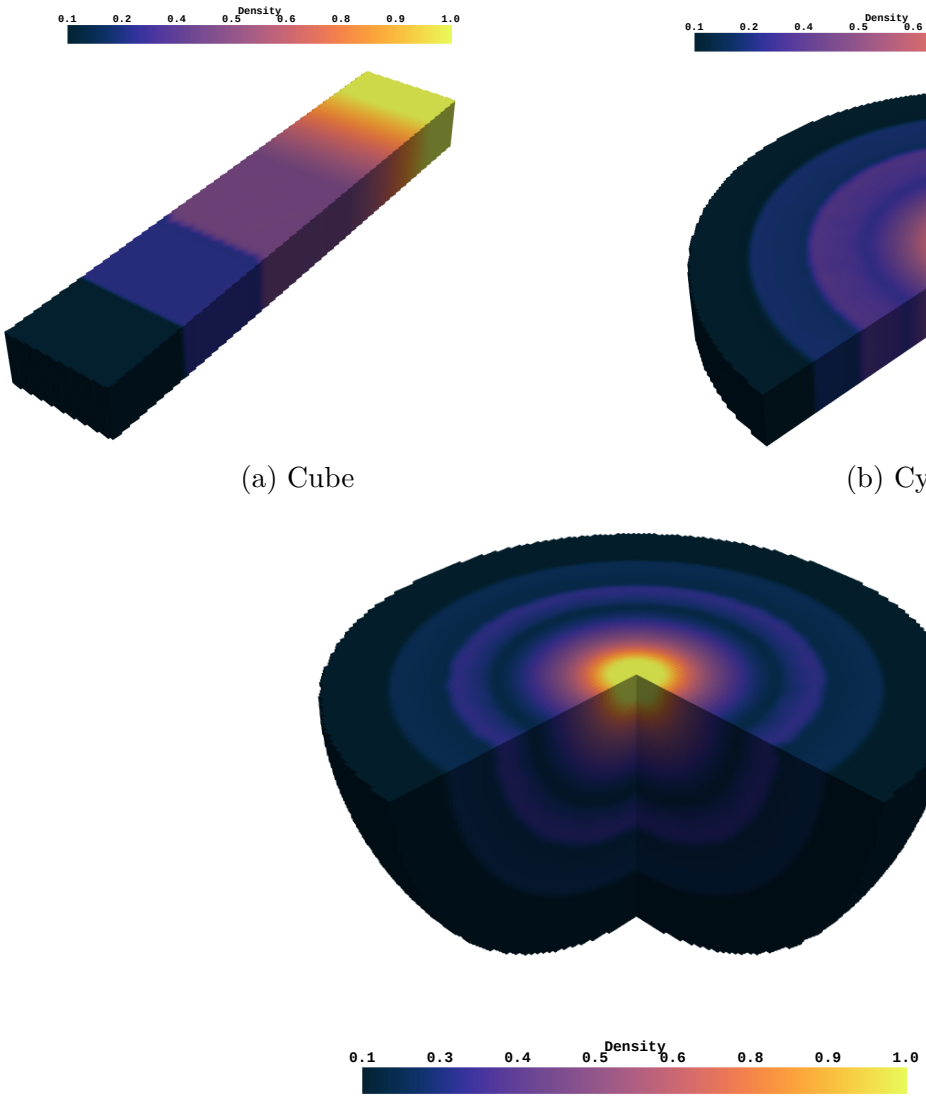



Figure 6.8 Geometries used for the shock tube problem.

In these equations, $\tilde{x}$ and $\tilde{u}$ are the coordinates and velocity in the symmetry direction, respectively. Parameter $\alpha$ represents the symmetry coefficient defined as $\alpha = 1$ for the cube, $\alpha = 2$ for the cylinder, and $\alpha = 3$ for the sphere. For the cubic geometry, the analytical solution is obtained from an exact Riemann solver [109]. For the cylinder and sphere geometries, the Random Choice Method (RCM) [131–133] in conjunction with a splitting of the source terms is employed to obtain the analytical solution. While the RCM method does not provide an exact solution, it produces results very close to the exact solution. The method captures discontinuities sharply without smearing, and although it may slightly misplace discontinuities due to its non-conservative nature, these errors become negligible when using a sufficiently fine mesh [109]. Figure 6.10 shows the analytical solutions for all three geometries.




(a) Cube

(b) Cylinder

(c) Sphere

Figure 6.9 Contour plots of the density for the three geometries at the final time $t = 0.25$.

The solutions exhibit similar behaviors with some differences in the amplitude and location of the shock waves and contact discontinuities. The curvature of the regions between waves varies significantly across the geometries. Some artifacts appear in the analytical solutions for the cylindrical and spherical cases, manifesting as small wiggles in the smooth regions such as the rarefaction waves. These artifacts result from the inherent randomness of the RCM method [133].

Figure 6.11 compares the numerical and analytical solutions for all three geometries using

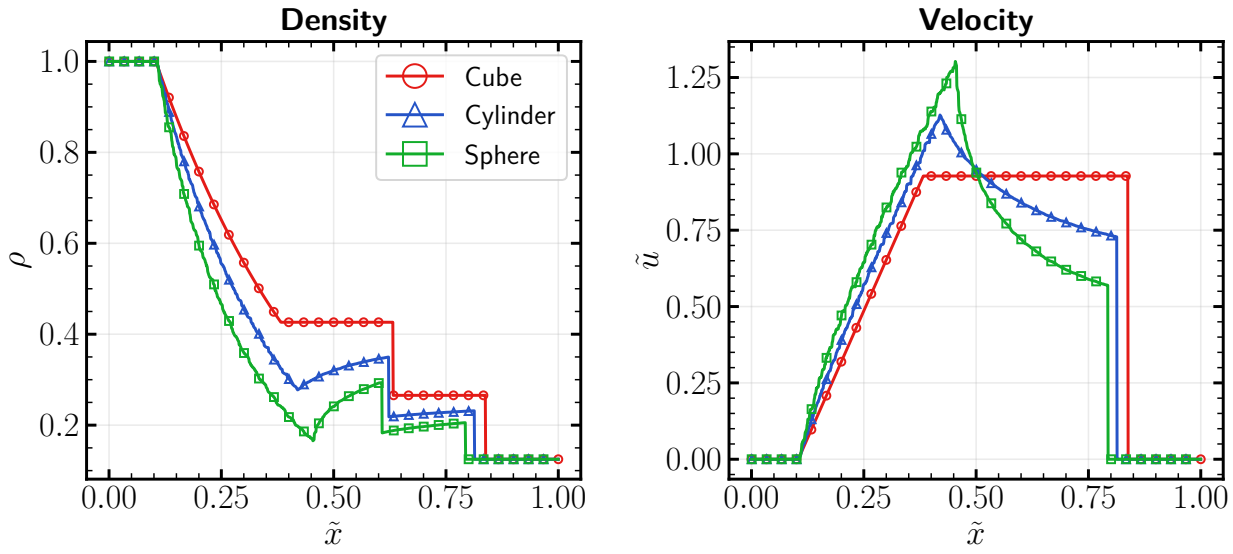
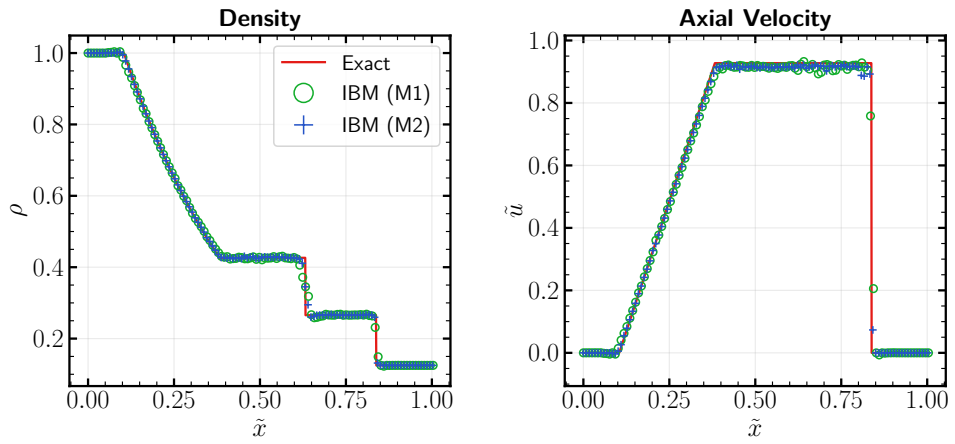


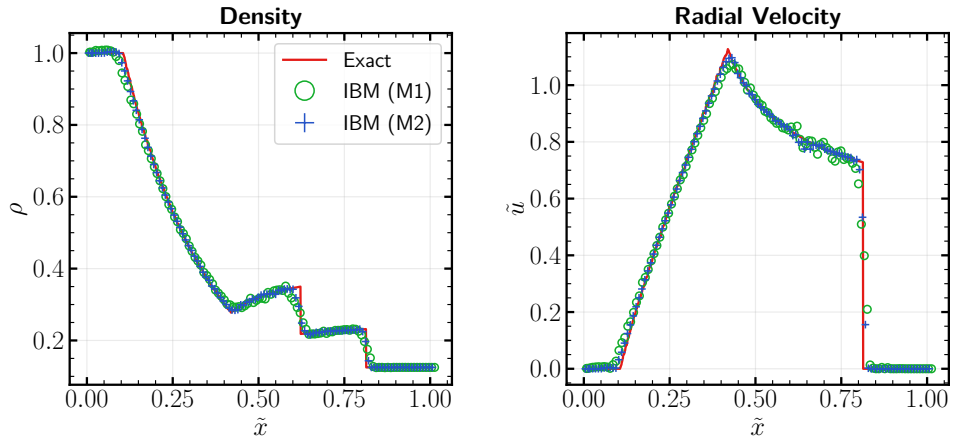

Figure 6.10 Analytical solution for the three geometries at the final time $t = 0.25$.

two different meshes: M1 and M2. Mesh M2, the finer mesh, corresponds to the grid used for the contour plots in Figure 6.9. Mesh M1 is a coarser version with half the number of cells in each direction. The numerical solutions match the analytical solutions remarkably well for all geometries, even on the coarser mesh M1. The simulations capture the shock waves within a span of just a few cells. The largest discrepancies appear in the contact discontinuity regions, where the numerical solutions show slight smearing. Differences between the two meshes are mainly visible at the interfaces between flow regions, with the finer mesh M2 capturing flow features in greater detail than the coarser mesh M1. Some multi-dimensional effects are also observed in the numerical solutions, particularly in the region between the shock wave and the contact discontinuity. These effects are attributed to the non-alignment of the Cartesian grid cells with the geometries symmetry lines and are accentuated by the boundary reconstruction process.

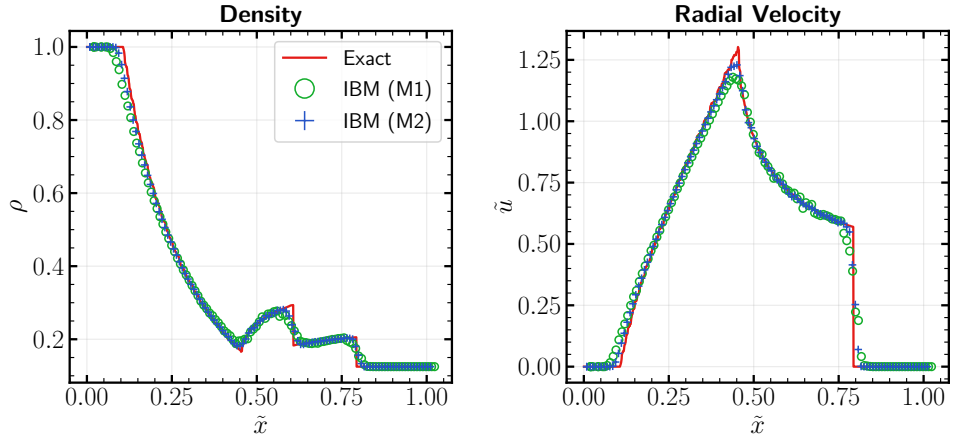
This test case successfully verifies the Euler solver for problems featuring various flow structures with known analytical solutions. The results demonstrate that the IBM solver produces accurate numerical solutions that compare well with analytical predictions. In the following section, a test case comporting more complex flow features is presented to further verify the IBM method.




(a) Cube




(b) Cylinder




(c) Sphere

Figure 6.11 Comparison between analytical and numerical solutions for the three geometries. The solid lines represent the analytical solution, while the symbols represent the numerical solution evaluated at the mesh cell centers.

### 6.2.3 Shock Diffraction over a Cone

Figure 6.12 shows the geometry of the test case featuring a cone with a semi-apex angle of $\Theta_w = 26.6°$ and a shock wave moving at a Mach number of $M_s = 1.49$ diffracting over the cone. Post-shock conditions are imposed at boundary $\Gamma_1$, while non-penetrating boundary conditions are imposed at boundary $\Gamma_2$. For all remaining boundaries, extrapolation boundary conditions (Neumann type with zero gradient for all primitive variables) are applied.

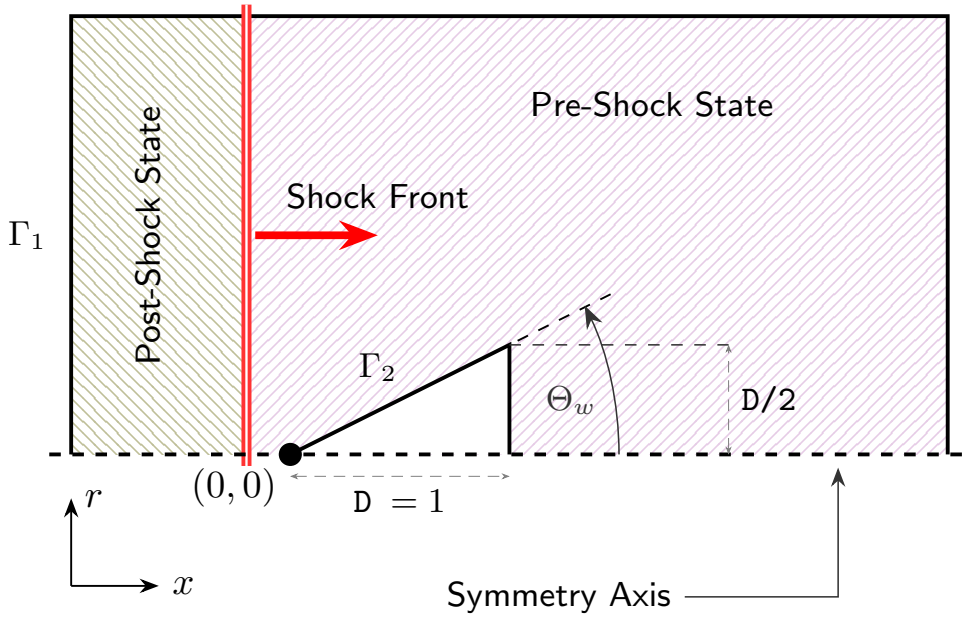


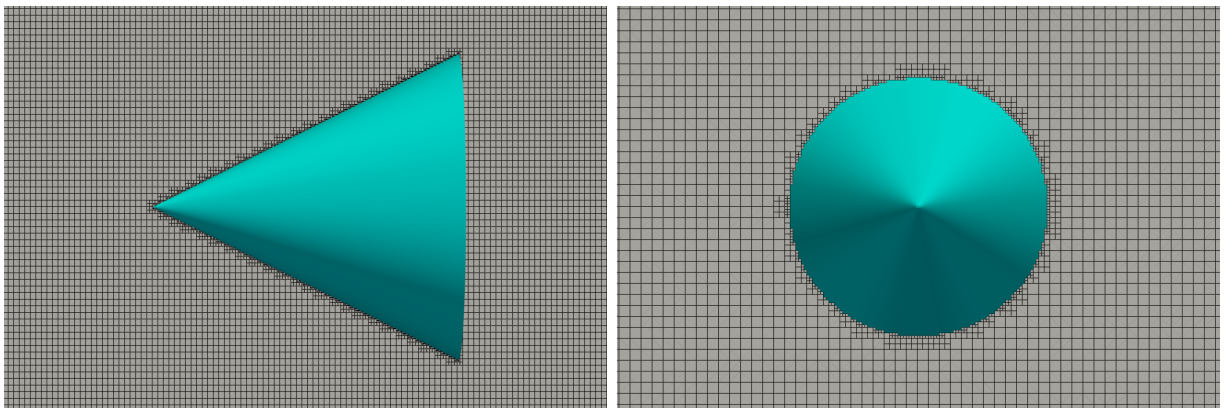

Figure 6.12 Geometry of the shock diffraction over a cone.

Figure 6.13 shows the mesh used for this test case, and Table 6.4 summarizes the mesh properties.

Table 6.4 Properties of the mesh used for the shock diffraction over a cone.

| Grid Size | Number of Solved Cells | Number of Boundary Cells | Average Cell Size | Average Boundary Cell Size | Number of Triangles |
|---|---|---|---|---|---|
| 240 × 200 × 200 | 7 780 327 | 363 820 | 2.0238e−2 | 1.7600e−2 | 236 356 |

Figures 6.14 and 6.15 display Mach number contour plots obtained from the numerical solution at various times. When the shock wave encounters the cone, it is reflected and a Mach stem forms. Due to the conical geometry (in contrast to a spherical one), the reflection is



(a) $xz$-plane (b) $yz$-plane

Figure 6.13 Mesh used for the shock diffraction over a cone. The apparent discrepancy in mesh density results from the different zoom levels in the two subfigures.

relatively weak and the Mach stem remains small. As the incident shock passes behind the cone, the flow accelerates, and later the upper and lower portions of the flow collide, which generates a new series of shock reflections and Mach stems. These complex interactions between different flow structures produce a rich flow field with diverse features.

To validate the numerical results, the Mach number at the cone surface is computed and compared with the experimental results from [134]. Table 6.5 presents this comparison, showing good agreement between numerical and experimental values. Some of the difference between these results can be attributed to the omission of viscous effects in the numerical model.

This test case demonstrates that the IBM method can be employed to solve complex flow phenomena. A test case using a more sophisticated geometry is presented next to evaluate the performance of the IBM method under more realistic conditions.

Table 6.5 Comparison between the numerical and experimental results for the Mach number at the cone surface.

| **Semi-Apex Angle $\Theta_w$** | **Numerical Results** | **Experimental Results** |
|---|---|---|
| 26.6° | 1.68 | 1.73 |

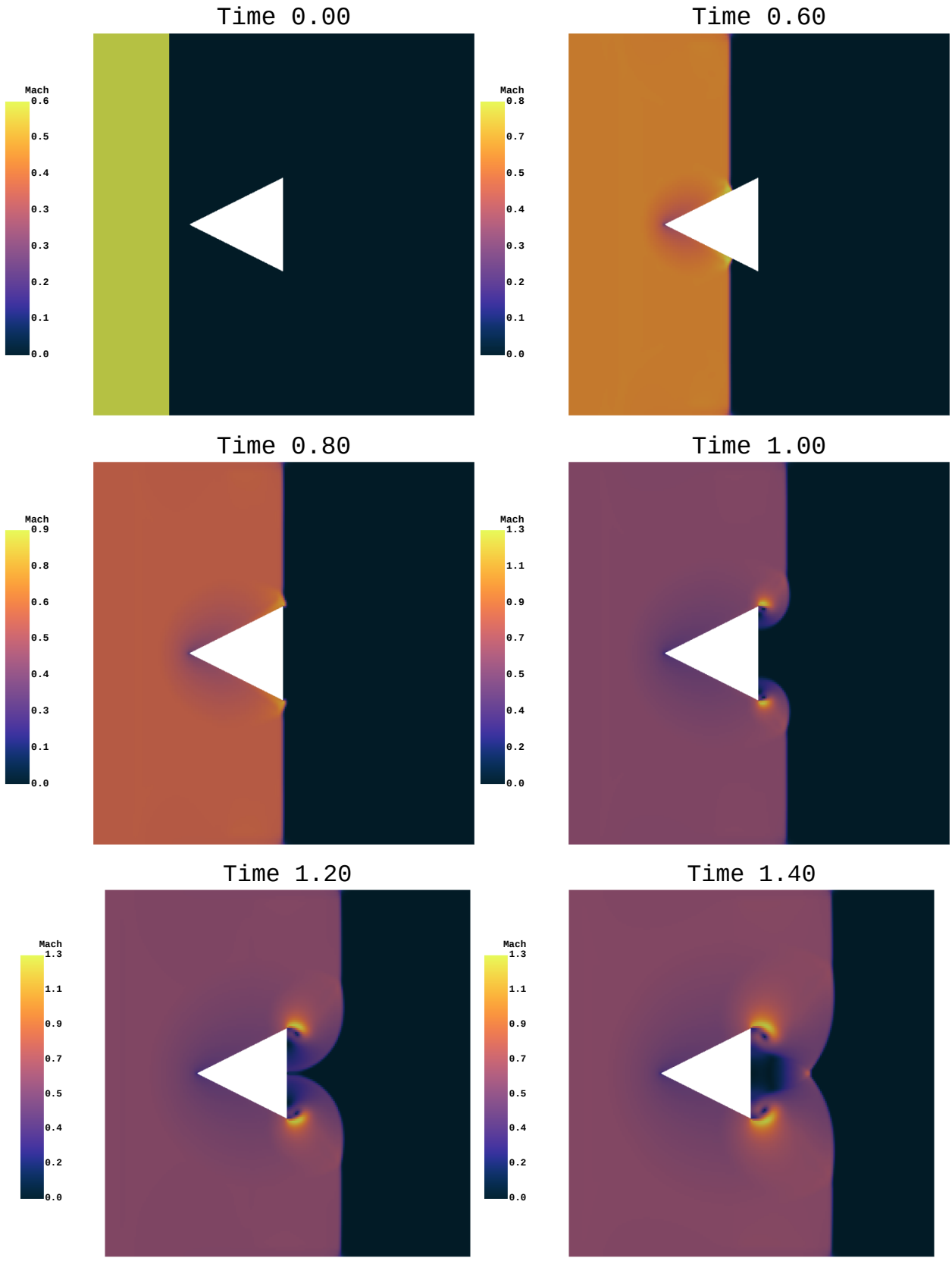



Figure 6.14 Contour plots of the mach number at various times for the shock diffraction over a cone (1/2).

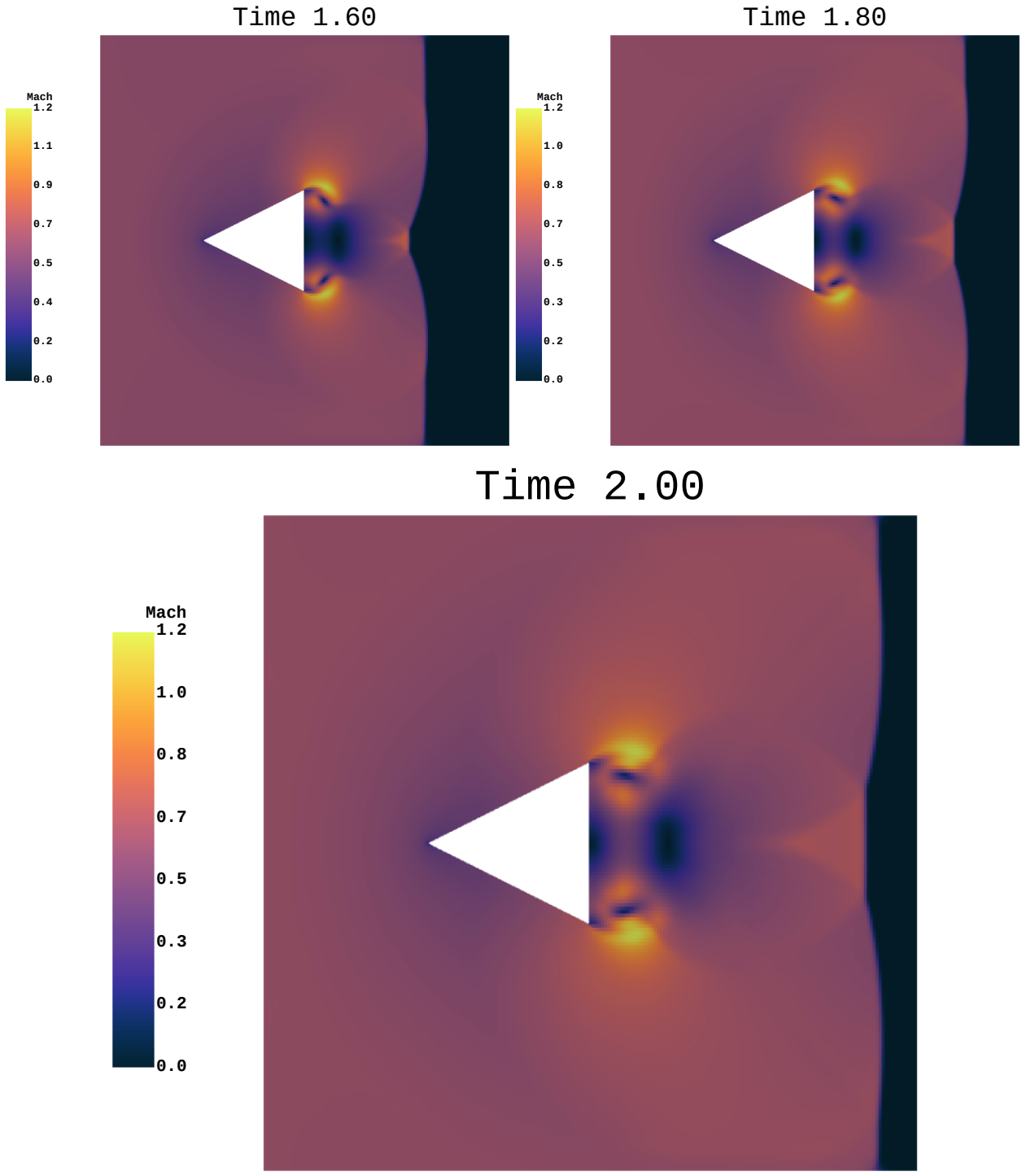



Figure 6.15 Contour plots of the mach number at various times for the shock diffraction over a cone (2/2).

### 6.2.4 Unsteady Flow inside a Simplified Three-Dimensional Circuit Breaker

The final test case examines an unsteady flow inside a simplified three-dimensional high-voltage circuit breaker.

#### Test Case Setup

The problem configuration is sketched in Figure 6.16, where compressed air is initially contained within the expansion volume of the circuit breaker. At time $t = 0$, the air is suddenly released, allowing it to expand freely through the thermal canal and into the arc region.

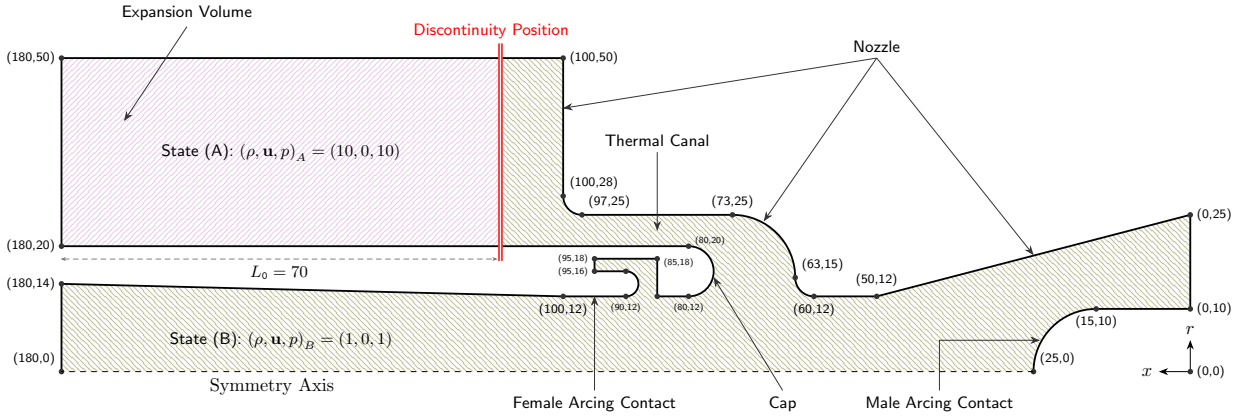



Figure 6.16 Geometry of the unsteady flow inside a simplified 3D high-voltage circuit breaker.

Figure 6.17 shows the discrete topology used for this simulation, while Figure 6.18 displays the computational mesh. Table 6.6 summarizes the mesh and topology properties. During mesh generation, all cells intersected by the topology were refined (with two refinement levels applied to cells near the cap or canal, and one level to all other intersected cells).

Table 6.6 Discrete topology and properties of the mesh used for the simplified 3D high-voltage circuit breaker.

| Grid Size | Number of Solved Cells | Number of Boundary Cells | Average Cell Size | Average Boundary Cell Size | Number of Triangles |
|---|---|---|---|---|---|
| $504 \times 280 \times 280$ | 20 223 329 | 4 352 560 | 3.2402e−1 | 1.5679e−1 | 1 089 364 |

#### Mach Number Contour Comparison

Figures 6.19, 6.20, and 6.21 present Mach number contours at different times. Each figure shows the IBM solution (top) compared with results obtained using COMSOL Multi-

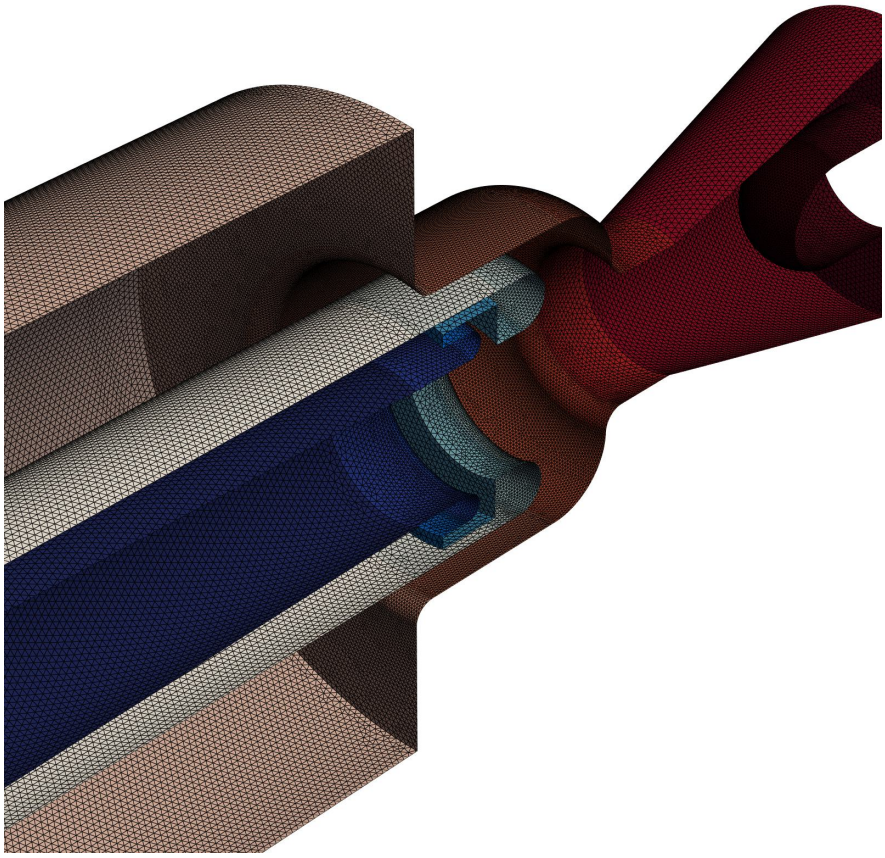
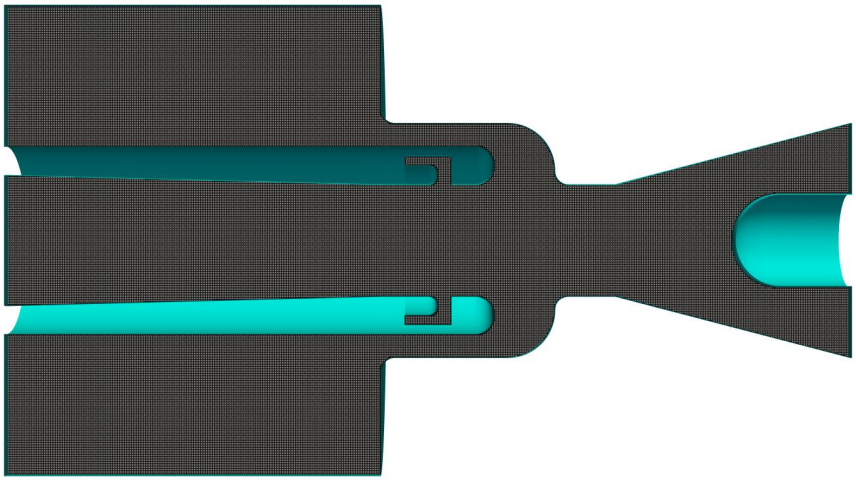


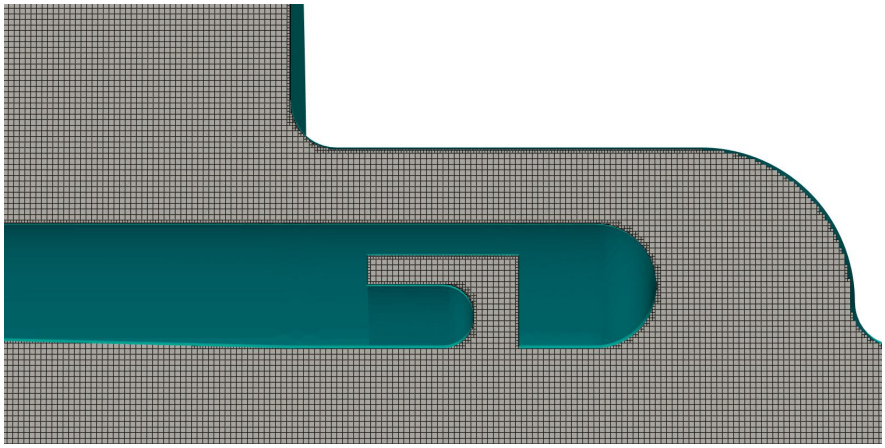
Figure 6.17 Discrete topology of the mesh used for the unsteady flow inside a simplified 3D high-voltage circuit breaker.

physics [135] (bottom). The COMSOL solution was generated using a fine mesh (average element size of $2.7368 \times 10^{-1}$) with an axisymmetric two-dimensional model and first-order Discontinuous Galerkin discretization.

Initially, the flow within the circuit breaker is at rest, with a Mach number at zero everywhere. When the air is released, the flow behavior resembles that of a shock tube problem, with a rarefaction wave propagating into the expansion volume and a shock wave moving toward the canal. Upon reaching the canal, a significant portion of the shock reflects from the vertical nozzle wall, while the remainder transmits into the canal. Within the canal, the flow undergoes several reflections and rapidly accelerates, reaching Mach 3.8 at the cap tip. The flow then enters the arc region, where it collides with itself, generating multiple reflections and shocks. Subsequently, the flow becomes increasingly complex, developing vortices, shocks, and reflections throughout the circuit breaker. Throughout the simulation, the flow remains unsteady with continuously evolving features. Both IBM and COMSOL



(a) $xz$-plane



(b) Zoom $xz$-plane

Figure 6.18 Mesh used for the unsteady flow inside a simplified 3D high-voltage circuit breaker.

solutions demonstrate good agreement with each other, with the IBM method accurately capturing flow features despite using a less refined mesh than COMSOL. The primary difference between the solutions appears as numerical artifacts in the symmetry plane of the circuit breaker in the IBM solution. This discrepancy occurs because the three-dimensional IBM grid lines do not align with the symmetry plane, whereas COMSOL enforces symmetry conditions directly at the discretization level.

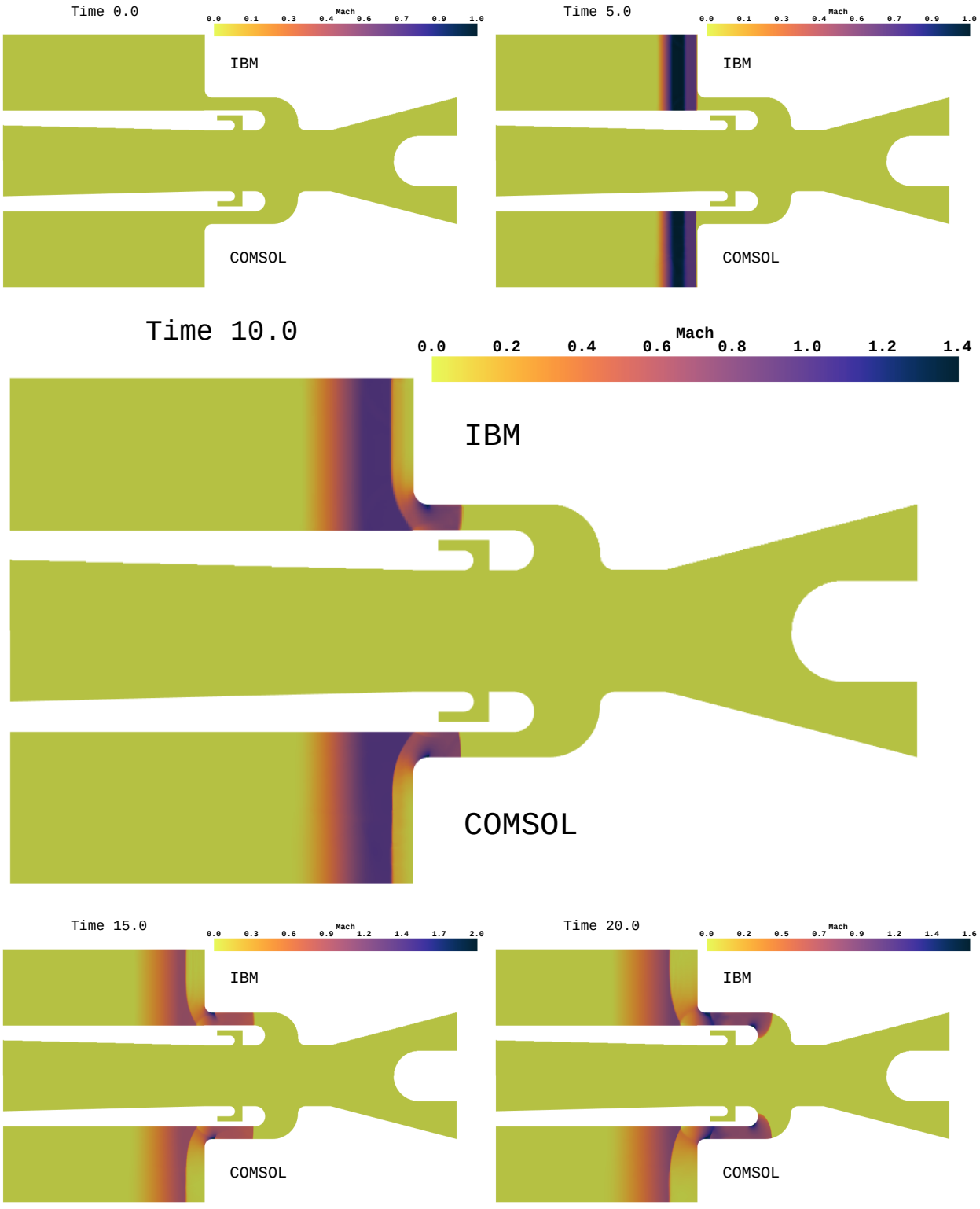



Figure 6.19 Mach number contour at various times for the unsteady flow inside a simplified 3D high-voltage circuit breaker (1/3).

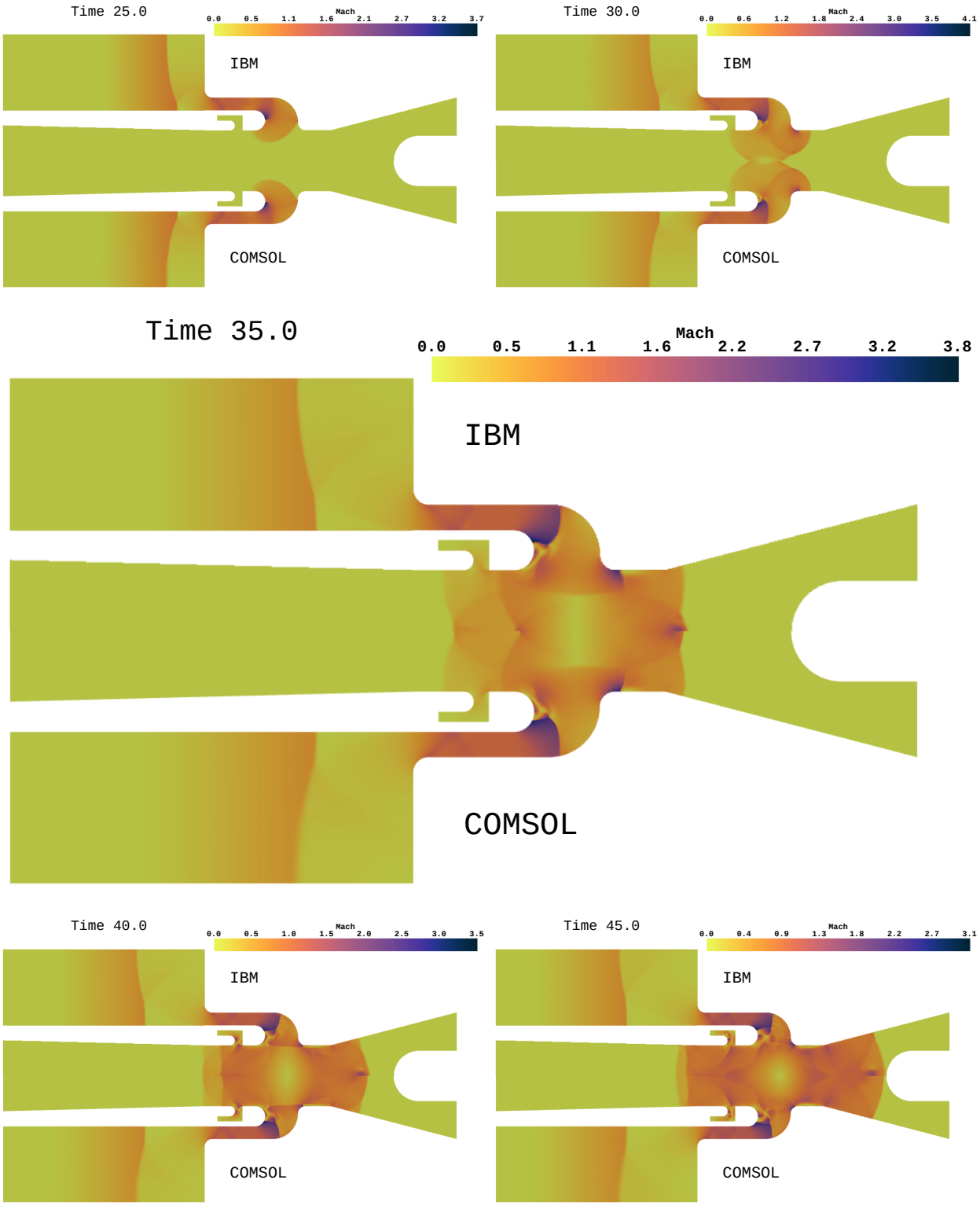



Figure 6.20 Mach number contour at various times for the unsteady flow inside a simplified 3D high-voltage circuit breaker (2/3).

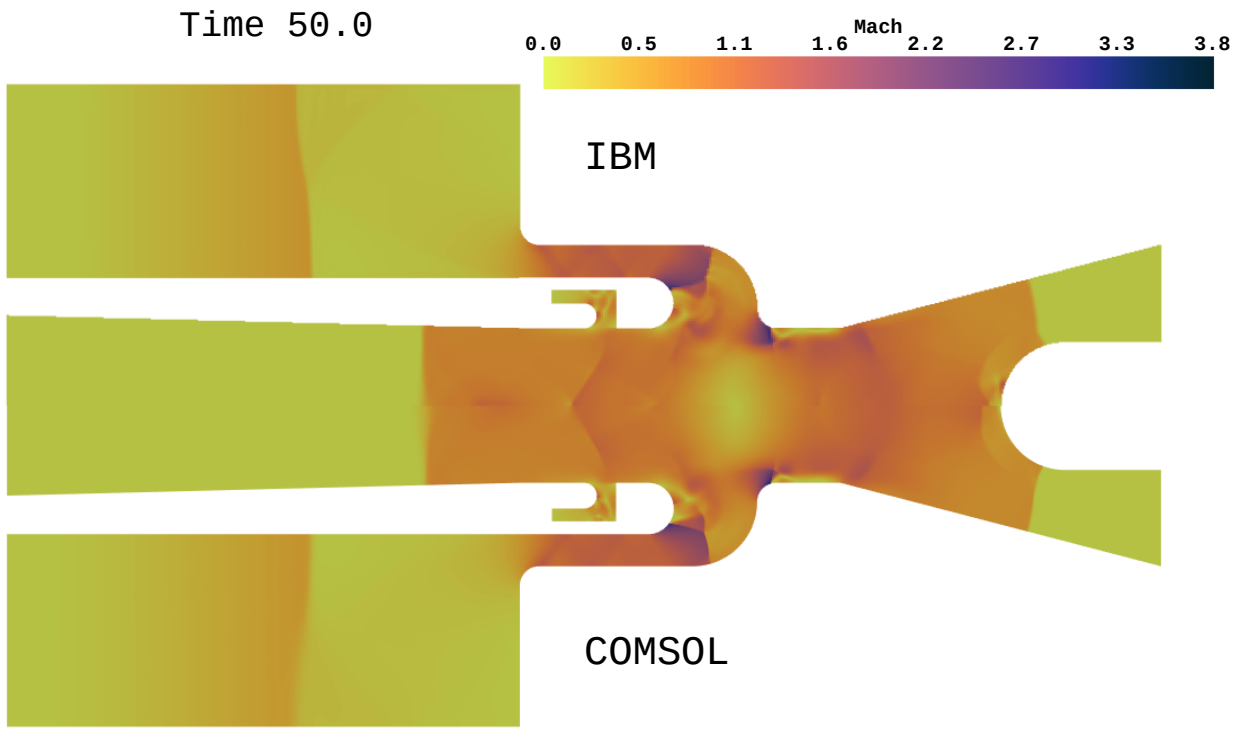



Figure 6.21 Mach number contour at various times for the unsteady flow inside a simplified 3D high-voltage circuit breaker (3/3).

### Mass Flux Comparison

For a more quantitative comparison between the two solutions, the mass flux given by Equation 6.13 is computed across an annular section located midway in the canal ($\Gamma = \{(x, r, \theta) \mid x = 90,\ 20 \leq r \leq 25\}$).

$$Q = \int_{\Gamma} \rho \mathbf{u} \cdot \mathbf{n}_{\Gamma} d\Gamma \tag{6.13}$$

where $\mathbf{n}_{\Gamma}$ is the normal vector to surface $\Gamma$ pointing in the global flow direction. Figure 6.22 compares the mass fluxes computed from both IBM and COMSOL solutions. The results show excellent agreement throughout most of the simulation. Initially, the flux remains zero while the flow is at rest within the canal. When fluid begins flowing from the expansion volume into the canal, the mass flux increases rapidly, eventually reaching a pseudo-steady state. Only during the unsteady phase do the solutions exhibit slight differences, but they converge to the same values after this period.

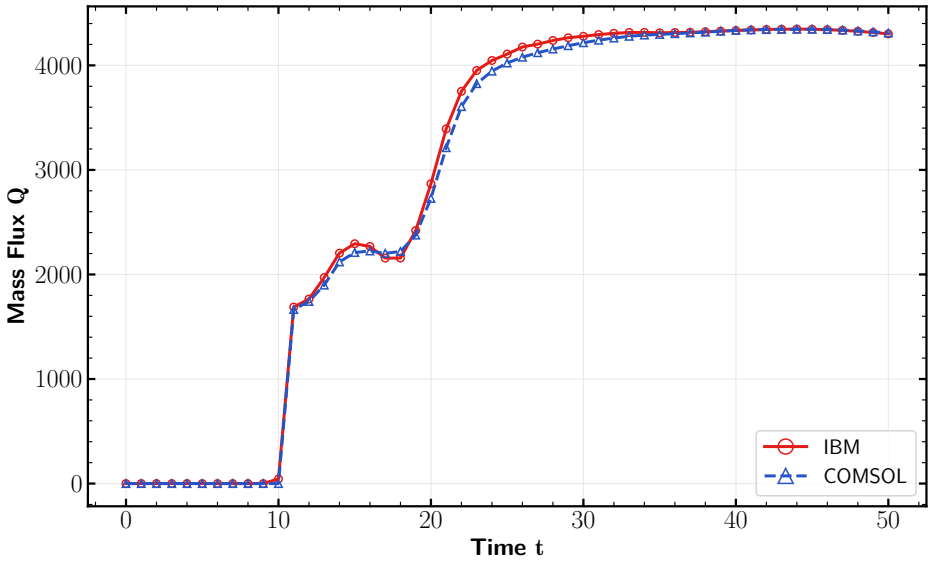



Figure 6.22 Mass fluxes computed using the IBM and COMSOL solutions.

### Conservation Properties of the IBM Method

To conclude this test case, the conservation properties of the IBM method is examined. Since solid walls bound the entire circuit breaker, the total mass $T_M(t)$ and energy $T_E(t)$ should remain constant throughout the simulation. Since the IBM method does not explicitly define the computational domain within the solver, an approximate value of the total mass and energy given by Equation 6.14 is used. In these calculations, the volume of the boundary cells are taken to be equal to that of their uncut counterparts.

$$T_M^n = \int_\Omega \rho^n d\Omega \approx \sum_{i=1}^{N} \rho_i^n \Delta V_i, \qquad T_E^n = \int_\Omega \rho^n E^n d\Omega \approx \sum_{i=1}^{N} \rho_i^n E_i^n \Delta V_i \tag{6.14}$$

Using the total mass and energy, two conservation errors metrics $\delta\epsilon$ (instantaneous error) and $\Delta\epsilon$ (global error) are defined for each quantity as follows:

$$\delta\epsilon^n = \frac{|T^{n+1} - T^n|}{T^n}, \qquad \Delta\epsilon^n = \frac{|T^{n+1} - T^0|}{T^0} \tag{6.15}$$

where $T$ represents either mass or energy. The instantaneous error quantifies conservation errors between consecutive time steps, while the global error measures the cumulative conservation drift from the initial state. Figure 6.23 tracks these errors throughout the simulation.

The instantaneous errors $\delta\epsilon$ remain very small, oscillating around $10^{-6}$ with occasional spikes when flow impacts the circuit breaker walls. The global errors $\Delta\epsilon$ follow a more predictable

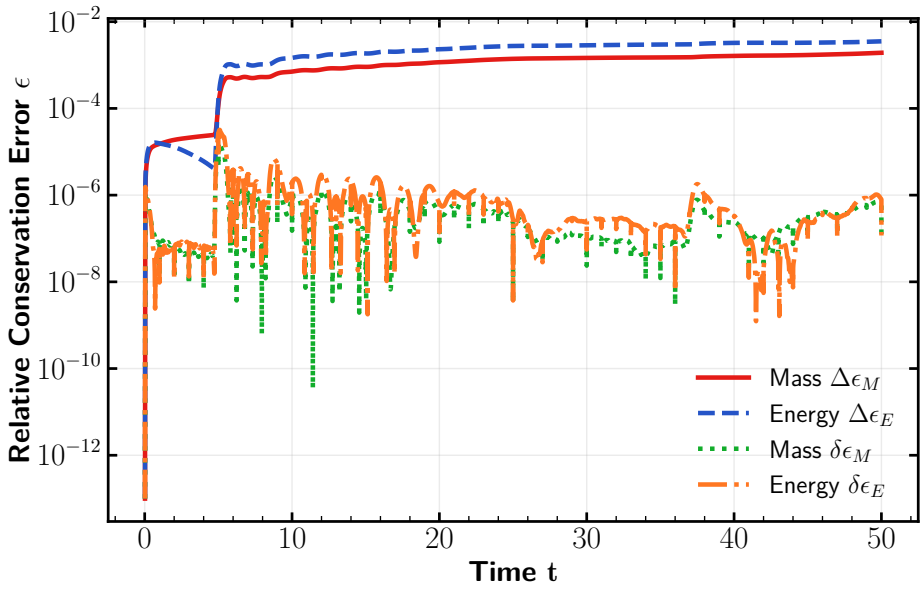



Figure 6.23 Conservation errors for the mass and energy computed using the IBM method.

pattern, initially accumulating but quickly stabilizing with minimal subsequent growth. At the beginning of the simulation, the flow within the expansion is parallel to the walls of the circuit breaker, and the the reconstruction introduced minimal conservation errors. However, when the flow first hit the nozzle wall, a spike is observed in the global errors. Thereafter, each subsequent flow-wall interaction caused spikes in the instantaneous errors, but the global errors remained stable and relatively small (below 1%).

### Conclusion

The IBM method was successfully implemented in three-dimensions for solving the Euler equations and was verified through multiple test cases. The method extend the semi-implicit least-squares reconstruction approach to three dimensions, and the reconstruction process was verified using polynomial functions and manufactured solutions. The solver accurately captures complex flow features including shock waves, contact discontinuities, and rarefaction waves, producing high-quality numerical solutions for complex three-dimensional geometries. One limitation of the method is the non-conservative nature of the reconstruction process. Nonetheless, the conservation errors were shown to be small even for moderately refined meshes. Extending the FRM, FIM, and FCM approaches developed in previous chapters to three dimensions could address the conservation issues in the IBM method through the use of cut-cells.

# CHAPTER 7 CONCLUSION

This chapter concludes the thesis by summarizing the main contributions and findings. It also discusses the limitations of the work and suggests directions for future research.

## 7.1 Summary of Works

The general objective of this thesis was to develop conservative and second-order accurate Cartesian grid methods for compressible flows applicable to complex three-dimensional geometry simulations. In pursuit of this objective, three main contributions were presented, addressing the conservation problem, the high-order accuracy requirement, and the application to three-dimensional configurations.

The first contribution, detailed in Chapter 4, addressed the conservation loss associated with the Immersed Boundary Method (IBM) when applied to compressible flows. It introduced three novel methods, namely Flux Redistribution Method (FRM), Flux Interpolation Method (FIM), and Flux Correction Method (FCM), that effectively recover conservation of mass, momentum, and energy in the context of IBM methods. The methods are based on a combination of the Cut-Cells (CC) method and the IBM method, where the discretization grid is constituted by cut-cells while boundary conditions are imposed using the IBM technique. To address the conservation loss inherent in the IBM approach, an additional conservative step is incorporated into the time integration iterative process. The FRM method redistributes cut-cells fluxes to neighboring cells under the condition of ensuring global conservation. In contrast, the FIM and FCM methods interpolate the fluxes at the cut-cells faces using optimization procedures. The modified fluxes are then used to update the cut-cells values, guaranteeing the conservation of mass, momentum and energy in each cell. FIM and FCM methods distinguish themselves by the way the modified fluxes are computed. The FIM fixes the final solution in the cut-cells region to the value obtained with the IBM reconstruction, then determines modified fluxes that minimize the error with respect to the original finite volume fluxes. Conversely, the FCM method employs a perturbation term to adjust the finite volume fluxes in the cut-cells region so that the difference between the final solution and that obtained with the IBM reconstruction is minimized. Numerical results using first-order accurate schemes on two-dimensional geometries demonstrated the effectiveness of these methods in preserving conservation of the conserved variables.

The second contribution, presented in Chapter 5, addressed the high-order accuracy require-

ment and the small cell problem associated with the Cut-Cells method. A quadratic semi-implicit least-squares reconstruction method was developed to achieve second-order accuracy in the cut-cells region. The Method of Manufactured Solutions (MMS) was employed to rigorously assess the numerical accuracy of the scheme, confirming that the approach attains second-order accuracy in both the interior and along the immersed boundaries. Additionally, the chapter presented complex flow test cases, including a simplified two-dimensional high-voltage circuit breaker (HVCB) geometry where the numerical results compared favorably with solutions obtained using body-fitted methods.

The final contribution, described in Chapter 6 and Appendices A and B, focused on the implementation of a high-order IBM method in three-dimensional geometries. A boundary representation approach with surface triangulation was employed to define general three-dimensional geometries as complex as high-voltage circuit breakers. The geometry was then immersed in a Cartesian grid that permits local refinement based on geometric features. The semi-implicit least-squares reconstruction method underlying the FRM, FIM and FCM methods was extended to three dimensions, enabling the imposition of boundary conditions on the immersed boundaries. This three-dimensional IBM method was verified through various test cases, including accuracy and convergence studies using the MMS approach and a simplified three-dimensional HVCB geometry. The circuit breaker simulation results exhibited complex flow features that were successfully captured by the method. Furthermore, comparisons with numerical results obtained using COMSOL software demonstrated good agreement between the two methods.

Collectively, these contributions demonstrate the effectiveness of the developed methods in addressing the challenges of conservation, high-order accuracy, and complex three-dimensional geometries. The numerical results obtained in this thesis indicate that these methods are capable of accurately simulating compressible flows in complex geometries, making them suitable for applications in high-voltage circuit breakers and other engineering problems.

## 7.2 Limitations

Although the methods developed in this research have shown promising results in addressing conservation challenges, high-order accuracy, and complex three-dimensional geometries, several limitations remain that warrant further investigation. The conservative methods (FRM, FIM and FCM) have been successfully applied to two-dimensional geometries; however, their extension to three-dimensional geometries using cut-cells was not implemented in this work. The conservative methods are based on a flux formulation that is dimension-independent and their extension to three-dimensional geometries is straightforward. With the reconstruction

procedure already implemented and verified in three dimensions, the only remaining step is the generation of the cut-cells in three dimensions. The computation of the cut-cells in three dimensions is more complex than in two dimensions, as it involves the exact clipping of the three-dimensional cells by the geometry. Potential simplifications include using pre-defined cut-cell templates to approximate actual cut-cells, thereby reducing the computational complexity of the cut-cells generation process. Such an approach would be particularly beneficial for simulations involving moving boundaries, where cut-cells must be regenerated at each time step.

The semi-implicit least-squares method employed for boundary condition imposition may become computationally expensive when applied to moving boundaries. With moving boundaries problems, the reconstruction stencil for each boundary cell changes at each time step, necessitating a complete reassembly of the global reconstruction system. This leads to increased computational costs, especially for complex three-dimensional geometries containing numerous boundary cells. Additionally, the cell tagging process used for cell classification (and the cut-cells computation for conservative methods) must be performed at each time step, further increasing computational demands. The tagging cost can be mitigated by using simpler definitions of the reconstructed cells and their centers, and by leveraging the tagging information from the previous time step. For the global reconstruction system, employing an iterative solver that uses the previous time step's solution as an initial guess could significantly reduce the number of iterations required for convergence. This approach would be particularly effective for circuit breaker simulations, where small time steps are typically taken which lead to minor variations in the solution between consecutive time steps.

The algorithm developed in the mesh generation process may face robustness issues due to floating-point precision errors. Certain geometric and topological queries within the mesh generation algorithms are highly sensitive to such errors, and inaccurate results can cause complete algorithmic failure. In the current implementation, some strategies have been developed to mitigate these issues (e.g. epsilon tolerance in floating-point comparisons, repetition of queries with different configurations, geometry perturbation, etc.). The ray-tracing algorithm used to determine whether a point lies inside or outside the geometry exemplifies a geometric query susceptible to floating-point precision errors. When a ray passes near the boundary of the geometry, floating-point errors may lead to misclassification of points. The current implementation addresses this by using tolerance values and, if uncertainty persists, repeating the algorithm with rays directed in different random directions. In practice, this strategy typically succeeds within one or two iterations. The strategies employed in the current implementation have been successful in most cases, but there are still some edge cases where the algorithm may probably fail. Consequently, a more comprehensive approach is

needed to ensure robustness in all cases without compromising algorithmic efficiency.

Finally, the mesh refinement criteria used in the current implementation are solely based on geometric features without considering flow characteristics. Integrating the mesh generation process with the flow solver would enable adaptive mesh refinement based on flow features, allowing for a more efficient use of computational resources and improved accuracy in regions of interest. Indeed, due to the robust reconstruction method used in this thesis, the solver can start with a relatively coarse mesh, without concern for geometric edge cases such as small gaps or highly concave regions, and let the mesh refinement process to adaptively refine flow regions of particular interest.

## 7.3 Future Research

Future research directions should address the limitations outlined above. Short-term objectives include extending the conservative methods to three-dimensional geometries, enhancing the robustness of the mesh generation algorithm, and incorporating flow-based criteria into the mesh refinement process.

Following completion of these initial tasks, attention should shift toward handling moving boundaries. This will require developing robust and efficient algorithms for geometry definition and mesh generation suitable for moving boundaries. Additionally, the semi-implicit least-squares method will need slight adaptation to efficiently accommodate moving boundaries. Overset grid techniques present a promising approach that could simplify many challenges associated with moving boundaries.

Another important research direction involves coupling Euler and Helmholtz solvers. This development would align the simulation more closely with the $MC^3$ model used for high-voltage circuit breaker simulation. Extending this coupling to specific applications in high-voltage circuit breakers would enable comprehensive multiphysics simulations of HVCB that incorporate various physical phenomena such as real gas behavior, radiative heat transfer and electromagnetic field effects.

# REFERENCES

[1] E. H. A. A. Ndiaye, J.-Y. Trépanier, R. D. H. Sousa, and S. Leclaire, "Improvement of the conservation properties of the immersed boundary method for inviscid compressible flows using cut-cells," in *The Canadian Society of Mechanical Engineering and Computational Fluid Dynamics Canada International Congress 2024*, . CSME/CFDSC, Ed., 2024.

[2] R.-H. Ni, "A multiple-grid scheme for solving the Euler equations," *AIAA Journal*, vol. 20, no. 11, pp. 1565–1571, 1982.

[3] E. H. A. A. Ndiaye, J.-Y. Trépanier, R. De Holanda Sousa, and S. Leclaire, "Conservative immersed boundary methods on cartesian grids for inviscid compressible flows simulation," *International Journal of Heat and Fluid Flow*, vol. 114, p. 109775, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0142727X25000335

[4] Y. Kieffel, T. Irwin, P. Ponchon, and J. Owens, "Green gas to replace sf6 in electrical grids," *IEEE Power and Energy Magazine*, vol. 14, no. 2, pp. 32–39, 2016.

[5] Jgremillot, "Self-blast circuit breaker chamber," 2006, [accessed 8-March-2025]. [Online]. Available: https://commons.wikimedia.org/wiki/File:Disjoncteur-selfblast.svg

[6] J.-Y. Trépanier, "Mc3 : algorithms for the modelling and computation of circuit-breaker chambers," École Polytechnique de Montréal, Technical Report, 1992. [Online]. Available: https://publications.polymtl.ca/9690/

[7] H. Pellegrin, "Étude numérique de l'influence du champ magnétique auto-induit dans les arcs électriques de disjoncteurs," Master's thesis, École Polytechnique de Montréal, 1995. [Online]. Available: https://publications.polymtl.ca/31851/

[8] E. S. D. Eby, "Simulation numérique du transfert radiatif dans les arcs de disjoncteurs à sf," Master's thesis, École Polytechnique de Montréal, 1997. [Online]. Available: https://publications.polymtl.ca/8986/

[9] D. Godin, "Calcul de compositions chimiques de plasmas à l'équilibre thermodynamique : application à la modélisation de l'ablation dans les disjoncteurs,"

Master's thesis, École Polytechnique de Montréal, 1998. [Online]. Available: https://publications.polymtl.ca/7531/

[10] A. Martin, "Simulation d'écoulements multi-espèces de plasma avec ablation des parois : applications aux disjoncteurs haute-tension," Ph.D. dissertation, École Polytechnique de Montréal, 2005. [Online]. Available: https://publications.polymtl.ca/7569/

[11] M. Melot, "Modélisation numérique du transfert radiatif par la méthode des volumes finis dans les disjoncteurs à sf6," Master's thesis, École Polytechnique de Montréal, 2009. [Online]. Available: https://publications.polymtl.ca/8417/

[12] G. Pernaudat, "Une approche multi-physiques et multi-échelles pour l'amélioration de l'efficacité de la modélisation et de la simulation des disjoncteurs haute-tension," Ph.D. dissertation, École Polytechnique de Montréal, December 2017. [Online]. Available: https://publications.polymtl.ca/2849/

[13] A. Mazaheri, "Numerical simulation of radiative heat transfer in a high voltage circuit breaker," Master's thesis, École Polytechnique de Montréal, April 2018. [Online]. Available: https://publications.polymtl.ca/3074/

[14] S. Namvar, "High fidelity numerical simulation of 3d arc extinction in a high voltage circuit breaker," Ph.D. dissertation, Polytechnique Montréal, April 2020. [Online]. Available: https://publications.polymtl.ca/5268/

[15] Y. Scheiffer, "Comparaison de résolutions elliptiques entre la méthode des frontières immergées et la méthode des volumes finis," Master's thesis, École Polytechnique de Montréal, December 2018. [Online]. Available: https://publications.polymtl.ca/3774/

[16] M. S. Ali, "Two dimensional compressible flow solver for moving geometries using immersed boundary method," Master's thesis, Polytechnique Montréal, December 2020. [Online]. Available: https://publications.polymtl.ca/5587/

[17] M. Awad, "An immersed boundary method approach using hierarchical and overlapping grids for unsteady aerodynamics," Ph.D. dissertation, Polytechnique Montréal, December 2021. [Online]. Available: https://publications.polymtl.ca/9911/

[18] R. D. H. Sousa, "Modélisation multiphysique des disjoncteurs à haute-tension avec le couplage entre les méthodes des frontières immergées et des maillages superposés," Ph.D. dissertation, Polytechnique Montréal, June 2022. [Online]. Available: https://publications.polymtl.ca/10375/

[19] J. W. Purvis and J. E. Burkhalter, "Prediction of critical mach number for store configurations," *AIAA Journal*, vol. 17, no. 11, pp. 1170–1177, 1979. [Online]. Available: https://doi.org/10.2514/3.7617

[20] B. Wedan and J. J. South, "A method for solving the transonic full-potential equation for general configurations," in *6th Computational Fluid Dynamics Conference Danvers*, 1983. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.1983-1889

[21] D. K. Clarke, M. Salas, and H. Hassan, "Euler calculations for multielement airfoils using cartesian grids," *AIAA Journal*, vol. 24, no. 3, pp. 353–358, 1986.

[22] J. R. Gaffney and H. Hassan, "Euler calculations for wings using cartesian grids," in *25th AIAA Aerospace Sciences Meeting*, 1987. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.1987-356

[23] J. Melton, M. Berger, M. Aftosmis, and M. Wong, "3d applications of a cartesian grid euler method," in *33rd Aerospace Sciences Meeting and Exhibit*, 1995. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.1995-853

[24] F. T. Johnson, E. N. Tinoco, and N. J. Yu, "Thirty years of development and application of cfd at boeing commercial airplanes, seattle," *Computers & Fluids*, vol. 34, no. 10, pp. 1115–1151, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045793005000125

[25] I. Stroud, *Boundary Representation Modelling Techniques*. Springer London, 2009. [Online]. Available: https://link.springer.com/book/10.1007/978-1-84628-616-2

[26] J. D. Foley, *Computer graphics: principles and practice*. Addison-Wesley Professional, 1996, vol. 12110.

[27] M. J. Aftosmis, "Solution adaptive cartesian grid methods for aerodynamic flows with complex geometries," communication presented at 28th Computational Fluid Dynamics Lecture Series, Belgium, 3-7 mars 1997. [Online]. Available: https://www.nas.nasa.gov/publications/software/docs/cart3d/pages/publications/aftosmis_97vkiNotes.pdf

[28] M. J. Berger and M. J. Aftosmis, "An ode-based wall model for turbulent flow simulations," *AIAA Journal*, vol. 56, no. 2, pp. 700–714, 2018. [Online]. Available: https://doi.org/10.2514/1.J056151

[29] W. Dawes, S. Harvey, S. Fellows, C. Favaretto, and A. Velivelli, "Viscous layer meshes from level sets on cartesian meshes," in *45th AIAA Aerospace Sciences Meeting and Exhibit*, 2007. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2007-555

[30] K. J. Fidkowski, "A simplex cut-cell adaptive method for high–order discretizations of the compressible Navier-Stokes equations," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2007. [Online]. Available: https://dspace.mit.edu/handle/1721.1/39701

[31] M. Berger, "Chapter 1 - cut cells: Meshes and solvers," in *Handbook of Numerical Methods for Hyperbolic Problems*, ser. Handbook of Numerical Analysis, vol. 18, 2017, pp. 1–22.

[32] R. Courant, K. Friedrichs, and H. Lewy, "Über die partiellen differenzengleichungen der mathematischen physik," *Mathematische Annalen*, vol. 100, no. 1, pp. 32–74, 1928. [Online]. Available: https://doi.org/10.1007/BF01448839

[33] J. D. Hunt, "An adaptive three-dimensional cartesian approach for the parallel computation of inviscid flow about static and dynamic configurations," Ph.D. dissertation, University Of Michigan, 2004. [Online]. Available: https://deepblue.lib.umich.edu/handle/2027.42/124078

[34] S. Bayyuk, K. Powell, and B. V. Leer, "A simulation technique for 2-d unsteady inviscid flows around arbitrarily moving and deforming bodies of arbitrary geometry," in *11th Computational Fluid Dynamics Conference*, 1993. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.1993-3391

[35] T. Ye, R. Mittal, H. Udaykumar, and W. Shyy, "An accurate cartesian grid method for viscous incompressible flows with complex immersed boundaries," *Journal of Computational Physics*, vol. 156, no. 2, pp. 209–240, 1999.

[36] N. Domel and J. Steve Karman, "Splitflow - progress in 3d cfd with cartesian omni-tree grids for complex geometries," in *38th Aerospace Sciences Meeting and Exhibit*, 2000. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2000-1006

[37] H. Udaykumar, R. Mittal, P. Rampunggoon, and A. Khanna, "A sharp interface cartesian grid method for simulating flows with complex moving boundaries," *Journal of Computational Physics*, vol. 174, no. 1, pp. 345–380, 2001.

[38] D. Ingram, D. Causon, and C. Mingham, "Developments in cartesian cut cell methods," *Mathematics and Computers in Simulation*, vol. 61, no. 3, pp. 561–572, 2003, mODELLING 2001 - Second IMACS Conference on Mathematical Modelling and Computational Methods in Mechanics, Physics, Biomechanics and

Geodynamics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378475402001076

[39] W. J. Coirier and K. G. Powell, "An accuracy assessment of cartesian-mesh approaches for the euler equations," *Journal of Computational Physics*, vol. 117, no. 1, pp. 121–131, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999185710509

[40] J. J. Quirk, "An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex two-dimensional bodies," *Computers & Fluids*, vol. 23, no. 1, pp. 125–142, 1994. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0045793094900310

[41] M. P. Kirkpatrick, S. W. Armfield, and J. H. Kent, "A representation of curved boundaries for the solution of the Navier-Stokes equations on a staggered three-dimensional cartesian grid," *Journal of Computational Physics*, vol. 184, no. 1, pp. 1–36, 2003.

[42] D. Hartmann, M. Meinke, and W. Schröder, "An adaptive multilevel multigrid formulation for cartesian hierarchical grid methods," *Computers & Fluids*, vol. 37, no. 9, pp. 1103–1125, 2008.

[43] ——, "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, pp. 1038–1052, 2011.

[44] M. Berger and A. Giuliani, "A state redistribution algorithm for finite volume schemes on cut cell meshes," *Journal of Computational Physics*, vol. 428, 2021.

[45] A. Giuliani, A. Almgren, J. Bell, M. Berger, M. Henry de Frahan, and D. Rangarajan, "A weighted state redistribution algorithm for embedded boundary grids," *Journal of Computational Physics*, vol. 464, p. 111305, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999122003679

[46] M. Berger and A. Giuliani, "A new provably stable weighted state redistribution algorithm," *SIAM Journal on Scientific Computing*, vol. 46, no. 5, pp. A2848–A2873, 2024. [Online]. Available: https://doi.org/10.1137/23M1597484

[47] I. Barrio Sanchez, A. Almgren, J. Bell, M. Henry de Frahan, and W. Zhang, "A new re-redistribution scheme for weighted state redistribution with adaptive mesh refinement," *Journal of Computational Physics*, vol. 504, p. 112879, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999124001281

[48] C. G. Taylor, L. C. Wilcox, and J. Chan, "An energy stable high-order cut cell discontinuous galerkin method with state redistribution for wave propagation," *Journal of Computational Physics*, vol. 521, p. 113528, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999124007769

[49] I.-L. Chern and P. Colella, "A conservative front tracking method for hyperbolic conservation laws," Lawrence Liver-more National Laboratory, Tech. Rep., 1987.

[50] J. B. Bell, M. L. Welcome, and P. Colella, "Conservative front-tracking for inviscid compressible flow," in *10th Computational Fluid Dynamics Conference*, 1991.

[51] R. B. Pember, J. B. Bell, P. Colella, W. Y. Curtchfield, and M. L. Welcome, "An adaptive cartesian grid method for unsteady compressible flow in irregular regions," *Journal of Computational Physics*, vol. 120, no. 2, pp. 278–304, 1995.

[52] P. Colella, D. T. Graves, B. J. Keen, and D. Modiano, "A cartesian grid embedded boundary method for hyperbolic conservation laws," *Journal of Computational Physics*, vol. 211, no. 1, pp. 347–366, 2006.

[53] P. Colella, D. Graves, T. Ligocki, D. Modiano, and B. Van Straalen, *EBChombo software package for Cartesian grid, embedded boundary applications*, 2014. [Online]. Available: https://crd.lbl.gov/assets/pubs_presos/ebmain.pdf

[54] R. J. Leveque, "A large time step generalization of godunov's method for systems of conservation laws," *SIAM Journal on Numerical Analysis*, vol. 22, no. 6, pp. 1051–1073, 1985. [Online]. Available: http://www.jstor.org/stable/2157537

[55] M. J. Berger and R. J. Leveque, "An adaptive cartesian mesh algorithm for the euler equations in arbitrary geometries," in *9th Computational Fluid Dynamics Conference*, 1989. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.1989-1930

[56] R. J. LeVeque and K.-M. Shyue, "Two-dimensional front tracking based on high resolution wave propagation methods," *Journal of Computational Physics*, vol. 123, no. 2, pp. 354–368, 1996. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999196900297

[57] C. Helzel, M. J. Berger, and R. J. Leveque, "A high-resolution rotated grid method for conservation laws with embedded geometries," *SIAM Journal on Scientific Computing*, vol. 26, no. 3, pp. 785–809, 2005.

[58] M. Berger and C. Helzel, "A simplified h-box method for embedded boundary grids," *SIAM Journal on Scientific Computing*, vol. 34, no. 2, pp. A861–A888, 2012.

[59] S. May and M. Berger, "An explicit implicit scheme for cut cells in embedded boundary meshes," *Journal of Scientific Computing*, vol. 71, no. 3, pp. 919–943, 2017.

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

[61] R. Verzicco, "Immersed boundary methods: Historical perspective and future outlook," *Annual Review of Fluid Mechanics*, vol. 55, no. Volume 55, 2023, pp. 129–155, 2023. [Online]. Available: https://www.annualreviews.org/content/journals/10.1146/annurev-fluid-120720-022129

[62] M. W. Evans and F. H. Harlow, "The particle-in-cell method for hydrodynamic calculations," Los Alamos Scientific Lab., N. Mex., Tech. Rep., 1957.

[63] M. Rich and S. S. Blackman, "A method for eulerian fluid dynamics," Los Alamos Scientific Lab., N. Mex., Tech. Rep., 1962.

[64] R. A. Gentry, R. E. Martin, and B. J. Daly, "An eulerian differencing method for unsteady compressible flow problems," *Journal of Computational Physics*, vol. 1, no. 1, pp. 87–118, 1966. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0021999166900143

[65] J. Viecelli, "A method for including arbitrary external boundaries in the mac incompressible fluid computing technique," *Journal of Computational Physics*, vol. 4, no. 4, pp. 543–551, 1969. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0021999169900199

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

[67] B. E. Griffith and N. A. Patankar, "Immersed methods for fluid–structure interaction," *Annual Review of Fluid Mechanics*, vol. 52, no. Volume 52, 2020, pp. 421–448, 2020. [Online]. Available: https://www.annualreviews.org/content/journals/10.1146/annurev-fluid-010719-060228

[68] C. S. Peskin, "The fluid dynamics of heart valves: Experimental, theoretical, and computational methods," *Annual Review of Fluid Mechanics*, vol. 14, no. 1, pp. 235–259, 1982.

[69] L. J. Fauci and C. S. Peskin, "A computational model of aquatic animal locomotion," *Journal of Computational Physics*, vol. 77, no. 1, pp. 85–108, 1988.

[70] L. J. Fauci and A. McDonald, "Sperm motility in the presence of boundaries," *Bulletin of Mathematical Biology*, vol. 57, no. 5, pp. 679–699, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/009282409500022I

[71] W. Kou, A. P. S. Bhalla, B. E. Griffith, J. E. Pandolfino, P. J. Kahrilas, and N. A. Patankar, "A fully resolved active musculo-mechanical model for esophageal transport," *Journal of Computational Physics*, vol. 298, pp. 446–465, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999115003897

[72] W. Kou, J. E. Pandolfino, P. J. Kahrilas, and N. A. Patankar, "Studies of abnormalities of the lower esophageal sphincter during esophageal emptying based on a fully coupled bolus–esophageal–gastric model," *Biomechanics and Modeling in Mechanobiology*, vol. 17, no. 4, pp. 1069–1082, 2018. [Online]. Available: https://doi.org/10.1007/s10237-018-1014-y

[73] J. H. Lee, A. D. Rygg, E. M. Kolahdouz, S. Rossi, S. M. Retta, N. Duraiswamy, L. N. Scotten, B. A. Craven, and B. E. Griffith, "Fluid–structure interaction models of bioprosthetic heart valve dynamics in an experimental pulse duplicator," *Annals of Biomedical Engineering*, vol. 48, no. 5, pp. 1475–1490, 2020. [Online]. Available: https://doi.org/10.1007/s10439-020-02466-4

[74] J. M. Huang, M. J. Shelley, and D. B. Stein, "A stable and accurate scheme for solving the stefan problem coupled with natural convection using the immersed boundary smooth extension method," *Journal of Computational Physics*, vol. 432, p. 110162, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999121000541

[75] E. Balaras, "Modeling complex boundaries using an external force field on fixed cartesian grids in large-eddy simulations," *Computers & Fluids*, vol. 33, no. 3, pp. 375–404, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045793003000586

[76] R. Mittal, H. Dong, M. Bozkurttas, F. Najjar, A. Vargas, and A. von Loebbecke, "A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries," *Journal of Computational Physics*, vol. 227, no. 10, pp. 4825–4852, 2008.

[77] 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, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999105004778

[78] 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, 2007.

[79] G. Iaccarino and S. Moreau, "Natural and forced conjugate heat transfer in complex geometries on cartesian adapted grids," *Journal of Fluids Engineering*, vol. 128, no. 4, pp. 838–846, 12 2005. [Online]. Available: https://doi.org/10.1115/1.2201625

[80] P. Łapka and P. Furmański, "Immersed boundary method for radiative heat transfer problems in nongray media with complex internal and external boundaries," *Journal of Heat Transfer*, vol. 139, no. 2, p. 022702, 11 2016. [Online]. Available: https://doi.org/10.1115/1.4034772

[81] R. Mittal and G. Iaccarino, "Immersed boundary methods," *Annual Review of Fluid Mechanics*, vol. 37, no. 1, pp. 239–261, 2005.

[82] P. Gould, *Introduction to Linear Elasticity*. Springer New York, 2012. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4614-4833-4

[83] C. S. Peskin, "The immersed boundary method," *Acta Numerica*, vol. 11, pp. 479–517, 2002.

[84] P. Angot, C.-H. Bruneau, and P. Fabrie, "A penalization method to take into account obstacles in incompressible viscous flows," *Numerische Mathematik*, vol. 81, no. 4, pp. 497–520, 1999.

[85] K. Khadra, P. Angot, S. Parneix, and J.-P. Caltagirone, "Fictitious domain approach for numerical modelling of navier–stokes equations," *International Journal for Numerical Methods in Fluids*, vol. 34, no. 8, pp. 651–684, 2000.

[86] G. K. Batchelor, *An Introduction to Fluid Dynamics*, ser. Cambridge Mathematical Library. Cambridge University Press, 2000.

[87] H. C. Brinkman, "A calculation of the viscous force exerted by a flowing fluid on a dense swarm of particles," *Flow, Turbulence and Combustion*, vol. 1, no. 1, pp. 27–34, 1949. [Online]. Available: https://doi.org/10.1007/BF02120313

[88] 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, 1993. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999183710818

[89] G. Iaccarino and R. Verzicco, "Immersed boundary technique for turbulent flow simulations," *Applied Mechanics Reviews*, vol. 56, no. 3, pp. 331–347, 2003.

[90] P. Lavoie, E. Radenac, G. Blanchard, E. Laurendeau, and P. Villedieu, "Immersed boundary methodology for multistep ice accretion using a level set," *Journal of Aircraft*, vol. 59, no. 4, pp. 912–926, 2022. [Online]. Available: https://doi.org/10.2514/1.C036492

[91] M.-C. Lai and C. S. Peskin, "An immersed boundary method with formal second-order accuracy and reduced numerical viscosity," *Journal of Computational Physics*, vol. 160, no. 2, pp. 705–719, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999100964830

[92] M. Gazzola, P. Chatelain, W. M. Van Rees, and P. Koumoutsakos, "Simulations of single and multiple swimmers with non-divergence free deforming geometries," *Journal of Computational Physics*, vol. 230, no. 19, pp. 7093–7114, 2011.

[93] R. P. Beyer, "A computational model of the cochlea using the immersed boundary method," *Journal of Computational Physics*, vol. 98, no. 1, pp. 145–162, 1992. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0021999192901807

[94] J. Mohd-Yusof, "Combined immersed-boundary/B-spline methods for simulations of flow in complex geometries," *Center for Turbulence Research, Annual Research Briefs*, vol. 161, no. 1, pp. 317–327, 1997.

[95] R. Verzicco, J. Mohd-Yusof, P. Orlandi, and D. Haworth, "Les in complex geometries using boundary body forces," in *Proceedings of the summer program.* Center for Turbulence Research Stanford, CA, USA, 1998, pp. 171–186.

[96] E. Fadlun, R. Verzicco, P. Orlandi, and J. Mohd-Yusof, "Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations," *Journal of Computational Physics*, vol. 161, no. 1, pp. 35–60, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999100964842

[97] P. De Palma, M. De Tullio, G. Pascazio, and M. Napolitano, "An immersed-boundary method for compressible viscous flows," *Computers & fluids*, vol. 35, no. 7, pp. 693–702, 2006.

[98] J. Liu, N. Zhao, O. Hu, M. Goman, and X. K. Li, "A new immersed boundary method for compressible navier–stokes equations," *International Journal of Computational Fluid Dynamics*, vol. 27, no. 3, pp. 151–163, 2013.

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

[100] H. Uddin, R. M. 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, 2014.

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

[102] 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, 2017.

[103] F. Capizzano, *A Compressible Flow Simulation System Based on Cartesian Grids with Anisotropic Refinements*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2007-1450

[104] M. de Tullio, P. De Palma, G. Iaccarino, G. Pascazio, and M. Napolitano, "An immersed boundary method for compressible flows using local grid refinement," *Journal of Computational Physics*, vol. 225, no. 2, pp. 2098–2117, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999107001222

[105] N. Peller, A. L. Duc, F. Tremblay, and M. Manhart, "High-order stable interpolations for immersed boundary methods," *International Journal for Numerical Methods in Fluids*, vol. 52, no. 11, pp. 1175–1193, 2006. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.1227

[106] C. Brehm, M. F. Barad, and C. C. Kiris, "An immersed boundary method for solving the compressible navier-stokes equations with fluid-structure interaction," in *34th AIAA Applied Aerodynamics Conference*, 2016, p. 3265.

[107] M. S. Ali, R. de Holanda Sousa, M. O. Awad, R. Camarero, and J.-Y. Trépanier, "Implicit interpolation method for immersed boundary methods," *Journal of Engineering Mathematics*, vol. 146, no. 1, p. 5, 2024. [Online]. Available: https://doi.org/10.1007/s10665-024-10357-z

[108] C. B. Laney, *Computational Gasdynamics*. Cambridge University Press, 1998.

[109] E. F. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics*, 3rd ed. Springer Berlin, Heidelberg, 2013.

[110] R. J. LeVeque, *Finite volume methods for hyperbolic problems*. Cambridge university press, 2002, vol. 31.

[111] W. L. Oberkampf and F. G. Blottner, "Issues in computational fluid dynamics code verification and validation," *AIAA Journal*, vol. 36, no. 5, pp. 687–695, 1998. [Online]. Available: https://doi.org/10.2514/2.456

[112] P. J. Roache, *Verification and validation in computational science and engineering*. Hermosa Albuquerque, NM, 1998, vol. 895.

[113] ——, "Code verification by the method of manufactured solutions," *J. Fluids Eng.*, vol. 124, no. 1, pp. 4–10, 2002.

[114] R. J. LeVeque, *Numerical methods for conservation laws*. Springer, 1992, vol. 214.

[115] P. L. Roe, "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, vol. 43, no. 2, pp. 357–372, 1981.

[116] J. Blazek, *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, 2015.

[117] G. A. Sod, "A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws," *Journal of Computational Physics*, vol. 27, no. 1, pp. 1–31, 1978.

[118] S. Brahmachary, G. Natarajan, V. Kulkarni, and N. Sahoo, "A sharp-interface immersed boundary framework for simulations of high-speed inviscid compressible flows," *International Journal for Numerical Methods in Fluids*, vol. 86, no. 12, pp. 770–791, 2018.

[119] J. Anderson, *Modern Compressible Flow: With Historical Perspective*, ser. McGraw-Hill Series in Aeronautical and Aerospace Engineering. McGraw-Hill Education, 2021.

[120] F. Delarue, F. Lagoutière, and N. Vauchelet, "Convergence order of upwind type schemes for transport equations with discontinuous coefficients," *Journal de Mathématiques Pures et Appliquées*, vol. 108, no. 6, pp. 918–951, 2017.

[121] M. Aftosmis, D. Gaitonde, and T. S. Tavares, "Behavior of linear reconstruction techniques on unstructured meshes," *AIAA Journal*, vol. 33, no. 11, pp. 2038–2049, 1995.

[122] L. Krivodonova and M. Berger, "High-order accurate implementation of solid wall boundary conditions in curved geometries," *Journal of Computational Physics*, vol. 211, no. 2, pp. 492–512, 2006.

[123] H. Forrer and R. Jeltsch, "A higher-order boundary treatment for cartesian-grid methods," *Journal of Computational Physics*, vol. 140, no. 2, pp. 259–277, 1998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999198958910

[124] C. J. Roy, C. Nelson, T. Smith, and C. Ober, "Verification of euler/navier–stokes codes using the method of manufactured solutions," *International Journal for Numerical Methods in Fluids*, vol. 44, no. 6, pp. 599–620, 2004.

[125] T. Barth and D. Jespersen, "The design and application of upwind schemes on unstructured meshes," in *27th Aerospace sciences meeting*, 1989, p. 366.

[126] V. Venkatakrishnan, "Convergence to steady state solutions of the euler equations on unstructured grids with limiters," *Journal of Computational Physics*, vol. 118, no. 1, pp. 120–130, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999185710844

[127] C.-W. Shu and S. Osher, "Efficient implementation of essentially non-oscillatory shock-capturing schemes," *Journal of computational physics*, vol. 77, no. 2, pp. 439–471, 1988.

[128] F. Navah and S. Nadarajah, "A comprehensive high-order solver verification methodology for free fluid flows," *Aerospace Science and Technology*, vol. 80, pp. 101–126, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1270963817318357

[129] M.-S. Liou and C. J. Steffen, "A new flux splitting scheme," *Journal of Computational Physics*, vol. 107, no. 1, pp. 23–39, 1993. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999183711228

[130] E. F. Toro, M. Spruce, and W. Speares, "Restoration of the contact surface in the hll-riemann solver," *Shock Waves*, vol. 4, no. 1, pp. 25–34, 1994. [Online]. Available: https://doi.org/10.1007/BF01414629

[131] J. Glimm, "Solutions in the large for nonlinear hyperbolic systems of equations," *Communications on Pure and Applied Mathematics*, vol. 18, no. 4, pp. 697–715, 1965. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160180408

[132] A. J. Chorin, "Random choice solution of hyperbolic systems," *Journal of Computational Physics*, vol. 22, no. 4, pp. 517–533, 1976. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0021999176900474

[133] G. A. Sod, "A numerical study of a converging cylindrical shock," *Journal of Fluid Mechanics*, vol. 83, no. 4, pp. 785–794, 1977.

[134] J. Yang, A. Sasoh, and K. Takayama, "The reflection of a shock wave over a cone," *Shock Waves*, vol. 6, no. 5, pp. 267–273, 1996. [Online]. Available: https://doi.org/10.1007/BF02535740

[135] "Comsol multiphysics® v. 6.1," COMSOL AB, Stockholm, Sweden, 2023. [Online]. Available: www.comsol.com

[136] C. Ericson, *Real-Time Collision Detection*. USA: CRC Press, Inc., 2004.

[137] T. Akenine-Möller, "Fast 3d triangle-box overlap testing," in *ACM SIGGRAPH 2005 Courses*. New York, NY, USA: Association for Computing Machinery, 2005. [Online]. Available: https://doi.org/10.1145/1198555.1198747

# APPENDIX A THREE-DIMENSIONAL GEOMETRY DEFINITION

The computational domain is defined through its boundary using a boundary representation (b-rep) approach [25]. This b-rep approach is well-suited for representing complex geometries and is widely supported in various CAD software applications. Furthermore, this approach is particularly appropriate for the immersed boundary method, where boundary geometric information is essential for identifying computational cells that lie inside or outside the computational domain. The boundary is represented by a set of triangular facets, which are organized in a hierarchical structure referred to as a *discrete topology*, as illustrated in Figure A.1.

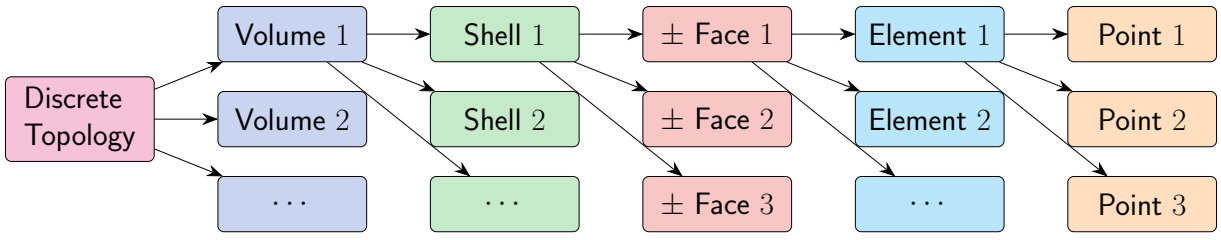



Figure A.1 Hierarchical structure of the discrete topology.

The discrete topology consists of one or multiple `Volume` structures, which constitute the top level of the hierarchy. Each volume describes a volumetric computational domain within which a partial differential equation (PDE) is solved. Different PDEs may be solved in different volumes, and these volumes can be disjoint or overlapping, thereby facilitating multiphysics simulations. Each volume is delimited by `Shell` structures, which are collections of triangular facets forming closed surfaces. The first shell represents the outer boundary of the volume, while any remaining shells constitute inner boundaries. To facilitate the definition of the computational domain, shells are further subdivided into `Face` structures, which are collections of `Elements` (triangular facets). Each element belongs to exactly one face, while each face may belong to multiple shells. Elements serve as the fundamental building blocks of the discrete topology and are each defined by three `Points` representing the vertices of the triangle that constitutes the element boundary. The element vertices are ordered such that normal vectors computed using the right-hand rule for elements within the same face are all oriented in a consistent direction. For instance, if a face is planar, all its element normal vectors are oriented in the same direction. Similarly, if a face forms a closed surface, all its element normal vectors point either inward or outward relative to the surface.

When forming a shell, each constituent face is assigned an orientation sign indicating the direction of the face's normal vector with respect to the shell, with a positive sign indicating that the normal vector points outward from the shell.

Figure A.2 presents an example of a discrete topology comprising two volumes, two shells, and seven faces. The first volume ($\texttt{V}_1$) corresponds to a ball delimited by the spherical shell ($\texttt{S}_1$), which consists of a single face ($\texttt{F}_1$). The second volume ($\texttt{V}_2$) is formed by a cube ($\texttt{S}_2$) containing an inner cavity ($\texttt{S}_1$). Shell $\texttt{S}_2$ comprises six planar faces ($\texttt{F}_2$ through $\texttt{F}_7$). Although shell $\texttt{S}_2$ could have been defined using a single face, multiple faces are employed in this example to demonstrate how faces can be associated with boundary conditions. Since boundary conditions are defined at the face level, dividing the cube into multiple faces allows for straightforward specification of different boundary conditions on each face. It is noteworthy that the number of elements differs significantly between the planar faces ($\texttt{F}_2$ through $\texttt{F}_7$) and the spherical face ($\texttt{F}_1$). As the geometry definition is independent of the mesh, larger elements can be used to represent planar faces, while smaller elements are required to accurately represent curved surfaces.

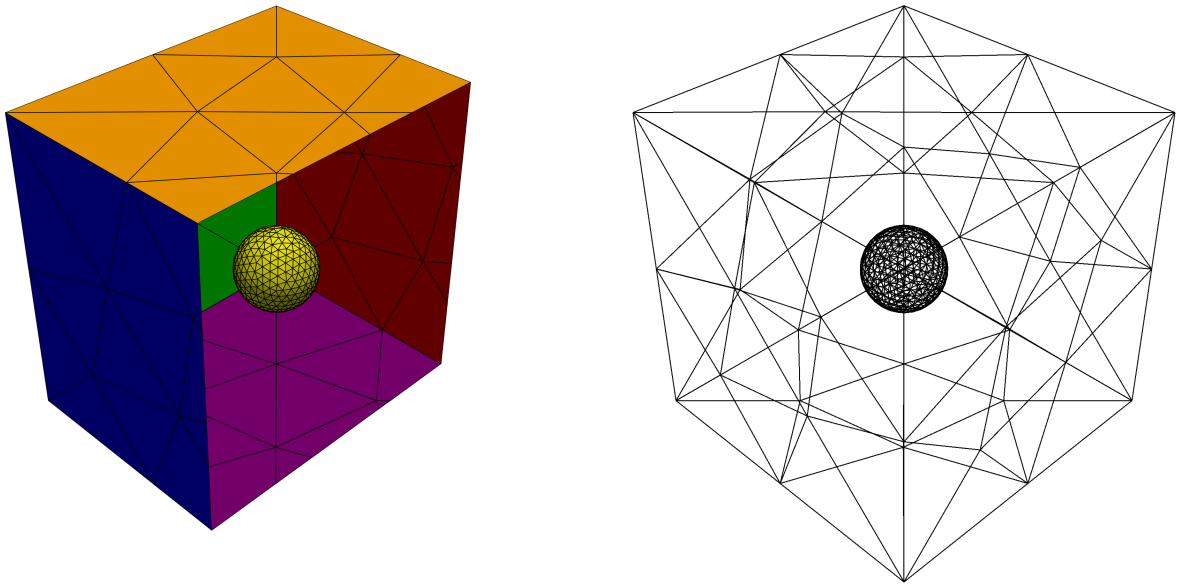


Figure A.2 Example of a discrete topology with two volumes, two shells, and seven faces.

Figure A.3 presents a more sophisticated example of a discrete topology representing a simplified high-voltage circuit breaker. This advanced discrete topology comprises a single computational volume bounded by one shell. The shell itself is partitioned into 25 distinct faces, which collectively contain 275,332 triangular elements.

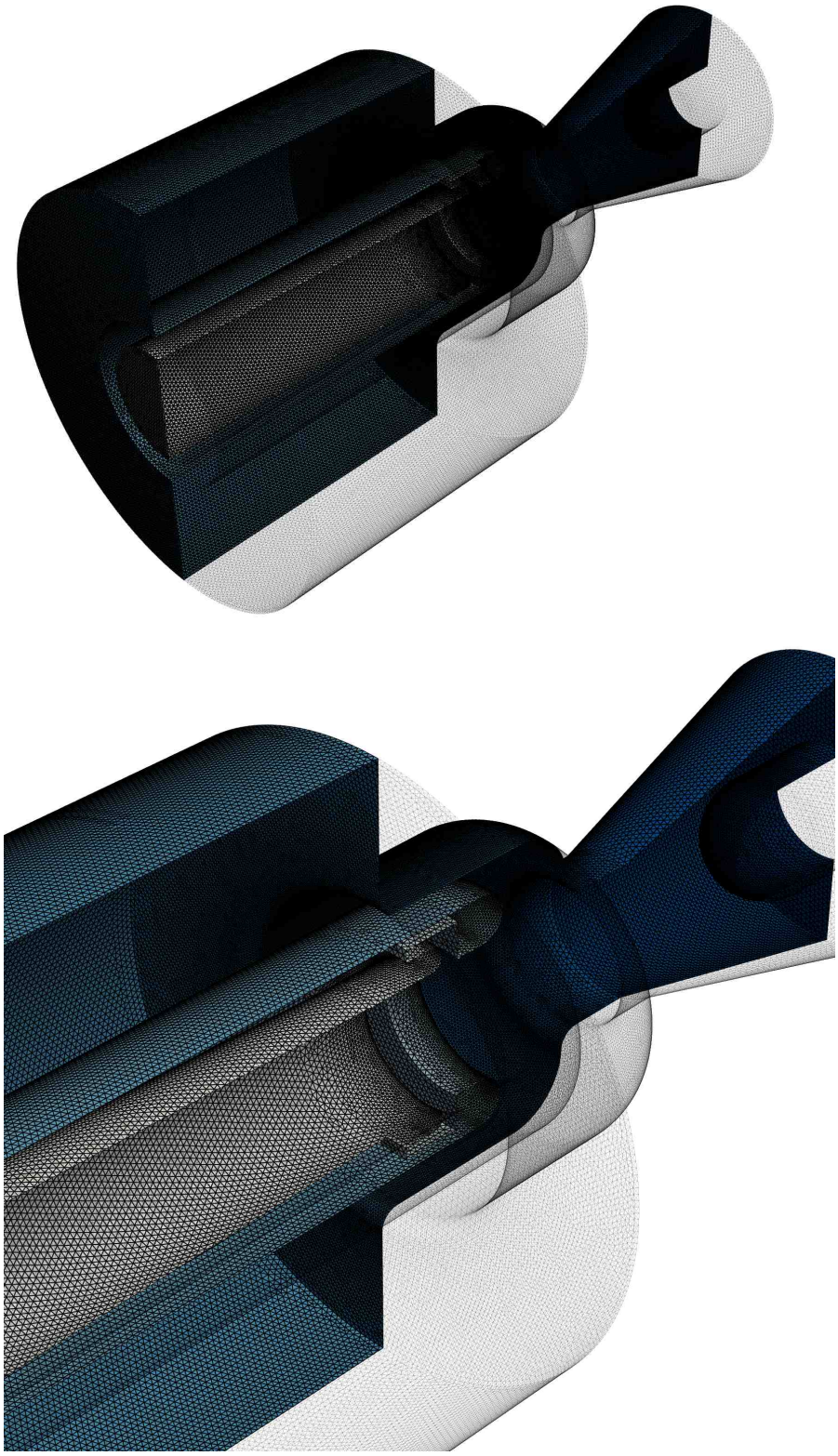


Figure A.3 Example of a complex discrete topology representing a simplified high-voltage circuit breaker.

# APPENDIX B CARTESIAN MESH GENERATION

The mesh generation process aims to generate a Cartesian grid around a given discrete topology. Key objectives include identifying the position of each grid cell relative to the topology (classified as inside, outside, or intersected) and enabling local grid refinement based on geometric criteria. Additionally, the list of elements of the topology intersecting with each grid cells is required for imposing boundary conditions. This appendix describes the complete process in detail, beginning with grid creation and refinement methodologies, followed by the computation of intersections between the grid and the topology, and concluding with the classification (tagging) of grid cells based on their position relative to the topology.

## B.1 Grid Definition and Refinement

This section describes the methodology for creating a Cartesian grid around the discrete topology and the algorithms used for local grid refinement.

### B.1.1 Grid Definition

The discrete topology is immersed inside a *hierarchical* Cartesian grid, composed of cells with varying refinement levels. Cells in the initial uniform grid are assigned refinement level 0, and when a cell with level $n$ is refined, it is subdivided into eight new cells at refinement level $n+1$..

The grid is fully defined by its constituent cells and the positions of the grid lines in the three-dimensional space. Each grid cell C is characterized by its refinement level $n$ and its local position in the Cartesian grid $(i, j, k)$, where $i$, $j$, and $k$ represent the local indices of the cell in the three coordinate directions. These local indices are defined relative to the cell's refinement level $n$. The notation $\mathrm{C}^n_{i,j,k}$ denotes a cell with refinement level $n$ and local indices $(i, j, k)$. The position of the cell in three-dimensional space is determined from the grid line positions in the initial uniform grid using Equation B.1.

$$\mathrm{C}^n_{i,j,k} : [x^n_i, x^n_i + \Delta x^n] \times \left[y^n_j, y^n_j + \Delta y^n\right] \times [z^n_k, z^n_k + \Delta z^n] \tag{B.1}$$

$$\alpha^n_i = \alpha_0 + i\Delta\alpha^n, \quad \Delta\alpha^n = \frac{1}{2^n}\Delta\alpha^0, \quad \Delta\alpha^0 = \frac{\alpha^g_{\max} - \alpha^g_{\min}}{N^0_\alpha}, \quad \alpha \in \{x, y, z\} \tag{B.2}$$

where:

- $x_i^n$, $y_j^n$, and $z_k^n$ represent the coordinates of the minimum corner of cell $\mathrm{C}_{i,j,k}^n$.
- $\Delta x^n$, $\Delta y^n$, and $\Delta z^n$ denote the cell dimensions at refinement level $n$ in the x, y, and z directions, respectively.
- $\Delta x^0$, $\Delta y^0$, and $\Delta z^0$ are the cell dimensions in the initial uniform grid (level 0).
- $N_x^0$, $N_y^0$, and $N_z^0$ specify the number of cells in the initial uniform grid along each direction.
- $[x_{\min}^g, x_{\max}^g]$, $[y_{\min}^g, y_{\max}^g]$, and $[z_{\min}^g, z_{\max}^g]$ are the coordinates of the grid bounding box.
- $x_0$, $y_0$, and $z_0$ denote the coordinates of the minimum corner of the grid (i.e., $x_0 = x_{\min}^g$, $y_0 = y_{\min}^g$, $z_0 = z_{\min}^g$).

In the mesh generation process, the grid is created using an initial uniform grid with a given number of cells in each direction ($N_x^0$, $N_y^0$, and $N_z^0$) and the given dimensions of the grid bounding box ($[x_{\min}^g, x_{\max}^g]$, $[y_{\min}^g, y_{\max}^g]$, and $[z_{\min}^g, z_{\max}^g]$). The grid bounding box is typically derived from the bounding box of the discrete topology ($[x_{\min}^t, x_{\max}^t]$, $[y_{\min}^t, y_{\max}^t]$, and $[z_{\min}^t, z_{\max}^t]$) by adding margins, as described by Equation B.3:

$$\alpha_{\min}^g = \alpha_{\min}^t - \epsilon_{\min}^\alpha(\alpha_{\max}^t - \alpha_{\min}^t), \quad \alpha_{\max}^g = \alpha_{\max}^t + \epsilon_{\max}^\alpha(\alpha_{\max}^t - \alpha_{\min}^t), \quad \alpha \in \{x, y, z\} \tag{B.3}$$

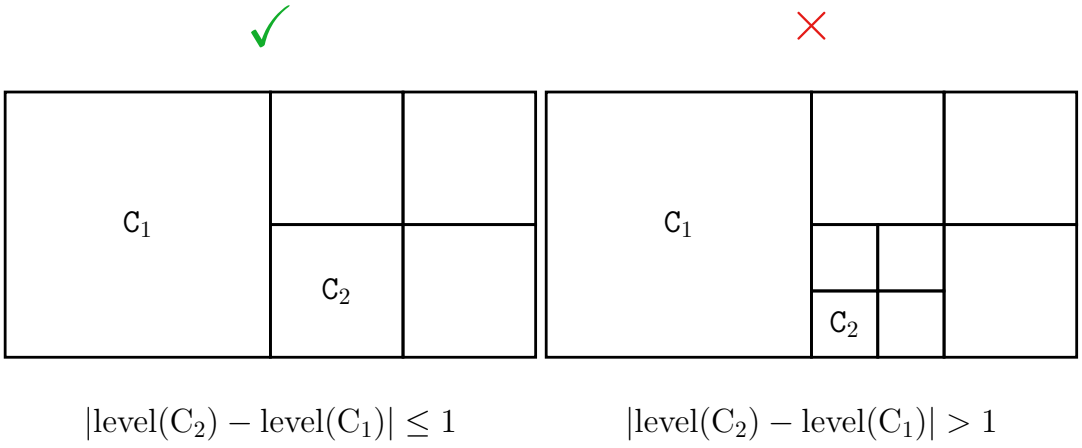
where $\epsilon_{\min}^\alpha$, $\epsilon_{\max}^\alpha$ represent the relative margin factors applied in each direction. The subsequent section details the algorithm used for local grid refinement.

### B.1.2 Grid Refinement

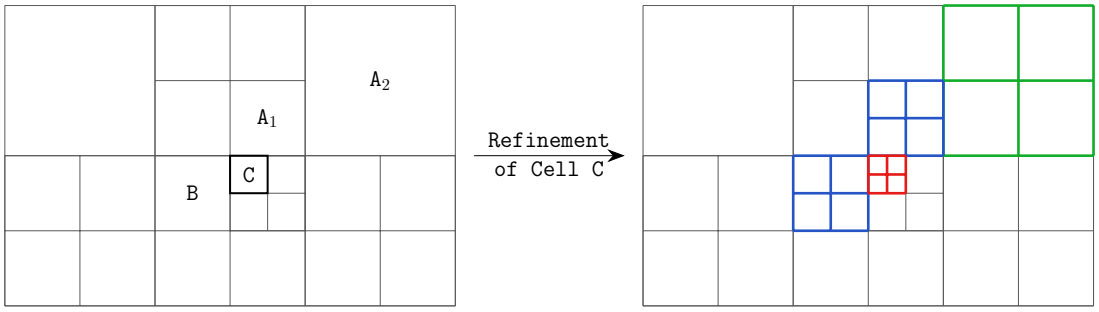
This subsection details the cell refinement algorithm and the criteria used to select cells for refinement.

#### Cell Refinement Algorithm

The refinement of a cell is performed by splitting the cell into 8 new cells, each belonging to the next refinement level. To prevent large size disparities between adjacent cells, a $2:1$ balance constraint is enforced during the refinement process, as illustrated in Figure B.1. This constraint mandates that the refinement levels of any two adjacent cells differ by at most one. To satisfy this requirement, when a cell is marked for refinement, any adjacent cells at a coarser refinement level must first be recursively refined until the $2:1$ balance is achieved before the target cell itself is refined.




(a) 2:1 balance constraint.




(b) Refinement with grid balancing.

Figure B.1 Illustration of the 2 : 1 balance constraint in the cell refinement process.

In uniform grids, identifying adjacent cells is straightforward. However, in hierarchical grids, adjacent cells may exist at different refinement levels. To manage this and facilitate neighbor finding (which is essential for the tagging process), a level difference variable $\Delta^m$ is defined for each of the six face directions ($m \in \{\pm x, \pm y, \pm z\}$) of a cell. The variable $\Delta^m$ represents the difference between the cell's level and its neighbor's level in direction $m$ and can take three values: $-1$, 0, or 1, indicating that the neighbor in direction $m$ is finer, at the same level, or coarser, respectively. Table B.1 illustrates how $\Delta^m$ is used to identify neighbors for a cell $\mathrm{C}^n_{i,j,k}$ in the $+x$ direction. Similar logic applies to the other five directions. Each cell stores these six $\Delta^m$ values.

Algorithm 2 outlines the recursive cell refinement process, ensuring the 2 : 1 balance constraint is maintained.

Table B.1 Neighbor identification using the level difference $\Delta^m$ in the $+x$ direction for cell $C^n_{i,j,k}$.

| $\Delta^{+x}$ | **Number of Neighbor** | **Neighbor Level** | **Neighbor Local Indices** |
|---|---|---|---|
| 0 | 1 | $n$ | $(i+1, j, k)$ |
| 1 | 1 | $n-1$ | $(\lfloor i/2 \rfloor + 1, \lfloor j/2 \rfloor, \lfloor k/2 \rfloor)$ |
| -1 | 4 | $n+1$ | $(2i+2, 2j+\{0,1\}, 2k+\{0,1\})$ |

**Algorithm 2** Recursive Cell Refinement Process

**function** RefineCell(cell $C^n_{i,j,k}$)
  Get the level difference $\Delta^m$ of the cell $C^n_{i,j,k}$ in each direction $m$
  **for** $m \in \{\pm x, \pm y, \pm z\}$ **do**
    **if** $\Delta^m = 1$ **then** ▷ The adjacent cell is at a coarser level
      Recursively refine coarser neighbor
    **end if**
  **end for**
  **for** $m \in \{\pm x, \pm y, \pm z\}$ **do** ▷ Adjust the level difference of the adjacent cells
    **if** $\Delta^m = 0$ **then** ▷ The refined cell is at the same level
      Set the level difference of the adjacent cell at the opposite direction to -1
    **else if** $\Delta^m = -1$ **then** ▷ The refined cell is at a coarser level
      Set the level difference of the adjacent cell at the opposite direction to 0
    **end if**
  **end for**
  Create the 8 new child of the cell $C^n_{i,j,k}$
  Delete the old cell $C^n_{i,j,k}$
**end function**

### B.1.3 Grid Refinement Criteria

The selection of cells for refinement is based on geometric criteria. Several approaches can be employed for grid refinement; three representative criteria are presented in this section:

- **Criterion 1:** Topological entity count.
- **Criterion 2:** Minimum refinement level for intersected cells.
- **Criterion 3:** Disjoint surfaces proximity.

The first criterion, being the most straightforward, is based on the number of topological entities contained within a cell. When the number of Points, Edges (of Elements), or Elements in a cell exceeds a specified threshold, the cell is refined. This criterion is computationally efficient to implement and provides a quick way to adapt the grid density to the local complexity of the topology. To prevent excessive refinement, a maximum refinement level is established for this and all other criteria.

The second criterion focuses on ensuring that intersected cells achieve at least a specified minimum refinement level. This criterion is particularly valuable for maintaining adequate resolution near boundaries, ensuring that the grid is sufficiently refined in the vicinity of the discrete topology. If the intersected cells are not adequately refined by other criteria, this criterion ensures they reach the desired level of refinement.

The third criterion addresses the proximity of disjoint surfaces, refining the grid in areas where distinct surfaces are closely spaced to ensure the solver can accurately capture solutions in these regions. Figure B.2 illustrates this criterion. The algorithm checks if the set of elements intersecting a cell forms a single connected component within that cell. If multiple disjoint components exist, the cell is refined. This helps ensure sufficient resolution in narrow gaps or channels between different parts of the boundary. This criterion can be supplemented with additional buffer layers of refined cells around such regions to ensure sufficient resolution between closely spaced boundaries.

Additional criteria can be based on intersection patterns with specific topological entities. For instance, cells intersected by particular types of topological entities (e.g., a specific Face, Shell, or Volume) can be selectively refined.

## B.2 Intersections Computation

This section describes the methodology for computing intersections between the grid cells and the discrete topology. Since refinement criteria often depend on intersection information,

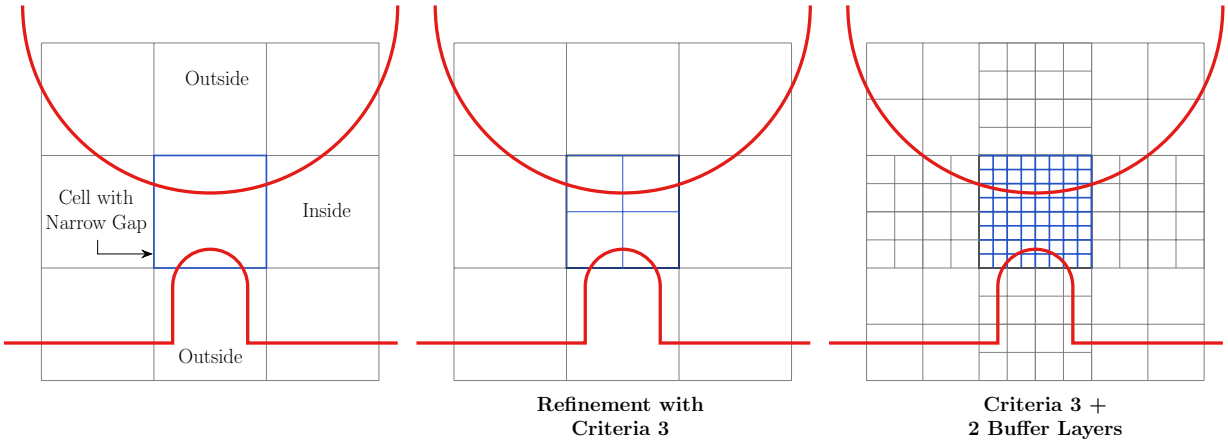



Figure B.2 Disjoint surfaces proximity refinement criterion.

intersection data must be available prior to the refinement process. Therefore, once the initial grid is created, intersections are computed before refinement begins, with each intersected cell storing a list of topology elements that intersect it. When an intersected cell is refined, the intersecting elements are transferred to the appropriate child cells. The process involves two main stages: initial intersection computation on the uniform grid and subsequent transfer of intersection information to the hierarchical grid during refinement. The following two subsections describe the methodology developed for these two stages.

### B.2.1 Uniform Grid Intersection

For each Element (triangle) of the topology, the algorithm identifies all grid cells that intersect it. The process begins by converting the coordinates of the Element's vertices from the topology coordinate system to the grid coordinate system (integer coordinates) using Equation B.4.

$$P_t(x,y,z) \to P_g(\tilde{i},\tilde{j},\tilde{k}) \quad \tilde{i} = \frac{x - x^g_{\min}}{\Delta x^0}, \quad \tilde{j} = \frac{y - y^g_{\min}}{\Delta y^0}, \quad \tilde{k} = \frac{z - z^g_{\min}}{\Delta z^0} \tag{B.4}$$

With only the Element's vertices expressed in grid coordinates as input, a pure function computes the intersected grid cells through the following steps:

- Compute the axis-aligned bounding box of the Element in the grid coordinate system.
- Identify grid planes (planes defined by constant integers i, j, or k indices) that intersect the element's bounding box.

- For each intersecting grid plane, compute intersection points between the Element and the plane, generating a line segment. Efficient closed-form formulas derived from barycentric coordinates are employed to compute these line segments.
- Project each line segment onto its corresponding grid plane and identify grid cells that intersect the projected segment.

The algorithm properly handles various degenerate cases, including Elements entirely contained within a single cell, Elements parallel to grid planes, and intersections that yield a point rather than a line segment.

### B.2.2 Transfer to Hierarchical Grid

While the uniform grid intersection algorithm could be reapplied to transfer intersections from a refined parent cell to its child cells, a simpler approach is employed. Since a refined cell generates exactly eight child cells and the intersections with the parent cell are already known, collision detection algorithms are used to determine which child cells intersect each element from the parent's intersection list. The algorithm implements the Separating Axis Theorem (SAT) to determine whether a three-dimensional triangle intersects a three-dimensional box. According to the SAT, two convex shapes do not intersect if there exists a plane (defined by an axis normal to it) that separates them. For a triangle-box intersection test, 13 potential separating axes must be checked to conclusively determine intersection status [136, 137]. If none of these axes yield a separating plane, the shapes intersect. This algorithm efficiently transfers intersection lists from parent to child cells during refinement.

This concludes the description of the intersection computation algorithms. The next section addresses the final step of the mesh generation process which corresponds to the grid cells tagging.

## B.3 Grid Tagging

After grid generation, refinement, and intersection computation, the final step is to tag each grid cell according to its position relative to the discrete topology. Three primary categories are defined: inside, outside, and intersected. The intersected cells are further subdivided based on whether their geometric centers lie inside or outside the topology. For non-intersected cells, the position of the cell center determines its classification. A naive approach would involve examining every cell and determining whether its geometric center lies inside or outside the topology using a point-in-polyhedron (PIP) algorithm. The PIP

algorithm functions by casting a ray from the point in a random direction and counting intersections with topology elements; an odd number of intersections indicates the point is inside, while an even number indicates it is outside. However, performing a global PIP test (considering all topology elements) for every cell is computationally prohibitive, especially for large topologies and highly refined grids. A more efficient approach leverages the fact that the set of intersected cells forms a watertight boundary layer separating the grid into inside and outside regions. The algorithm proceeds as follows:

1. First, identify all cells intersected by the topology (already determined during intersection computation).
2. Select a non-intersected cell adjacent to an intersected cell.
3. Apply the PIP algorithm to determine whether this adjacent cell is inside or outside the topology.
4. Employ a flood-fill algorithm to propagate this classification to all reachable non-intersected neighbor cells.
5. For geometries with narrow gaps, the flood-fill algorithm may get stuck. In such cases, the algorithm restarts with another adjacent cell to continue the classification process.
6. Once all cells adjacent to intersected cells are classified, all cells inside the topology are guaranteed to be tagged as "inside," and all remaining cells are tagged as "outside."

To further enhance computational efficiency, a local version of the PIP algorithm illustrated in Figure B.3 has been developed that considers only the elements intersecting a given cell when classifying adjacent cells. This algorithm calculates the nearest point on intersecting elements to the test point and uses the element's normal vector to determine whether the point is inside or outside. While efficient, this local PIP approach occasionally fails (e.g., when the nearest point falls on an element edge), in which case the algorithm reverts to the global PIP method as a fallback.

After classifying inside and outside cells, the intersected cells themselves are classified using another local PIP algorithm illustrated in Figure B.4. This approach is more robust than the one used for adjacent cells, as it casts rays from the intersected cell's center to random points on shared faces with already-tagged cells. The parity of intersections and the shared faces status determines whether the cell center is inside or outside the topology.

Figures B.5 and B.6 present an example of a grid generation process applied to a simple discrete topology of a three-dimensional volume featuring two cylindrical holes. The discrete

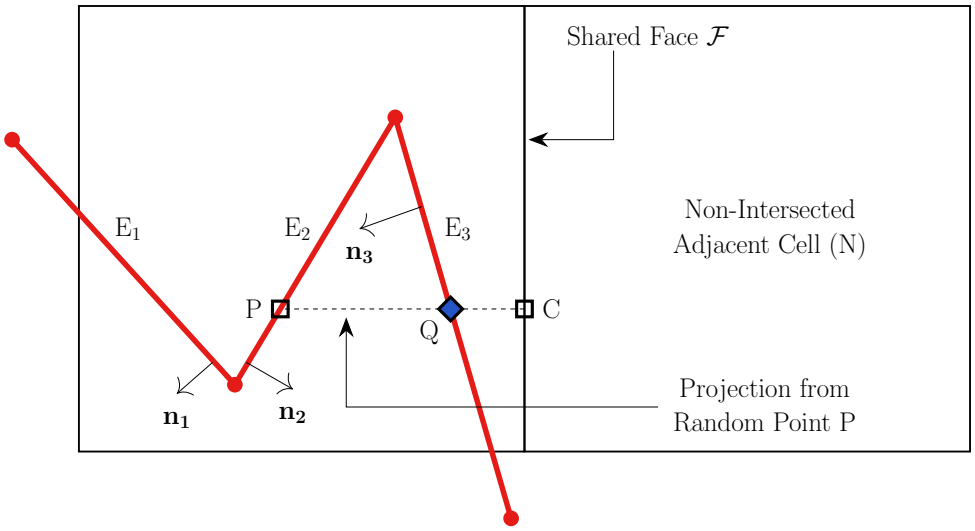



Figure B.3 Local PIP algorithm for adjacent cells classification. Step 1: Select a random point (within the intersected cell) and located in one of the intersected elements (Point P of Element 2). Step 2: Project the point P onto the shared face (Point C) Step 3: Compute the nearest point in the ray CP (Point Q) on the intersected elements. Step 4: Determine the status of the adjacent cell using the normal vector $\mathbf{n_3}$ and the vector CQ.

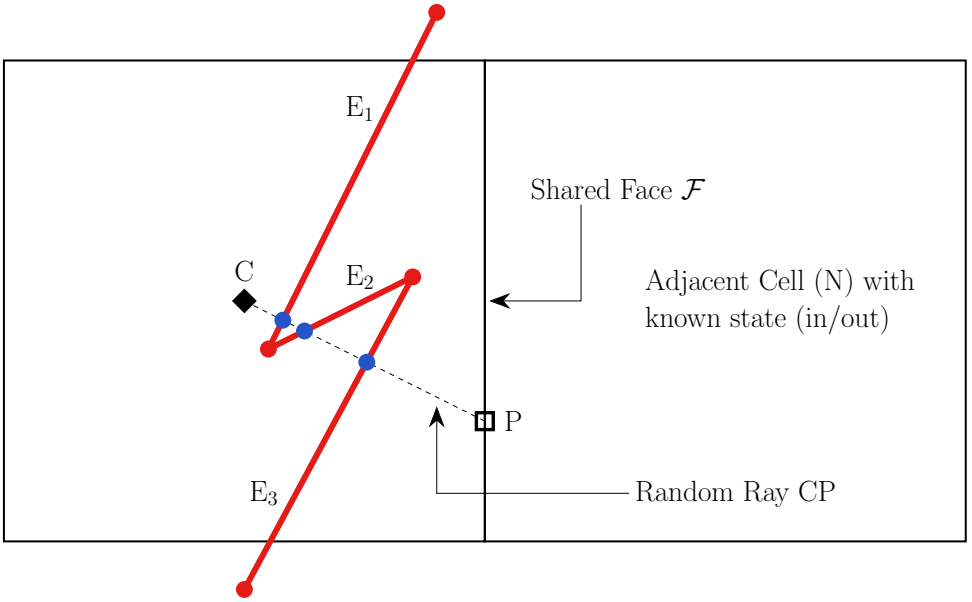
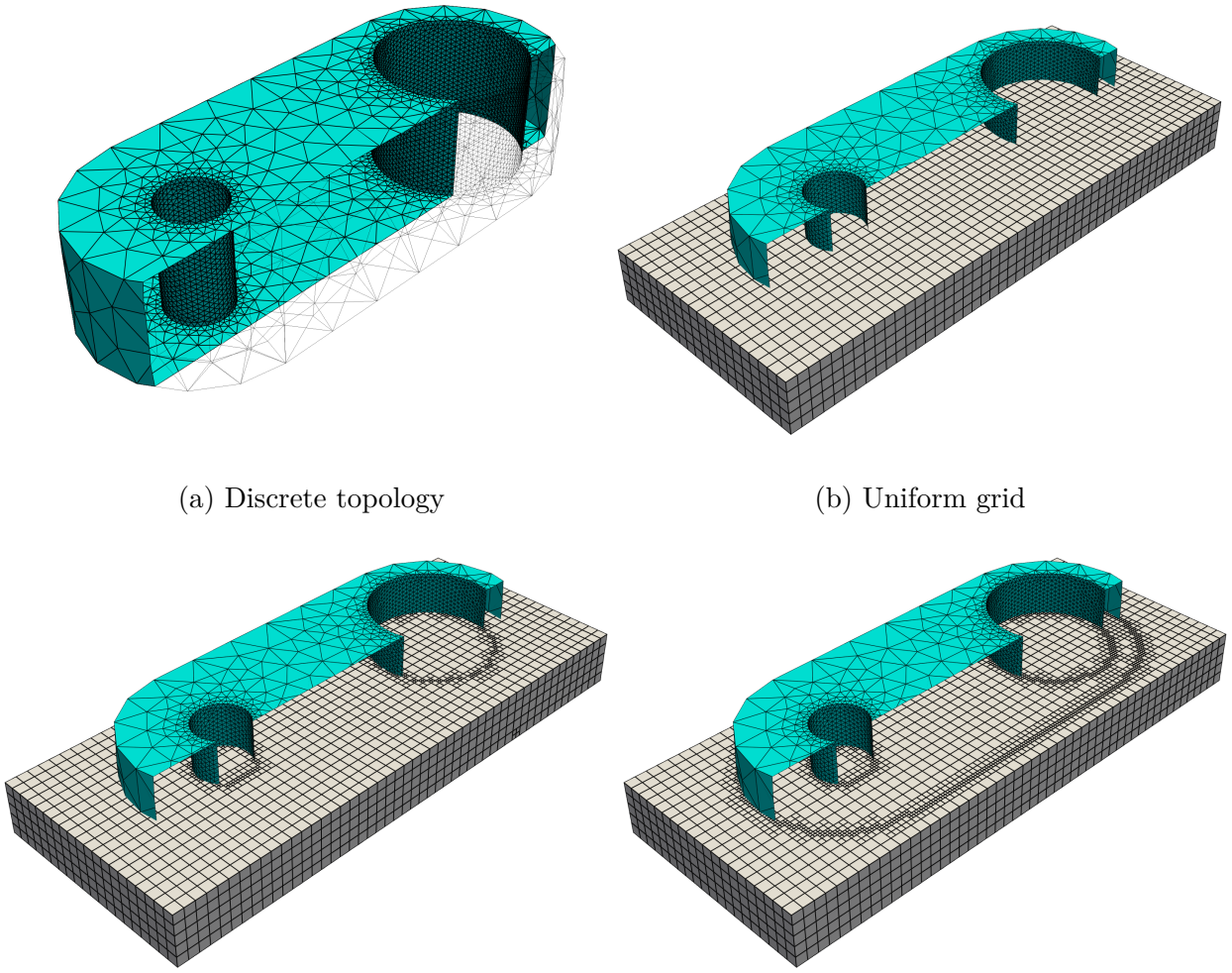



Figure B.4 Local PIP algorithm for intersected cells classification. Step 1: Select a random point on the shared face of the intersected cell. Step 2: Cast a ray from the center of the intersected cell to the random point. Step 3: Count the number of intersections with the topology elements. Step 4: Determine the status intersected cell using the parity of the number of intersections and the adjacent cell status.

topology and the uniform grid generated around it are shown in Figure B.5a and B.5b, respectively. In Figure B.5c and B.5d, the grid is refined based on the first and second refinement criteria, respectively. For the first criterion, the refinement is based on the number of intersected elements in each cell, and as a result, the grid is refined only in the vicinity of the holes where the topology contains more elements. For the second criterion, the refinement is based on the minimum refinement level for intersected cells, and consequently, all boundary cells are refined to the same level. Finally, Figure B.6 shows the final grid after applying the tagging process. Grid cells are colored according to their classification: inside (green), outside (gray), and intersected (magenta).



(a) Discrete topology

(b) Uniform grid

(c) Refinement using Criterion 1

(d) Refinement using Criterion 2

Figure B.5 Grid generation process for a simple topology (1/2).

Figures B.7 and B.8 further demonstrate the application of the grid generation process,

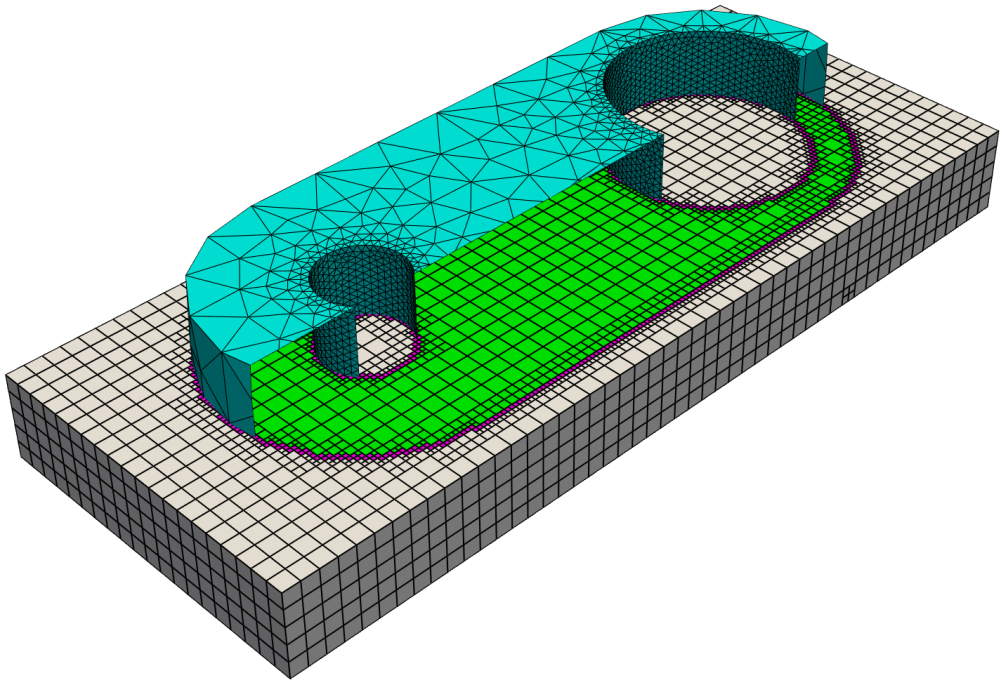


Figure B.6 Final tagged grid for the simple topology (2/2).

this time to the more complex discrete topology representing the simplified high-voltage circuit breaker previously shown in Figure A.3. These figures illustrate the capability of the Cartesian mesh generator to handle intricate geometries.

**Summary**

This appendix has presented a comprehensive methodology for generating adaptive Cartesian grids around discrete topologies. The grid is constructed using a hierarchical Cartesian structure with refinement based on geometric criteria. Intersections between the grid and topology are computed efficiently using both uniform grid intersection algorithms and specialized transfer methods for hierarchical refinement. The final tagging process employs optimized point-in-polyhedron tests and flood-fill techniques to classify cells as inside, outside, or intersected relative to the topology. This approach enables efficient generation of Cartesian grids suitable for complex geometries within reasonable computational time.

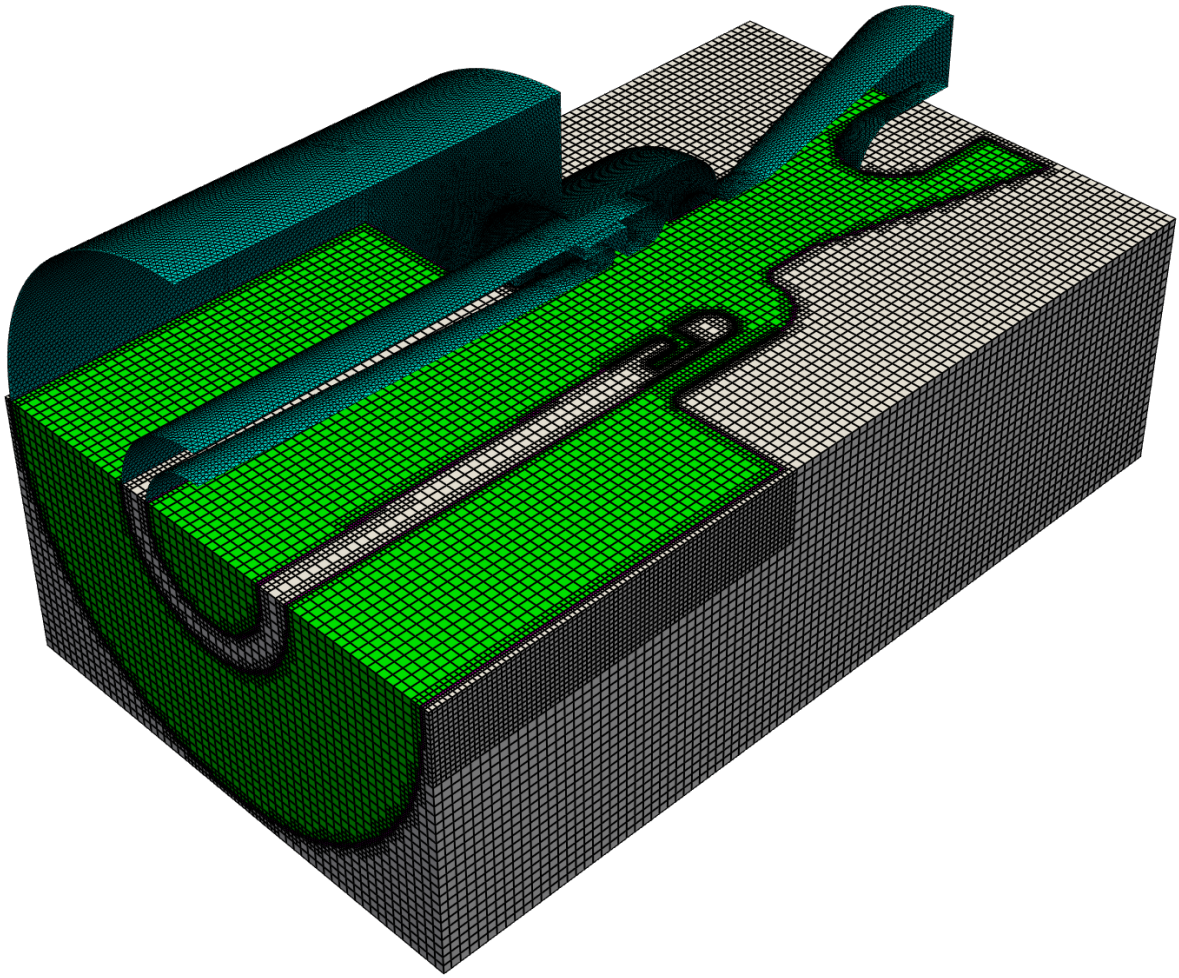


Figure B.7 Tagged grid for the simplified high-voltage circuit breaker topology (cf. Figure A.3). (1/2).

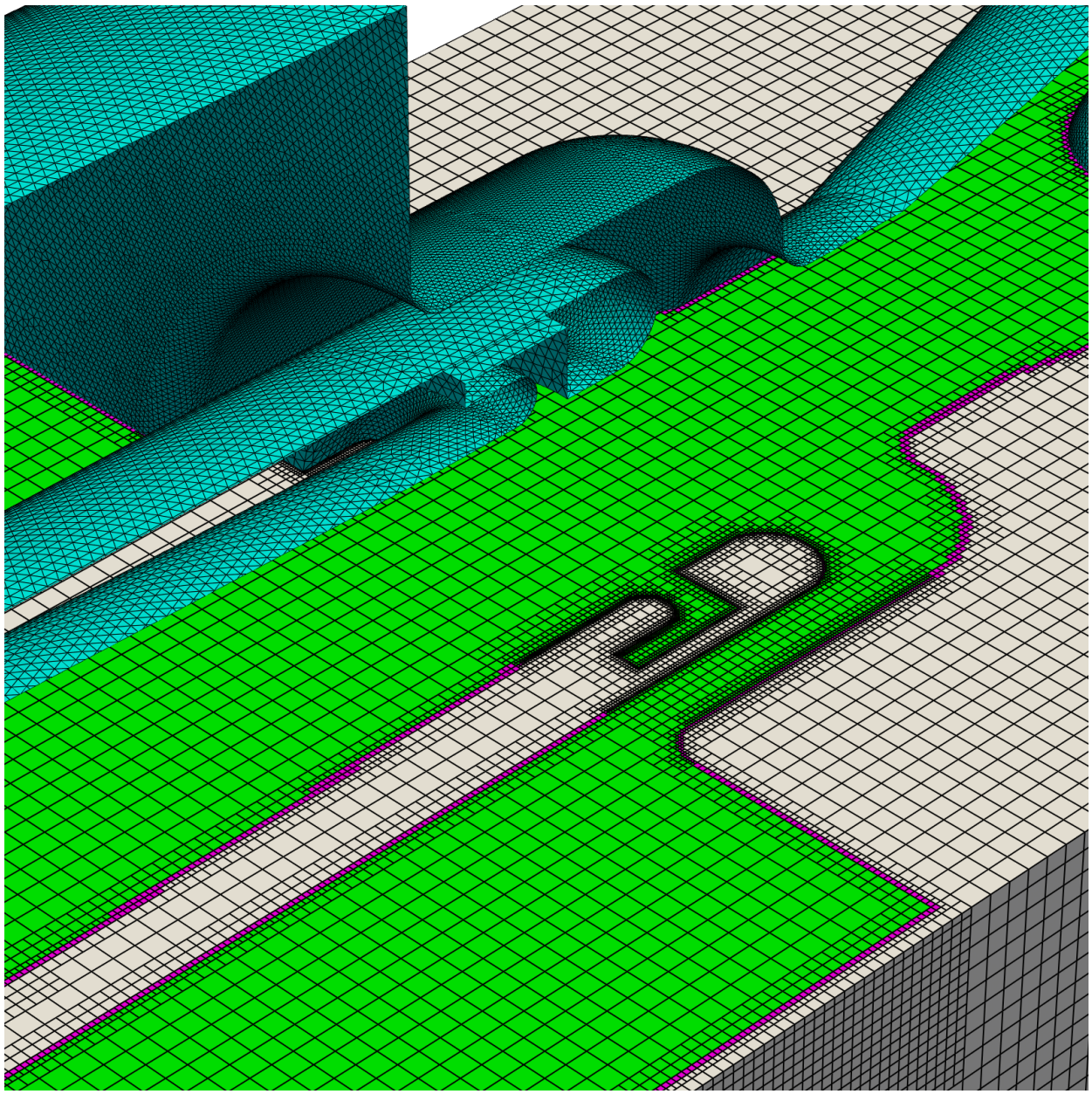


Figure B.8 Tagged grid for the simplified high-voltage circuit breaker topology (cf. Figure A.3). (2/2).