National Technical University of Athens
School of Mechanical Engineering
Fluids Section
Laboratory of Thermal Turbomachines
Parallel CFD & Optimization Unit

The Cut-Cell Method for the Prediction of 2D/3D Flows in Complex Geometries and the Adjoint-Based Shape Optimization

Ph.D. Thesis

Konstantinos D. Samouchos

Supervisor: Kyriakos C. Giannakoglou
Professor NTUA

Athens, 2022

National Technical University of Athens
School of Mechanical Engineering
Fluids Section
Laboratory of Thermal Turbomachines
Parallel CFD & Optimization Unit

The cut-cell method for the prediction of 2D/3D flows in complex geometries and the adjoint-based shape optimization

PhD Thesis

Konstantinos D. Samouchos

Athens, 2022

Accompanied by my deepest gratitude and love,
I dedicate this dissertation to
my mother who has never left my side,
my father who showed me the roughed path of morality,
my brother who has brightened my days, and
my grandparents who taught me the virtues of a balanced life.
They will always be the guiding light of my life and
an inexhaustible source of inspiration.

Acknowledgments

I would like to express my gratitude to the following people who have significantly contributed to the completion of this dissertation. Firstly, I would like to thank my supervisor, Pr. Kyriakos Giannakoglou, for introducing me to the world of Computational Fluid Dynamics and Adjoint-Based Optimization. I'm also thankful for his meaningful recommendations on the thesis text and presentation.

I am also deeply grateful to the members of the research group of the Parallel CFD & Optimization Unit (PCOpot), NTUA without the help of whom I would have never completed this thesis. More specifically, I would like to sincerely thank Dr. Xenofon Trompoukis, who shared important information about advanced numerical methods and their application to practical cases. Moreover, I'm more than appreciative to Dr. Konstantinos Tsiakas, who taught me the essentials of CFD as well as techniques for efficient scientific programming. Words are not enough to thank Dr. Evangelos Papoutsis-Kiachagias for sharing valuable details about the mathematical formulation of the continuous and discrete adjoint theory. Furthermore, I feel honored collaborating with Panagiotis-Giannis Vrionis, who applied and further improved the cut-cell method developed in this dissertation.

I am also profoundly grateful to the rest, current or former, members of the PCOpt family, namely Dr. Christos Kapellos, Dr. Flavio Gagliardi, Dr. Ioannis Kavvadias, Dr. Dimitrios Kapsoulis, Dr. Konstantinos Gkaragkounis, Dr. Morteza Monfaredi, Dr. Varvara Asouti, James Koch, Christos Veziris, Themistoklis Skamagkis, Andreas Margetis, Mehdi Ghavami Nejad and Ioannis Trompoukis. They were always supportive and willing to help with any problem I was facing, scientific or not. Last but not least, I am honestly thankful to Kimon Velitzanidis, Ioannis Stasinopoulos, Panagiotis Meletis, Konstantinos Zarnaris, and Konstantinos Boudounis. Their genuine friendship accompanied me in good and bad times, giving me the necessary encouragement to continue working on this thesis.

All of old. Nothing else ever. Ever tried. Ever failed.
No matter. Try again. Fail again. Fail better.
[...]
All of old. Nothing else ever. But never so failed.
Worse failed. With care never worse failed.

Wortsward Ho, 1983
Samuel Beckett

You Can Never Hold Back Spring, 2005
Tom Waits & Kathleen Brennan

Matrix, 2020
Marcel Caram

Abstract

This dissertation thesis develops integrated, robust, and reliable Computational Fluid Dynamics (CFD) methods and software for the analysis and shape optimization in real-world applications in fluid mechanics and aerodynamics. To this end, the cut-cell method, which removes mesh generation barriers from the flow analysis and design process is adopted. The computational domain is firstly covered with a Cartesian mesh and then parts occupied by the solid bodies are discarded, giving rise to the cut-cell mesh. The benefits of this method are profound in fluid problems with moving solid bodies which are allowed to move upon the stationary background mesh, avoiding the use of mesh deformation tools. Moreover, contrary to body-conforming approaches, the changes in shape during an optimization loop do not affect the surrounding mesh, preventing mesh generation failure and the premature breakdown of the optimization loop. Therefore, this dissertation thesis exploits these beneficial features and develops a cut-cell-based flow solver and shape optimization tool for compressible and incompressible flow problems.

Firstly, a fast and automated mesh generation method with low memory requirements is developed, which guarantees smooth mesh refinement close to solid boundaries and flow features that require higher mesh resolution. Cells intersected by the geometry get rid of their solid part by giving rise to the so-called cut-cells. New algorithms are proposed for computing their topological characteristics needed by the flow-solver and post-processor. Numerical instabilities caused by the presence of small cut-cells adjacent to much larger ones are avoided by cell-merging, according to which small cell fragments are geometrically merged with their neighbors. Furthermore, algorithms for fast neighbor detection, mesh connectivity computation, and mesh-partitioning are also developed and used.

Then, compressible and incompressible flow solvers are developed, the latter being based on the artificial compressibility method, to numerically solve the (U)RANS equations. The presented numerical scheme takes advantage of the Cartesian mesh structure and uses a cell-centered, finite volume approach employing the MUSCL scheme and the approximate Riemann solver of Roe for the convection terms. In applications concerning moving geometries, the mesh is continuously adapted to their motion, employing local refining and coarsening operations. Strategies to accurately extrapolate the current flow solution to the mesh of the next time step are presented. Additionally, a novel method to impose the flow conservation laws even in large geometry displacements is developed by performing a cell clustering algorithm which properly treats the sudden change in cells' status from solid to fluid and vice-versa.

The resulting software is parallelized using the Open MPI protocol and assessed in a series of tests concerning internal and external, inviscid and laminar flows. Moreover, comparisons with data provided by conventional body-conforming approaches indicate its ability to deliver flow solutions of the same accuracy. The method's effectiveness is demonstrated in several challenging applications of practical interest. Among other, the flow simulation in a scroll machine, which is quite rare in the literature due to its high complexity, is presented. Another application concerns the flow inside a valveless diaphragm micropump, where the mass conservation is successfully imposed despite the intensive deformation of the diaphragm. Finally, the flow within an Electrical Submersible Pump (ESP) stage is studied, introducing the cut-cell method as an alternative to address the rotor-stator interaction problem.

In the field of gradient-based shape optimization, the continuous and discrete adjoint approaches are developed, programmed, and used. These methods compute the gradient of the objective function at a cost, which is independent of the number of design variables, providing a viable tool for industrial design processes. Their mathematical formulation, software development, and implementation in cut-cell meshes for viscous and unsteady flows are presented for the first time in the literature. Concerning the continuous approach, different discretization schemes for the adjoint Partial Differential Equations (PDEs) are investigated, resulting in adjoint schemes which are equivalent to the FVS, HLLC, and Roe's approximate Riemann primal solvers.

Moreover, a discrete adjoint software is developed by accurately hand-differentiating both the compressible and incompressible flow cut-cell solvers. Particular emphasis is laid on properly treating the discrete adjoint time integration by differentiating algorithms dealing with flow field extrapolation between meshes of subsequent time steps. Furthermore, the adjoint cut-cell software is verified and applied to industrial optimization problems, such as the total pressure losses minimization of a duct, the lift maximization of a wing, and the outlet tangential velocity minimization of the Electrical Submersible Pump stage. Finally, the multi-objective optimization under uncertainties of the diaphragm micropump is carried out. In all cases, solutions of adequately improved performance are delivered, confirming the effectiveness of the developed method and software.

Keywords: Navier-Stokes Equations, Computational Fluid Dynamics, Cut-Cell Method, Compressible Flow, Incompressible Flow, Unsteady Flow, Shape Optimization, Continuous Adjoint Method, Discrete Adjoint Method

Abbreviations

ALE	Arbitrary Lagrangian-Eulerian
CAD	Computer-Aided Design
CFD	Computational Fluid Dynamics
CG	Conjugate Gradient
CSAMR	Cell-based Structured Adaptive Mesh Refinement
DNS	Direct Numerical Simulation
EA	Evolutionary Algorithm
EASY	Evolutionary Algorithms SYstem
ESP	Electrical Submersible Pump
FDs	Finite Differences
FVS	Flux Vector Splitting
GB	Gradient-Based
GCL	Geometric Conservation Law
HLLC	Harten-Lax-van Leer-Contact
IBM	Immersed Boundary Method
LES	Large Eddy Simulation
l.h.s.	Left-Hand Side
LTT	Lab of Thermal Turbomachines
MAEA	Metamodel-Assisted EA
NTUA	National Technical University of Athens
PAD	Pareto Advancement Direction
PCA	Principal Component Analysis
PCE	Polynomial Chaos Expansion
PCOpt	Parallel CFD & Optimization Unit
PDE	Partial Differential Equation
PGD	Proper Orthogonal Decomposition
RANS	Reynolds-Averaged Navier-Stokes
r.h.s.	Right-Hand Side
SDF	Signed Distance Function
SD(s)	Sensitivity Derivative(s)
STL	Standard Triangle Language
SVD	Singular Value Decomposition
(U)RANS	Unsteady RANS
w.r.t.	with respect to

Contents

Chapter 1

Introduction

Over the last decades, the exponential growth of computational power coupled with the maturity of numerical methods has made Computational Fluid Dynamics (CFD) an indispensable and cost-effective tool for analysis and product design in numerous engineering fields. CFD is implemented in a wide range of problems from aeronautical, aerospace, and automotive applications to the weather forecast and biomedical technology. Moreover, its accessibility and ability to test various combinations of different geometries and flow conditions in relatively low turnaround time makes it a valuable tool, especially for sensitivity analysis, optimization, and preliminary design.

Therefore, accurate flow computations around complex geometries, usually associated with practical applications, are highly valued. In addition, the increased interest in the aerodynamic or hydrodynamic analysis of moving bodies requires efficient and reliable tools to deal with this challenging problem. To this end, fast and robust mesh generation is a vital prerequisite, which, however, is still a strenuous, costly, and not fully automated task.

A remedy to this issue is to adopt the Immersed Boundary Methods (IBMs), which remove the mesh generation bottleneck from the evaluation and design process. Amongst different approaches constituting this vast scientific spectrum, the cut-cell method holds a central position due to its increased accuracy and reliability. Its implementation to the flow analysis and optimization process in practical applications defines the two central axes of this dissertation. In particular, the first axis is concerned with original techniques for mesh generation and flow simulation around complex moving geometries. An introduction to these methods accompanied by the corresponding literature survey is presented in section 1.1. The second axis deals with the development of efficient optimization tools assisted by the continuous and discrete adjoint formulations on the cut-cell method. An introductory examination of these approaches and their effective performance in real-world problems is given in section 1.2. Finally, the thesis outline is described in section 1.3.

1.1 The Immersed Boundary Methods

Contrary to the conventional concept of body-conforming meshes, IBMs employ meshes that do not necessarily fit the geometry's boundary. This concept was first introduced by Peskin [244] in 1972, and since then, the field of immersed boundary methods has flourished with the development of a wide variety of different approaches. The general idea behind these methods is subdividing the computational domain into rectangular hexahedra constituting a Cartesian mesh extended through the geometry's boundary to the fluid and solid regions. Thus, the volume mesh generation is decoupled from the complexity of the solid surface description. Moreover, in cases including moving bodies, the immersed geometry is allowed to move upon the background stationary Cartesian mesh, avoiding the mesh deformation or regeneration, which is common in body-conforming mesh approaches. Fig. 1.2 compares a body-conforming unstructured mesh around an isolated airfoil, fig. 1.2a, and a Cartesian mesh around the same geometry, fig. 1.2b.

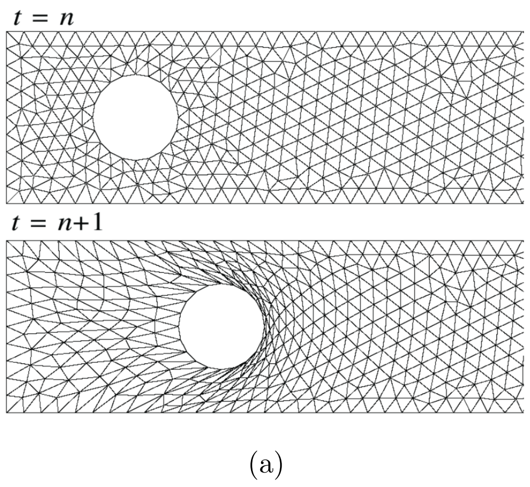
These methods have proven remarkably useful in flow simulation around complex stationary or moving geometries performing large displacements. Section 1.1.1 compares IBMs with other alternatives and indicates their advantages and drawbacks. From that point of view, using Cartesian meshes transforms the problem of conforming the mesh to the boundary into the imposition of the flow conditions at the wall, which requires modifications to the discretization scheme of the governing equations close to the immersed boundary. The proper boundary treatment gives rise to various approaches presented in brief in section 1.1.2. Then, section 1.1.3 focuses on a subclass of IBMs called the cut-cell method providing a broader context in which the present work is placed.

1.1.1 Mesh Generation Methods

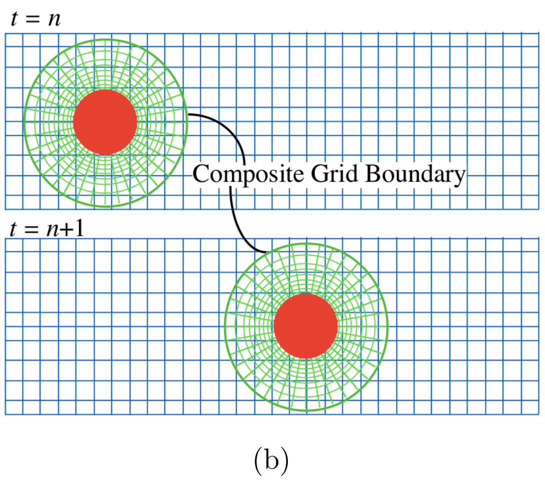
As an introduction to IBMs, a comparison is presented between this approach and two alternatives dealing with complex moving geometries. These are structured or unstructured deforming, body-conforming meshes, and composite overlapping mesh approaches [33], [199], [8]. Unstructured or structured multi-block mesh methods [305] appear extremely powerful for arbitrarily complex geometries. In moving boundary problems, the volume mesh should follow the surface motion deforming its elements [185], [23]. However, this process may damage the mesh quality, deteriorating the flow solution's accuracy, fig. 1.1a. In addition, mesh deformation fails to proceed after large boundary displacements, and re-meshing is unavoidable, requiring the user's intervention. Then, interpolation of the flow solution to the new mesh is needed, which is far from trivial.

On the other hand, in overlapping-mesh or overset approaches, individual separately generated meshes overlay a background mesh, fig. 1.1b. In cases including moving boundaries, the associated mesh follows the geometry's motion, while the rest remain stationary, avoiding the mesh deformation. During the flow solution, information is interpolated between meshes. Nevertheless, overset methods cannot guarantee conservation across composite mesh boundaries without sophisticated geometric constructions in the overlap region. Moreover, the difficulties of mesh generation around complex geometries remain untreated.

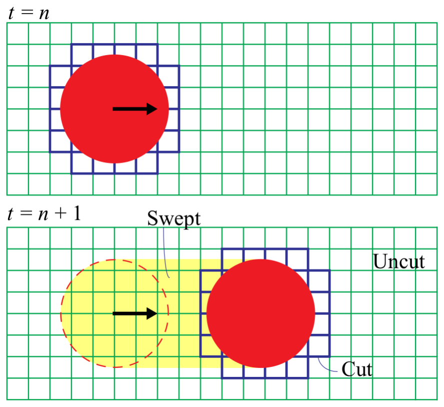
Contrary to the before-mentioned approaches, stationary Cartesian meshes considerably simplify and automate the mesh generation process, fig. 1.1c. However, implementing a discretization scheme that maintains conservation is not straightforward. Moreover, the mesh resolution is expected to be higher in a Cartesian mesh than in a body-fitted one to maintain the same level of accuracy in the flow solution, resulting in higher computational and memory requirements. Therefore, the formulation of a highly accurate IBM, capable of maintaining conservation even for gross boundary motions while retaining its robustness, is among the targets of this thesis.



(a)



(b)



(c)

Figure 1.1: (a) A body-conforming unstructured mesh is deformed following the motion of a cylinder. Invalid elements appear at time step n+1 damaging the mesh quality. (b) A body-conforming mesh (green) is moving upon a background stationary mesh (blue), depicting the overset approach. Special treatment is needed for imposing conservation across the composite mesh boundary. (c) A cylinder is displaced upon a stationary background Cartesian mesh over two successive time steps. Covered and uncovered cells lying within the swept region require additional numerical manipulations to prevent spurious sinks and sources that harm the accuracy of the flow simulation. Figures drawn from [219].

1.1.2 Literature Survey

This section aims to present a short overview of the broad field of IBMs, highlighting the wide variety of available methods and identifying some of the most effective approaches. The interested reader may find extensive reviews in Mittal et al. [209], Sotiropoulos et al. [289], and Maxey [198].

IBMs can be classified regarding a variety of criteria. Among others, Sotiropoulos et al. [289] introduced a taxonomy based on the representation of the immersed geometry, dividing the field into diffused and sharp interface methods. Diffused interface methods employ a blur representation of the immersed boundary, the effect of which is introduced implicitly over a narrow zone of computational cells. In contrast, sharp interface methods use a crisp representation of the fluid-solid interface throughout the flow simulation, avoiding its spatial smearing over a range of mesh cells.

Diffused interface methods can further be subdivided into continuous and discrete forcing approaches [35]. The first approach exerts an artificial volume force employed in the vicinity of the interface to effectively represent the effect of the immersed boundary on the flow field. The forcing is expressed as an additional term in the continuous equations and, therefore, is independent of the chosen discretization. The continuous forcing approaches have been applied to flows around elastic and rigid boundaries. Such examples are given below.

The original work of Peskin [244], [245], who introduced the IBMs, belongs to the first class of the continuous forcing approaches. In this method, blood flows within a beating heart upon a stationary Cartesian mesh. The heart's muscle contraction is represented by elastic fibers consisting of massless points moving along with the flow in a Lagrangian manner. A forcing term added to the momentum equations encapsulates the effect of the fibers' motion to the fluid. In particular, a smoothed Dirac function distributes the force of each Lagrangian point to the surrounding cells. The correct choice of the smoothed function is of paramount importance, and it has been studied by numerous researchers giving rise to various distributions [44], [270], [172]. Second-order accurate methods of this kind have been developed by Lai et al. [172], Griffith et al. [117], and others.

However, employing this method in cases concerning rigid bodies causes numerical instabilities. An approach to surpass this issue is by using highly stiff springs that approximate the rigid behavior of the body [44], [172]. Other approaches solve the governing equations by imposing a rigidity constraint to the fluid inside the solid region of the mesh [107], [240], [19] or assuming that the entire flow occurs in a porous medium [20], [161]. The exerted spring force and the porosity assumption can be considered subclasses of the more generic force formulation introduced by Goldstein et al. [111].

On the other hand, in the discrete forcing approach, the forcing is incorporated straight into the discretized governing equations. It can be explicit [316] or implicit [295] and allows for direct control over the solver's numerical accuracy and stability. The method was firstly introduced by Mohd-Yusof [213], who used the difference between the velocity at the mesh nodes and the desired velocity at the boundary to define the forcing term. Then, Fadlun et al. [89] extended this work by implementing the discrete forcing approach on a 3D marker-and-cell (MAC) staggered mesh. Further improvements concerning stability and accuracy were made by Balaras [24], Gilmanov et al. [105], Zhang et al. [344], and Choi et al. [62]. Finally, this approach has been implemented in various applications, including the turbulent flow inside an internal combustion engine [322].

The great advantage of the diffused interface methods is that the presence of the wall does not affect the equations' discretization scheme apart from the term expressing the distributed force. In particular, they avoid the complicated intersection computation of the immersed geometry with the mesh cells offering a method for straightforward software development. However, these methods are unable to predict boundary layers accurately. Therefore, the sharp interface approaches were introduced, emphasizing the direct imposition of the boundary conditions by modifying the discretization scheme to cells next to the wall. The two main representatives of this category are the ghost-cell and the cut-cell methods described below.

In the ghost-cell method, described by Tseng et al. [310], cells intersected by the boundary were explicitly detected, avoiding the artificial smearing of the flow field close to the wall. The method's principle is to repeatedly extrapolate the current solution to a zone of ghost cells adjacent to the wall and enforce the necessary boundary condition by modifying the flow variables stored at each ghost cell. Various interpolation or extrapolation schemes have been proposed, with the simplest one being the linear interpolation by Ferziger et al. [93]. However, the accuracy of this approach is questionable, especially in high Reynolds flows. A better alternative is employing linear interpolation in the tangential to the wall direction and a quadratic one along the normal direction [190]. Another technique proposed by Gibou et al. [102] slightly modifies the solid boundary, avoiding erroneous boundary layer predictions.

Several researchers have contributed to the method's improvement. For example, Mittal et al. [208] introduced the concept of image points to fulfill the divergence-free criterion. Moreover, Berthelsen et al. [43] presented a method to handle highly irregular boundaries constituting sharp corners or solid thin plates without losing accuracy. Furthermore, Pan et al. [234] and Gao et al. [100] used different techniques to improve the approach of Tseng et al. [310] by mitigating the instabilities caused in the extrapolation scheme when fluid nodes are very close to the boundary. Additionally, Shinn et al. [284] increased the discretization's accuracy at the boundaries by preserving the mass continuity for ghost cells on a staggered mesh. Recently, Grosse et al. [114] developed a second-order technique by adequately solving the Riemann problem at cells located on the boundary. Finally, the ghost fluid method, introduced by Fedkiw [91], [90], is worth noticing as well as its improvements concerning accuracy and robustness by Terashima et al. [304] and Liu et al. [184]. According to that, each cell in the computational domain is equipped with a ghost-cell being in contrast to the rest ghost-cell methods, where the ghost-cells are defined exclusively alongside the immersed boundary.

A significant drawback of the methods presented so far is that none guarantees the satisfaction of the flow conservation laws in the vicinity of the wall. On the other hand, the cut-cell method offers strict conservation of mass, momentum, and energy at each mesh cell, constituting a valuable alternative to the above approaches. The extensive study of this method has resulted in a wide field of different approaches, the most remarkable of which are presented in subsection 1.1.3.

1.1.3 The Cut-Cell Method

Over the last two decades, the cut-cell method has become increasingly popular as an alternative to simulate the flow around complex geometries. It is considered one of the most reliable IBMs since it accurately represents the fluid-structure interfaces. Consequently, it avoids the generation of spurious pressure fluctuations caused by violating the conservation laws, observed typically in other approaches like the ghost-cell method [218].

Contrary to the rest IBMs, the cut-cell method uses only the fluid part of the background mesh. In particular, the Cartesian structure of the mesh is retained in all but cells intersected by the immersed boundary. These cells discard their solid part to conform to the wall. The remaining fluid part of each cell constitutes a new polygon in 2D or polyhedron in 3D called cut-cell. An example of such a mesh is shown in fig. 1.2c. Therefore, the advantage of using a Cartesian mesh is preserved for the interior cells, and a more delicate treatment is needed only for the cut-cells.

In other words, the case-specific and challenging mesh generation around complex geometries is replaced by the general problem of constructing the cut-cells. Thus, this approach can be considered a consistent extension of the finite volume method, at least for stationary geometries, and as such, it guarantees the satisfaction of the conservation laws.

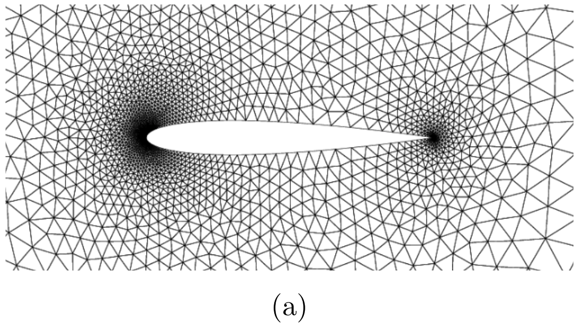
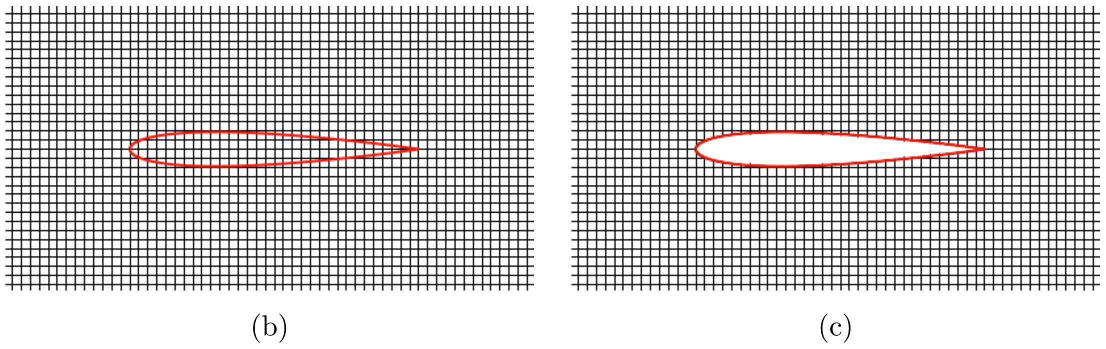



Figure 1.2: (a) A body-conforming unstructured mesh around an isolated airfoil. (b) The airfoil (red) is immersed into a Cartesian mesh (black). (c) The solid part of the Cartesian mesh has been discarded, giving rise to a cut-cell mesh type.

The concept of cut-cells was firstly proposed by Purvis et al. [250] in 1979 and then by Wedan et al. [331] in 1983. These authors applied a finite volume method on a cut-cell mesh to solve the fully nonlinear potential equation. Later, in 1986, Clarke et al. [65] extended this idea to the 2D Euler equations. In that work, each cut-cell was describing the geometry in a piecewise linear fashion. Gaffney et al. [99] used the same technique to solve the 3D Euler equations. At around the same period, Rubbert et al. [264] applied a cut-cell finite element method to discretize the 3D potential flow equation.

The above-mentioned pioneering works did not only establish the cut-cell method but also described its main drawbacks. A common implication is related to the reshaping of the intersected cells. In particular, the fluid part of these cells may become very small, harming the stability of the flow solver. Such an example is illustrated in fig. 1.3. First, Clarke et al. [65] resolved this problem by adopting an agglomeration technique, in which small cell fragments were incorporated into adjacent cells. Similarly, Ye et al. [340] suggested an approach in which cut-cells, with centroids located in the solid region, were absorbed by neighboring ones, generating new trapezoidal cells. Over the next years, several techniques have been proposed to tackle this issue while retaining the conservative nature of the finite volume method. Some of them are the cell-merging [53], [40], [134], [27], cell-linking [165], flux redistribution [70], [132], mesh reshaping [268], and the H-Box method [40].

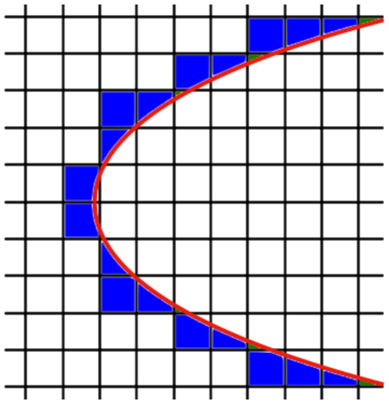


Figure 1.3: A Cartesian mesh (black) over an arbitrary embedded solid boundary (red). Regular cut-cells are colored blue, while 6 very small cut-cells are indicated with green.

In the late '80s, the cut-cell method gained popularity, and the first techniques for mesh adaptation appeared. Berger et al. [40] and Quirk [252] used a similar isotropic adaptive mesh refinement to capture strong shock waves accurately. Moreover, Pember et al. [242] applied a solution-based adaptation to solve the 3D Euler equations, and Melton et al. [201] developed a 3D Euler cut-cell method that incorporated a geometry-based mesh refinement technique.

The extension to viscous cases was made by Quirk [251]. However, the demonstration was brief and was applied to simple test cases. Then, Coirier et al. [69] used the cut-cell method to simulate the laminar flow around more complex 2D geometries. Moreover, Hartmann et al. [124] were the first to implement a fully conservative cut-cell-based method for 3D problems of compressible laminar flows. However, properly treating the viscous terms on irregularly-shaped cut-cells was quite challenging because high mesh resolution is needed to represent the developed boundary layer accurately. A remedy to this issue is the conjunction of the Cartesian approach with structured curvilinear meshes. In such methods, a body-fitted mesh is generated close to the immersed boundary, and the rest domain is filled with a Cartesian mesh intersected with the outer layer of the structured mesh [157], [330], [77]. Although these methods increase simulation's accuracy without excessively refining the mesh, their implementation in complex or moving geometries is quite delicate.

An attractive alternative is to employ the Cartesian mesh down to the wall and properly change the discretization scheme to cells close to the wall. Following this approach, Hartmann et al. [124] and Ji et al. [145] developed a second-order accurate discretization scheme. However, its accuracy decreased to nearly one in the vicinity of the wall. Moreover, Ye et al. [340] investigated a new interpolation scheme next to the immersed boundaries, capable of retaining second-order accuracy. In addition, a promising approach was presented by Berger et al. [36], who used quadratic polynomials to compute the flow variables and their derivatives stored in cut-cells. A similar strategy was proposed by Anagnostopoulos [11], who suggested polynomials of different degrees for each flow variable. Furthermore, other researchers alleviate the non-alignment of the mesh to the geometry by increasing the order of the discretization stencil [218], [167], [10]. Finally, the cut-cell method has been extended to turbulent flows in the Reynolds-averaged Navier–Stokes (RANS) [36], [37], the Large Eddy Simulation (LES) [204], [203], [56], or the Direct Numerical Simulation (DNS) [80] framework.

New challenges emerge introducing the cut-cell method to cases involving moving geometries. In particular, additional complexities arise due to the change in the Cartesian cells' nature from fluid to solid and vice versa caused by the boundary's displacement, fig. 1.1c. In such cases, Seo et al. [279] proved that the violation of the Geometric Conservation Law (GCL) causes the violation of the mass conservation leading to significant pressure oscillations, which damage the simulation's accuracy. As a result, various attempts have been made to increase to solution's accuracy while retaining the conservative identity of the method.

Schneiders et al. [278], [277] extended the work done by Hartmann et al. [124] to compressible viscous flows around moving boundaries. Their technique was distributing the loss in mass to neighboring cells without though discussing its effect on the accuracy of the flow solution. Moreover, Guthner et al. [120] introduced a level-set approach to keep track of the boundary's motion, sacrificing the accuracy of the boundary's representation. Recently, Muralidharan et al. [218] developed a second-order cell clustering algorithm to ensure the conservative laws' imposition even for large structural displacements. However, this method imposes strict limitations to the maximum allowed displacement of the boundary at each time step.

Another alternative, Aftosmis et al. [219] and Asao et al. [21], is based on the geometric construction of space-time finite volumes. Although this approach offers a proper treatment for the solidified and newborn cells, its complexity limits its use only to the development of simpler methods. Other implementations include cell merging methods [29], [338], [27], implicit time-stepping [5], and flux redistribution methods [200].

The cut-cell method has been applied to a great variety of real-world problems. These applications include, among others, flapping foils [211], flow-induced vibrations [207], diaphragm-driven synthetic jets [317], and objects in free fall [210]. Moreover, it has been used to capture the air-water interface [133], including the detonation of ships [145]. In addition, the cut-cell method has been proven beneficial in simulating internal flows of combustion engines [120], [276], centrifugal pump impellers [12], flows with cavitation [232], particle-laden turbulence [277], and the flow around a space shuttle orbiter [38]. Finally, its use is advantageous in fluid-structure interaction [206] and multiphase flow [334] problems.

Despite the development of several variations of the cut-cell method and its application to various cases, there is much space for improvement. This thesis develops techniques to increase the method's efficiency by contributing to the automatic generation and adaptation of the mesh around stationary or moving geometries. Firstly, it deals with the challenging process of the cut-cells' construction. In particular, it extends the method of Aftosmis et al. [6] by proposing a robust algorithm capable of computing any intersection between a Cartesian cell and an arbitrary triangulated surface. Moreover, the algorithm consistently handles degeneracies of the immersed geometry without the user's intervention offering an autonomous and valuable tool for the mesh generation in 3D practical applications.

Although several attempts have been made to simulate unsteady phenomena around moving geometries accurately, none has gained wide acceptance from the scientific community. Indeed, the previously presented survey illustrates the existing gap in the literature and indicates the difficulties of this challenging task. Hence, the present research supports this effort by improving the aforementioned cell merging method. More specifically, it introduces a technique that extends the existing methods' capabilities (e.g., the one in [219]), offering a smooth representation of the flow close to the moving wall. Furthermore, the method's accuracy has been verified by comparing the developed software's results with experimental and numerical data from solvers based on body-fitted meshes. Moreover, its efficiency is demonstrated by successfully applying the method to 3D industrial problems.

1.2 Shape Optimization in Fluid Dynamics

Shape optimization applied in fluid dynamics modifies a given geometry to maximize its aerodynamic or hydrodynamic performance. The geometry's boundary is controlled by a set of variables called design variables ($\vec{b}$). These can be the coordinates of control points defining the shape of the geometry through a parameterization tool. Alternatively, the coordinates of the nodes comprising the discretized solid boundary can be used. The performance of a given geometry, and thus, the set of the corresponding design variables, is measured by computing the objective function F of the optimization problem. This function is the quantity of interest defined in each application, e.g., the drag of a wing or the efficiency of a turbomachinery blade row. The value of the objective function is computed by solving the Partial Differential Equations (PDEs) describing the physical phenomenon under consideration, such as the Euler or Navier-Stokes equations. However, the absence of an analytical solution to these equations makes the optimization process challenging, giving rise to numerous approaches seeking the optimal set of design variables that maximizes or minimizes the objective function. Optimization methods can be classified according to various criteria. A common taxonomy is based on the method used to search the design space for the optimal solution(s) and categorizes the optimization methods into stochastic and deterministic ones.

After a short presentation of stochastic methods and, in particular, evolutionary algorithms in subsection 1.2.1, the analysis proceeds in subsection 1.2.2, focusing on gradient-based methods, which are mainly used in this thesis. In addition, it briefly presents methods for computing the necessary gradient of the objective function. Then, subsection 1.2.3 introduces the adjoint method and explains its formulations. Finally, a literature survey about the implementation of adjoint methods on IBMs and the cut-cell method is documented in subsections 1.2.4 and 1.2.5, respectively, indicating the research gap addressed by this work.

1.2.1 Evolutionary Algorithms

Stochastic methods explore the design space in a heuristic-based manner [205], [290]. Evolutionary Algorithms (EA) are a notable representative of this class, and they are based on a population-based optimization inspired by the Darwin's theory of biological evolution. According to that, individuals correspond to different sets of design variables and are organized into generations. During the optimization process, crossover, parent selection, and mutation operators are applied to the members of each generation producing new individuals progressively closer to the optimal solution.

The use of EAs is beneficial in several aspects. In particular, based on the randomized search of the design space, they avoid local extrema reaching the global optimum. Moreover, their use in multi-objective optimization problems is advantageous due to their ability to compute the Pareto front of non-dominated solutions directly. Another important characteristic is their non-intrusiveness. Hence, no direct access is needed to the evaluation tool, i.e., the CFD solver, which is used as a black box. Finally, compared to deterministic methods, the straightforward treatment of constraints makes it a valuable tool for real-world applications.

However, an extensive number of evaluations is usually needed to reach the optimal solution. Therefore, the optimization's computational cost may be prohibitive, especially in CFD-based applications where a flow simulation is needed for each evaluation. Various techniques have been proposed to surpass this problem, such as implementing surrogate evaluation models [155], [46] or developing distributed and hierarchical optimization schemes [79], [156]. Moreover, EAs may be computationally expensive in applications identified by a large number of design variables. In such cases, even using the above methods may not reduce the cost at a reasonable level. A remedy to this issue is the development of unsupervised learning techniques such as the Principal Component Analysis (PCA) method [127], [169].

Although EAs have been used supplementary in this thesis, they are beyond its primary interest, and, thus, they will not be discussed further. A detailed study of this field can be found in [151]. Instead, the principal contributions of this dissertation are in the area of deterministic methods, discussed in subsection 1.2.2.

1.2.2 Gradient-Based Optimization Methods

The most well-known deterministic approaches belong to the Gradient-Based (GB) optimization methods class. These approaches use the derivatives of the objective function with respect to (w.r.t.) the design variables, also called sensitivity derivatives, to explore the design space. Since every new update of the design variables is based on the direction indicated by the sensitivity derivatives, significantly fewer optimization steps are necessary than those required by stochastic methods [98]. However, contrary to the latter, GB methods are often trapped into local minima, impotent to reach the global optimum. Moreover, their implementation in multi-objective optimization problems is not straightforward.

Hybrid optimization methods attempt to combine the advantages of the stochastic and deterministic methods treating their previously discussed drawbacks. Such approaches use gradient-based methods to improve promising individuals during the optimization employed by evolutionary algorithms. Thus, in multi-objective optimization problems, a single objective function is defined each time as the target of the gradient-based method, while the rest act as constraints [163]. Another alternative uses weights to combine multiple objective functions to a single one, targeted by the gradient-based method [149], [166]. Although this thesis mainly deals with single-objective optimization problems, the tools developed in [153] are used elsewhere.

Various GB methods have been proposed implementing different techniques to update the design variables at each optimization cycle. They are separated into two primary strategies, the line search [95] and the trust-region [341] ones. In particular, line search methods initiate from a starting point in the design space and choose a descent direction $\vec{p}$ (for minimization problems) along which the optimization will proceed. Then, they solve a 1D optimization problem to compute the appropriate step size that best maximizes/minimizes the objective function. In contrast, the trust-region methods first choose a maximum distance at which a model function accurately enough approximates F. Then, they seek the step size and direction along which the best improvement is attained.

In line search methods, the derivatives of F are used to determine the descent direction. According to the most straightforward approach, referred to as the steepest descent method, the descent direction is aligned to that of the objective's gradient. However, due to its poor efficiency, other methods have been proposed. Among them, the Newton method is one of the most remarkable. It uses the second derivatives, i.e., the Hessian matrix, of F providing fast convergence of the optimization process. However, the additional effort required for the extra derivatives' computation restricts its use in CFD-based problems. Therefore, quasi-Newton methods, such as BFGS [95], have been introduced, approximating the Hessian matrix using only first-order derivatives. Another method that balances simplicity and efficiency is the conjugate gradient method [96] adopted in this thesis. This method computes the i^{th} component of the descent direction as

$$p_i^{new} = - \left. \frac{\partial F}{\partial b_i} \right|_{new} + \beta^{new} p_i^{old}$$

where various alternatives exist for the computation of β^{new} using exclusively first-order derivatives from the current and previous optimization cycle. Although its convergence is slower than in the two previous methods, its low memory requirements make it suitable for CFD-oriented applications. Finally, a detailed presentation of gradient-based optimization methods can be found in [229].

The computation of the gradient of F strongly affects the efficiency of the optimization. Unfortunately, in applications related to fluid dynamics, the objective function's expression w.r.t. the design variables is rarely available in a closed form, making its differentiation pretty challenging. Therefore, in the absence of an analytical computation, the gradient is numerically approximated. Finite Differences (FDs) stand for the most straightforward way to approximate the gradient. When a central finite difference scheme is adopted, each design variable is subjected to a positive and negative perturbation by a small quantity ϵ while the rest remain constant,

$$\frac{\partial F}{\partial b_i} \simeq \frac{F(b_1, b_2, \cdots, b_i + \epsilon, \cdots, b_N) - F(b_1, b_2, \cdots, b_i - \epsilon, \cdots, b_N)}{2\epsilon}$$

Each time, the geometry's boundary slightly changes, affecting the surrounding flow field and resulting in a different objective function value, which is evaluated by solving the governing equations anew. Therefore, the gradient's computational cost is proportional to the number of design variables, making the implementation impractical in multivariable problems.

Moreover, the resulted derivatives are sensitive to the value of the user-defined variable ϵ. Large values reduce the method's accuracy, while small values cause round-off errors, damaging the prediction of the gradient. The repetitive process for determining its appropriate value further increases method's computational cost. However, its simplicity makes it valuable for validating other, more efficient methods. The dependency of ϵ is eliminated by using the Complex Variable Method [194], [17], the cost of which is reduced to half but remains proportional to the design variables' number. Another alternative is the Direct Differentiation Method [282], [28], which is preferred for computing the Hessian matrix or in applications with more objective functions than design variables. Since both are out of this thesis scope, and thus, this method will not be discussed further.

1.2.3 The Adjoint Method

In contrast to the previously presented approaches, the adjoint methods compute the gradient at a low cost, independent of the number of design variables [139], [247]. This property makes them a viable alternative, appropriate for industrial-scale applications. Its remarkable efficiency originates from introducing the so-called adjoint variables, which satisfy the field adjoint equations. At each optimization cycle, the flow equations are firstly solved. Next, the resulted flow field is used to solve the system of adjoint PDEs. The latter comes at a cost comparable to that of the governing equations. Finally, the flow and adjoint fields are introduced to the sensitivity derivatives expression, computing the required gradient. Hence, the total computational cost is equivalent to solving the flow equations twice.

The concept of the adjoint approach was introduced by Lions in 1971 [183]. However, its first application in the field of fluid dynamics was made much later, in 1984, by Pironeau [247], who studied physical systems described by an elliptic PDE. Later, Jameson extended this work by mathematically developing [139] and applying [257], [140] the adjoint method to the Euler equations. Over the following years, this method has been employed in various real-world applications, such as in aeronautical [259], [193], [168], [311] and automotive industries [233], [239], [237], [150]. A detailed literature survey on adjoint methods can be found in [237].

Two main approaches constitute the adjoint methods depending on the way they derive the adjoint equations. Firstly, the discrete approach [86], [16], [103] uses the discretized objective function and governing equations, and through their differentiation, it defines the adjoint PDEs, the corresponding boundary conditions, and the sensitivity derivatives expression in a discretized form. In particular, the flow solver is differentiated either "by hand" or in a more automated way called Algorithmic Differentiation [116], [71], [125]. The second approach is usually preferred when the software under differentiation is quite complex, offering a straightforward process to build its adjoint counterpart. Many tools performing automatic differentiation are available, such as TAPENADE [300], ADIFOR [196], and ADOL-C [328]. However, the resulting software tends to have high memory requirements, and thus, its use may be prohibitive in large-scale applications. In contrast, "hand differentiation" can prove tedious but avoids memory overuse.

In the second approach, called continuous adjoint method [18], [258], [164], the objective function and governing equations are differentiated in their continuous form. After the appropriate mathematical development, the field adjoint equations arise, accompanied by the corresponding boundary conditions and the expression of sensitivity derivatives. Then, an adjoint discretization is chosen, usually equivalent to the one used for the governing PDEs. Finally, the resulted adjoint field is substituted in the sensitivity derivatives expression, which allow two different formulations. The first one comprises only surface integrals and is referred to as Surface Integral (SI) formulation [235], while the second one also involves field integrals and is called Field Integral (FI) formulation [238]. In practice, the SI approach does not take the impact of the volume mesh displacement during the optimization under consideration, and thus, is unable to compute the accurate value of the gradient consistently. On the other hand, the FI approach is more reliable but computationally expensive. Thus, the Enhanced-Surface Integral (E-SI) formulation [160] has been proposed, which bridges the gap between the aforementioned methods by introducing the concept of the adjoint mesh displacement and by additionally solving the corresponding adjoint equation.

The discrete and continuous approaches mentioned above are not equivalent in the sense that they result in different approximations of the gradient. Hence, their unique features and benefits have extensively been discussed in the literature [104], [220], [138]. The main advantage of the discrete approach is its ability to deliver the necessary gradient accurately. Moreover, the development of the software computing the adjoint field is straightforward. On the other hand, in the continuous formulation, the flow and adjoint solvers share many similarities, which reduces the invested time for software development and the demand for memory resources. However, choosing the appropriate adjoint discretization scheme is far from trivial and crucial for accurately predicting the necessary gradient. The present study underlines the usefulness of both discrete and continuous approaches by contributing in both directions. In particular, it develops a discrete adjoint solver by hand-differentiating the cut-cell software. Additionally, it investigates the accuracy of various schemes used for the continuous adjoint PDEs discretization. Finally, it applies both approaches to practical optimization problems.

The previous short discussion on the adjoint methods denotes not only the extent of various formulations but also the central role of mesh perturbation caused by the geometry's displacement during the optimization. Most implementations on body-fitted meshes use mesh deformation tools to smoothly adjust the volume mesh on the continuously modified optimized shape. The effect of the mesh perturbation on the objective function is taken into account by differentiating the corresponding tool [212], [256], [104]. Several studies have been performed on the impact of mesh sensitivities on the gradient's accurate computation, showing that their elimination severely damages the optimization process [18], [188], [160]. On the other hand, their computation entails the solution of an extra adjoint equation in both the discrete [197] and continuous [160] methods (at least in the E-SI formulation), increasing the computational cost of the optimization. Furthermore, mesh deformation tools may fail to handle considerably large shape modifications, causing the premature termination of the optimization process. Then, user's intervention is unavoidable, harming the unsupervised performance of the optimization. In contrast, this thesis suggests the use of IBMs, which circumvent these issues by offering a robust and reliable framework for the optimization implementation.

1.2.4 Adjoint Formulation to the Immersed Boundary Methods

Contrary to the body-conforming approaches, the immersed boundary methods overcome the previously mentioned problems by restricting the effects of the geometry's deformation to a narrow zone of cells close to the wall. In particular, their capability to decouple the mesh generation from the geometry's complexity allows for a fully automated optimization regardless of the extent of the shape's modification. Moreover, IBMs combine the high accuracy of methods incorporating the mesh displacement differentiation (e.g., the FI formulation) with the efficiency of strategies that avoid the mesh perturbation effect (e.g., the SI formulation). Although several approaches exploit the advantages of the immersed boundary methods using various optimization strategies [66], [225], [221], [261], limited research has been conducted on the adjoint-based gradient computation implemented in Cartesian meshes. Some of the few examples are listed below.

Firstly, Dadone et al. [73], in 2005, introduced the discrete adjoint to the ghost cell method and applied it in 2D flows. The necessary differentiation of the discretized flow equations and the ghost boundary condition were approximated by FDs. Some years later, Hinterberger et al. [129] performed topology optimization supported by the continuous adjoint method, applied in automotive exhaust systems. A staircase representation of the boundary was used, where each mesh cell was marked as fluid or solid. Furthermore, Xu et al. [336] implemented the continuous adjoint method in 2D flows around moving bodies, where a continuous forcing approach expressed the presence of the immersed geometry within the flow. More recently, Okubo et al. [231] applied the discrete adjoint formulation in a flow solver implementing a ghost cell technique in steady 2D problems. In a similar framework, Rutkowski et al. [265] presented an adjoint Lattice Boltzmann Method for multi-objective optimization of a 2D flapping airfoil.

1.2.5 Adjoint Formulation to the Cut-Cell Method

The implementation of adjoint methods on a cut-cell framework is remarkably rare in the literature. Nemec et al. [224] first introduced in 2005 the discrete adjoint formulation to the cut-cell method applied in transonic and supersonic flow optimization problems governed by the 3D steady Euler equations. In particular, they hand-differentiated the state equations w.r.t. the flow variables assuming that the limiter remains constant during the shape transformation. Moreover, the sensitivity of the governing equations to the design variables was approximated by finite differences. Then, in 2006 [222], they extended that work by incorporating an accurate computation of the mesh sensitivities through the exact linearization of the cut-cell geometry. In a succeeding study [223], they examined the effect of limiters on the optimization, showing that the constant limiter assumption may harm the gradient computation documenting a relative error of around 16% compared to FDs. Finally, a later work of the same group [332] proposed an adjoint-based adaptive mesh refinement method to minimize discretization errors of the flow simulation.

Moreover, worth saying is the research in the field of cut finite elements. Firstly, Benk et al. [34] applied this discretization method in fluid-structure interaction optimization problems governed by the 3D Stokes equations. In particular, they employed Nitsche's penalty discretization method [195], which supports the computation of finite element integrals in cut-cells and computed the objective's gradient by using the discrete adjoint formulation. Moreover, Jenkins et al. [142] carried out a topology optimization taking advantage of the cut-cell method's accurate representation of the fluid-solid interface. Again, Nitsche's method was used and the continuous adjoint formulation was implemented for the gradient computation. The method was applied in 2D steady optimization problems, the fluid part of which was modeled by the incompressible Navier-Stokes equations. Finally, the cut element method assisted by discrete adjoint has been applied to acoustic shape optimization problems [42], [82].

According to the above literature survey, there is much space for investigation concerning the adjoint methods implemented in a cut-cell environment. The present thesis contributes to the existing literature in the following ways. Firstly, it extends the work of Nemec et al. by encapsulating the limiter differentiation in the discrete adjoint formulation of the 3D compressible Euler equations. Then, it proceeds to the implementation of the method in viscous and unsteady flows. Finally, the method is applied in 3D incompressible flow optimization problems for the first time in literature. Another innovative aspect of this study is the introduction of the continuous adjoint formulation to the cut-cell method by using a finite volume discretization. Finally, an important side product of this thesis is the development of the ghost-cell variation of the continuous adjoint method for the 3D unsteady Euler equations.

1.3 Thesis Outline

Motivated by the open issues indicated in the previous sections concerning the cut-cell method and its use in conjunction with the adjoint methods, this dissertation develops strategies and computational methods that bring the CFD-based analysis and optimization closer to industrial reality. The thesis is structured along two axes. Firstly, chapters 2-5 are dedicated to the flow simulation employing the cut-cell method. Then, chapters 6-9 are concerned with the adjoint methods and optimization problems. Finally, chapter 10 summarizes the thesis contributions and proposes some concepts for future development. The subject of each chapter is shortly presented below.

Chapter 2 describes the algorithm generating a Cartesian mesh appropriate for flow simulation based on the cut-cell method. The mesh is adapted to the immersed boundaries and to flow phenomena of particular interest. The process is based on an octree data structure and guarantees a smooth transition between regions of different refinement levels. Emphasis is laid on the detailed presentation of the algorithm constructing the cut-cells by intersecting the mesh with the geometry's boundary. Small cut-cells were merged with their neighbors to avoid instabilities during the flow solution. Furthermore, a mesh partitioning technique is discussed based on the Hilbert space-filling curve. In addition, a strictly conservative method is developed dealing with Cartesian meshes over moving geometries. Finally, the derivatives of geometric quantities required during the gradient-based optimization process are computed.

Chapter 3 presents the numerical discretization of the compressible or incompressible flow equations to a cut-cell mesh. The examined flow model concerns the steady or unsteady Navier-Stokes equations, where the artificial compressibility method is applied to stabilize the incompressible solver. Discretization is based on a cell-centered, second-order finite volume method employing the MUSCL scheme and Roe's approximate Riemann solver. In unsteady simulations, the time integration is based on a dual time-stepping technique. Special treatment is made for cells that appear or disappear from the fluid region of the mesh in cases involving moving bodies. Finally, this chapter describes a ghost-cell approach for steady and unsteady flow simulations.

Chapter 5 aims to validate/verify the developed cut-cell flow solver in compressible and incompressible cases selected from the literature. Firstly, inviscid flows in external and internal aerodynamics are considered, demonstrating the benefits gained by the direct imposition of the wall conditions. Then, laminar flow cases are examined, focusing on the effect of the cut-cells' irregularities on the boundary layer representation. Emphasis is laid on the ability of the developed software to produce highly accurate results, equivalent to those obtained by body-conforming meshes. Finally, the proposed method's ability to correctly predict flows around moving bodies satisfying the conservation laws is investigated.

Chapter 5 illustrates the ability of the developed software to handle industrial cases. Moreover, it indicates the benefits of implementing IBMs in various applications due to the absence of limitations that usually accompany body-conforming meshes. Indeed, the chosen applications accommodate complex geometries in relative motion, proving the superiority of the cut-cell method against other CFD approaches. More specifically, they deal with the unsteady compressible or incompressible internal flow in a moving valve, a scroll machine, a diaphragm pump, and a submersible pump.

Chapter 6 is concerned with the mathematical development of the continuous adjoint method for compressible and incompressible flows implemented to the cut-cell and the ghost-cell method. Each term of the governing equations is separately differentiated, computing the contributions to the formulation of the adjoint equation and the corresponding boundary conditions and sensitivity derivatives. Moreover, an investigation is carried out about the adjoint Riemann problem definition and the discretization of the field adjoint equations. Subsequently, the unsteady variant of the adjoint method is studied, and data compression techniques are used to deal with the increased demand for memory resources.

Chapter 7 discusses the discrete adjoint formulation to the cut-cell method. Therefore, a hand-differentiation process is presented for all terms of the steady and unsteady viscous flow equations for both compressible and incompressible flows. Moreover, no simplifications are introduced, giving rise to the exact discrete adjoint expressions and the accurate computation of the objective's gradient. Furthermore, a comparison is made between the resulting discrete terms and the corresponding discretization schemes proposed for the continuous adjoint equations. Particular emphasis is put on the proper differentiation of algorithms treating the appearance and disappearance of mesh cells in applications involving moving geometries. Finally, smoothing techniques of the resulting sensitivity derivatives are proposed.

Chapter 8 demonstrates the ability of the developed adjoint software to accurately compute the objective's gradient. The assessment of the adjoint software concerns compressible or incompressible flows around stationary or moving geometries. The computed sensitivity derivatives are compared with central FDs in each case, resulting in minor deviations. After confirming the accuracy of the computed derivatives, a shape optimization is carried out each time, using the conjugate gradient method. Prompted by this investigation, the physical meaning of the adjoint variables is shortly discussed.

Chapter 9 implements gradient-based optimization assisted by the adjoint method in real-world applications. The coordinates of the surface nodes constituting the geometry under modification are the design variables of the optimization problem. Thus, the high number of design variables suggests the adjoint method as the only reasonable approach for gradient computation. The presented optimization cases are concerned with the total pressure losses minimization in an S-duct, the lift maximization of a wing, the outlet tangential velocity minimization in a submersible pump, and the back-flow minimization along with the volume flow rate maximization of a diaphragm micropump. In the last multi-objective optimization, a hybrid approach is used, which combines evolutionary and gradient-based methods. Additionally, uncertainties are introduced to the design variables of this optimization problem. Finally, contrary to the previous applications, the adjoint to the ghost-cell method is used to optimize a compressor rotor.

The theoretical development of the flow or adjoint problem presented in the previous chapters is facilitated by 16 Appendices. In particular, Appendices A and B complete the analysis of the mesh generation. Appendices C to I further describe the theoretical background of the discretization used in the compressible and incompressible flow equations. Appendices J to L discuss details about the boundary conditions accompanying the continuous adjoint PDEs and the schemes used for discretizing the adjoint convection term. Appendices M and N present the SVD and PGD methods, respectively, for data compression used in unsteady optimization problems. Finally, Appendices O and P give details about the mathematical formulation of the convection term of the discrete field adjoint equations.

Computations have been performed on the high-performance computational platform “VELOS” of the PCOpt/NTUA Unit. Two DELL PowerEdge blade servers were used for simulations carried out by the cut-cell software and its adjoint counterpart. Each of them is provided with 48 double-threaded AMD EPYC 7401 processors with 128 Gb RAM and 2 GHz clock speed. Communication between blades employs the MPI protocol. In addition, a supplementary cluster of NVIDIA Tesla K20 GPUs was used for the ghost-cell flow and adjoint software operation.

Chapter 2

The Cut-Cell Mesh Generation

The unique characteristic that distinguishes the cut-cell method from other CFD approaches is the special structure of the mesh used for the flow simulation. This chapter describes a rapid, robust, and automated 3D Cartesian mesh generation method for stationary and moving solid bodies capable of supporting inviscid and laminar flow solvers.

The mesh generator takes a triangulated surface as input and generates a volume mesh through the repetitive subdivision of an initial cell which defines the domain boundaries. The process is based on an octree data structure, which is presented in section 2.2. Moreover, methods responsible for the mesh quality improvement and the enhancement of the flow simulation's accuracy are discussed in section 2.3. These approaches consider the smooth transition between regions of different refinement levels and the mesh adaptation in the vicinity of solid boundaries, shock waves, and large eddies.

However, the tree-like data structure introduces several complications damaging the flow solver's efficiency. These are related to the detection of neighboring cells and the computation of geometric quantities necessary for the flow simulation. Thus, a fully unstructured approach is preferred, presented in section 2.5. Moreover, this section studies the faces and nodes numbering as well as the mesh separation into its fluid and solid parts.

The most challenging part of the mesh generator is cutting the Cartesian hexahedra intersected by solid surfaces, producing polyhedral control volumes. A robust algorithm is discussed in section 2.4, which handles any possible intersection, clipping the hexahedra against the body's surface, and creating complex cut-cell topologies. Issues relating to roundoff errors resulted from the intersection computation are also studied. The generated cut-cells can be polyhedra of any shape and size. Thus, small cell fragments may appear next to much bigger cells, causing numerical instabilities during the flow solution process. This problem is usually mentioned in the literature as the "small cell problem". A cell-merging approach is developed in section 2.6 to address this problem by geometrically merging small cells with their neighbors.

The mesh partitioning has a major impact on the efficiency of the flow solver's parallel behavior. The partitioner shown in section 2.7 is developed explicitly for Cartesian meshes taking advantage of their special structure. It is based on the Hilbert space-filling curve [4] exploiting its essential properties, further explained in the same section.

In applications concerning moving solid bodies, the refined mesh follows their motion employing local refinement and de-refinement operations. A strictly conservative method is presented in section 2.8 to handle large geometry displacements by enforcing a cell clustering algorithm. In particular, the sudden appearance or disappearance of cells from the domain is successfully treated, ensuring the satisfaction of the conservation laws during the unsteady flow simulation. Additionally, the section focuses on schemes used to extrapolate the current flow field to the mesh of the following time step.

Finally, considering that this thesis deals with gradient-based shape optimization algorithms, mesh differentiation is necessary. Therefore, all geometric quantities computed by the mesh generator should be differentiated w.r.t. the nodes describing the input surface. Section 2.9 provides the corresponding mathematical development resulting in the appropriate formulas for the computation of the requested derivatives.

2.1 The Cartesian Mesh Data Structure

A wide variety of Cartesian mesh generation methods has been proposed in the literature, dating back to the 1970s [55]. The firstly appeared meshes of this kind were uniform, and their use was limited to few academic problems. Then, the Adaptive Mesh Refinement (AMR) [41] extended the Cartesian meshes' ability to handle practical applications by accurately representing the flow solution at a low cost. Furthermore, this refining process through cell-splitting operations effectively combined the computational efficiency offered by a Cartesian structured mesh with the flexibility of the widely used unstructured meshes.

Before discussing the AMR in detail, it is essential to introduce the terminology adopted from computer science for the commonly used tree data structures. The tree is a collection of hierarchically structured nodes linked with one parent node starting from an initial root node. Fig. 2.1 shows a simple tree, where each node has at most two children. Moreover, a leaf is a node located at the bottom of the tree having no children. Every other node is called internal. Finally, an octree (or quadtree in 2D meshes) is a tree where each internal node has at most eight children (or four children in 2D).

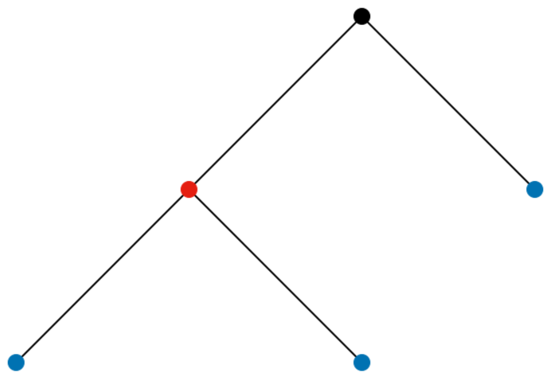


Figure 2.1: A tree data structure of 3 levels. The root, internal, and leaf nodes are colored black, red, and blue, respectively.

Two main approaches based on the AMR have been developed in the literature. The first approach uses a sequence of overlapped structured meshes at different hierarchies or levels [39], [251]. The nested hierarchical nature of the mesh perfectly matches the tree-like data structure, which assists the communication between the regular Cartesian meshes represented by tree nodes. Although efficient solvers for structured meshes can be applied to each sub-mesh, the lack of flexibility in the sub-meshes definition may lead to regions covered with unnecessarily dense mesh, wasting substantial computational resources.

The computational efficiency of the AMR method can be increased by employing the second approach, which matches each tree node with a mesh cell, allowing better control of the mesh resolution [343], [242], [68], [55]. However, the computation of the connectivity between cells is a very time-consuming process, especially in big meshes, because a significant part of the tree data structure must be traversed before a neighbor can be found.

The development of the Fully Threaded Tree (FTT) [162] further reduced the memory overhead required by the second AMR method. This new data structure stores its information in structures called octs. Each oct contains a pointer to the parent cell, a pointer to each child cell, the parent cell's refinement level, and its position in the domain. Although FTT allows for more efficient access to the information stored in the tree, a complete tree traversal starting from the root cell is still frequently required.

A better version of the FTT data structure is the Cell-based Structured Adaptive Mesh Refinement (CSAMR) data structure [144], which constitutes the basis for developing the data structure used in this thesis. Its novelty was the introduction of Cartesian-like indices, which identify each cell by mimicking the cells' enumeration in structured meshes. A triplet (or a pair in 2D) of indices is stored for each cell, which gives all the needed information about its parent, children, neighbors, and Cartesian coordinates, significantly reducing the memory usage. Therefore, the traversal of a considerable part of the tree, required by the previous approaches, becomes unnecessary.

Although the CSAMR reduces the cost of the mesh connectivity computation, the repetitive access to neighboring cells through the tree structure during the flow simulation still delays the solution process. Thus, this thesis proposes an alternative method that combines the CSAMR's flexibility with the advantages of a conventional unstructured data structure, which considers the mesh as a collection of arbitrary polyhedra. Hence, despite using an unstructured framework, the hexahedral shape of most cells leads to a pretty compact data structure. The developed method consists of two stages. Initially, it uses the CSAMR to efficiently generate the

Cartesian mesh, keeping the required memory as low as possible. Then, it transfers the necessary data to a face-based data structure, similar to the one presented in [51], where the connectivity is explicitly stored and easily accessed by the flow solver. Sections 2.2 and 2.5 explain in detail each of the two stages, respectively.

2.2 The Octree Mesh Generation

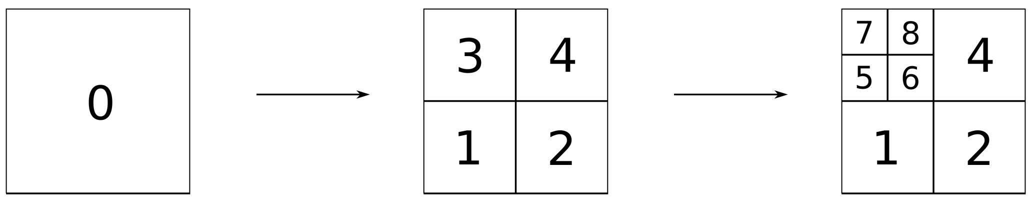
This section presents the first part of the mesh generation process, which results in a initial mesh version. Its assistance by the tree data structure is discussed in subsection 2.2.1. Additionally, the mesh adaptation to the geometry's surface is studied in detail. Next, subsection 2.2.2 introduces the integer coordinates used for quickly identifying each cell. Their properties, expressed by mathematical formulas, are given as well. Furthermore, methods to detect the immersed geometry are examined in subsection 2.2.3. The whole algorithm is presented in a pseudocode format in subsection 2.2.4.

2.2.1 The Octree Data Structure

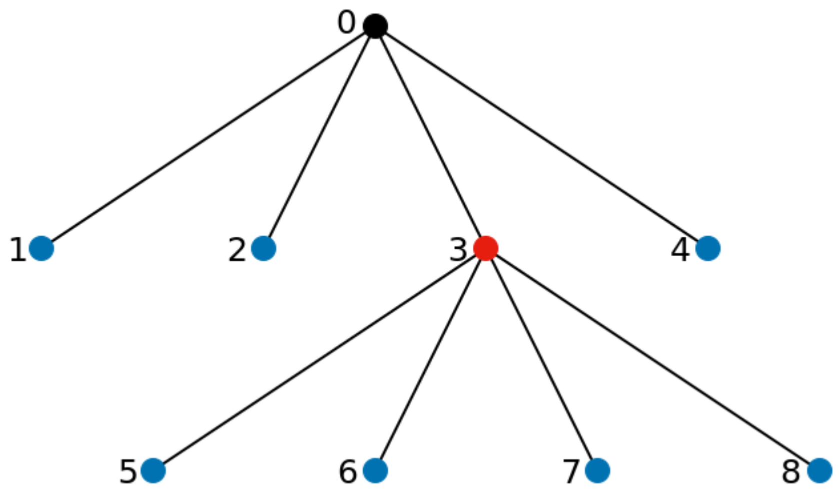
The input of the volume mesh generator is the geometry's surface in the Standard Tessellation Language (STL) format, typically provided by a Computer-Aided Design (CAD) package. Moreover, the box-shaped domain is described by inserting its length along each Cartesian direction (d_x, d_y, d_z) and the coordinates of its centroid (x_0, y_0, z_0). The mesh generation begins with the definition of the root cell, which coincides with the box itself. Then, the root cell is equally subdivided into 8 (or 4 in 2D meshes) children constituting the new generation of cells in the tree data structure.

Subsequently, the cell splitting procedure repeatedly produces new generations of cells extending the octree data structure. Each newborn cell is geometrically contained within its parent's boundaries and is placed below the parent cell in the tree data structure. Therefore, whenever a cell is deemed appropriate for subdivision, a new sub-branch is created below its position in the tree, and the mesh generation process is driven through the new sub-branch, implying the partition criteria to the newly created cells. Fig. 2.2a presents a 2D root cell split into four children, the one of which is further subdivided into four offspring. The resulted tree is shown in fig. 2.2b.

The splitting of each cell can be isotropic [51], [124] or anisotropic [67], [6]. Contrarily to an isotropic division, the anisotropic refinement allows for different splitting in each direction. The latter provides a reduced mesh size but sacrifices the simplicity of the data structure supported by the Cartesian mesh nature, and thus it is avoided.



(a)



(b)

Figure 2.2: (a) The root cell indexed 0 is subdivided into four children, namely 1, 2, 3, and 4. Then, cell 3 is further subdivided into cells 5, 6, 7, and 8. (b) The process is depicted in the tree structure, where each node corresponds to one cell. The nodes' colors are explained in fig. 2.1.

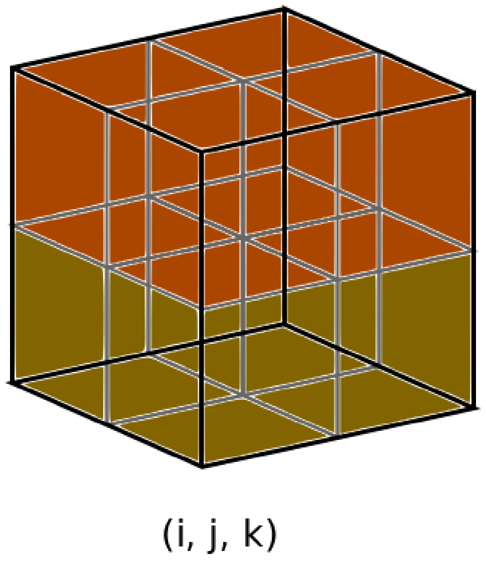
Next, the refinement criteria are discussed. Initially, cells are subdivided until an acceptable refinement level, defined by the user, is reached. The satisfaction of this condition produces a uniform Cartesian mesh. Subsequently, the subdivision process continues, triggered by either geometric or flow field requirements. Nevertheless, only geometry-based adaptation criteria are used during the initial mesh generation due to the absence of the flow solution. Mesh adaptation guided by local flow phenomena will be discussed in section 2.3. So far, the mesh is adequately refined in the wall's proximity by subdividing cells cut by the solid boundary.

During the subdivision process, the maximum refinement level difference between neighboring cells is limited to one. This restriction is imposed along the mesh generation process by splitting each cell adjacent to more than four (or two in 2D) neighboring cells through a single face. Hence, the intersected cells' division quickly propagates through the mesh, affecting cells far from the immersed geometries, increasing the mesh quality by smoothly varying its resolution from dense regions close to the bodies to coarser areas in the far-field. Furthermore, this constraint accelerates the mesh connectivity computation avoiding the time-consuming traversal of considerable parts of the octree structure.

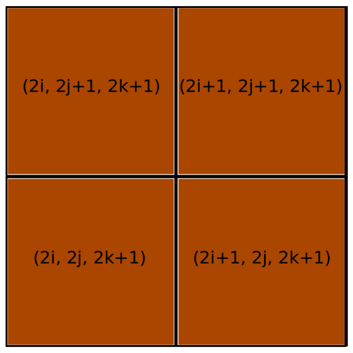
The mesh refinement process terminates after a predefined limit is met. Two different limits can be imposed, leading to a differently refined mesh. In the first case, the user specifies the minimum allowed cell size, preventing the subdivision of cells smaller in volume than the limit's value. This condition provides an almost uniform mesh resolution close to the wall, independent of the geometry's structure. Alternatively, cells are subdivided until the mesh resolution becomes similar to the adjacent triangles' size of the solid surface. This requirement is achieved by refining the mesh until all cells intersected by the wall contain at most two surface nodes. Although the mesh is usually independent of the surface mesh structure in all IBMs, defining the local cell size by the surface resolution offers a direct and flexible way to control mesh generation. In such a case, the surface discretization should be finer in high curvature regions, which usually induce complex flow phenomena.

2.2.2 The Integer Coordinate System

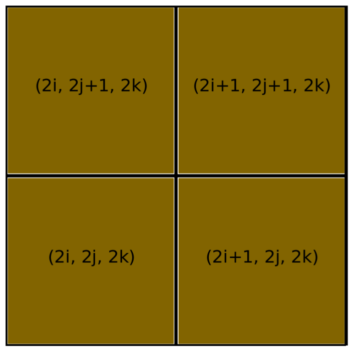
A standard structured mesh indexing (i, j, k) is stored to identify each cell efficiently. By definition, the triplet of indices determining the root cell is $(1, 1, 1)$. The rest of the cells are automatically labeled following the rule illustrated in fig. 2.3. Fig. 2.3a depicts an arbitrary cell, identified by integer coordinates (i, j, k), split into eight equally sized subcells. Colors are used to separate the two quartets occupying the bottom and top half of the cell. The indices given to each subcell of the two quartets are shown in figs. 2.3b and 2.3c.



(a)



(b)



(c)

Figure 2.3: (a) A Cartesian cell with integer coordinates (i, j, k) is subdivided into 8 children. Their integer coordinates are shown in (b) and (c) for the upper and lower quartet, respectively.

Based on this rule, the children and the parent of each cell are easily computed as

$$(i_c, j_c, k_c) = (2i + l_1, 2j + l_2, 2k + l_3) \quad \forall\ l_1, l_2, l_3 = 0, 1 \tag{2.1}$$

and

$$(i_p, j_p, k_p) = \left(int\left[\frac{i}{2}\right], int\left[\frac{j}{2}\right], int\left[\frac{k}{2}\right] \right) \tag{2.2}$$

where the function $int[x]$ returns the integer part of x. Fig. 2.4 defines a local numbering for the children of each cell, useful in later topics.

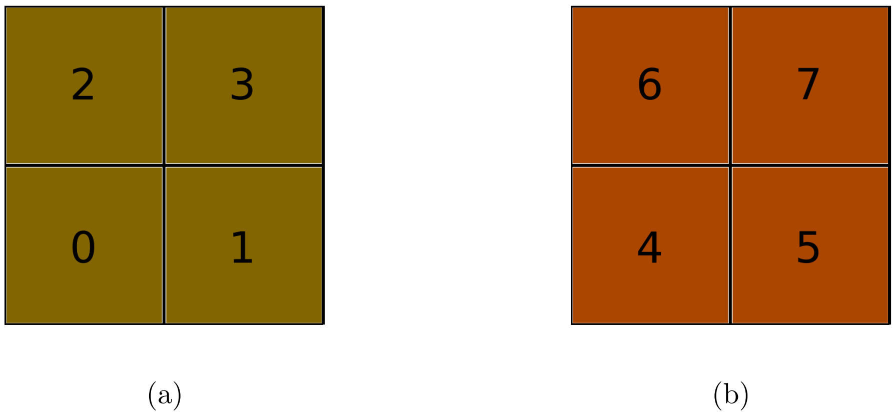


Figure 2.4: Children local numbering for (a) bottom and (b) top cells of fig. 2.3.

Moreover, each generation of cells is identified by integer L, which shows the level of refinement. Initially, the level of the root cell is zero representing the coarsest possible level. The 8 cells of the next generation correspond to $L = 1$. Generally, the level of an arbitrary cell is equal to the level of its parent increased by one and is given in one of the three following equivalent alternatives

$$L = int\left[log_2(i)\right] = int\left[log_2(j)\right] = int\left[log_2(k)\right] \tag{2.3}$$

Finally, by using the stored index information, the dimensions and centroid of each cell can explicitly be calculated as

$$(\Delta x, \Delta y, \Delta z) = \frac{1}{2^L}\left(d_x, d_y, d_z\right) \tag{2.4}$$

and

$$\begin{aligned} x_c &= x_0 - \frac{3}{2}d_x + \left(i + \frac{1}{2}\right)\Delta x \\ y_c &= y_0 - \frac{3}{2}d_y + \left(j + \frac{1}{2}\right)\Delta y \\ z_c &= z_0 - \frac{3}{2}d_z + \left(k + \frac{1}{2}\right)\Delta z \end{aligned} \tag{2.5}$$

Even though the integer coordinates identify each cell successfully, introducing a single integer that uniquely specifies each cell would occasionally be beneficial. Consequently, the index ID is defined as

$$ID = 4^L(k-1) + 2^L(j-1) + (i-1) - \frac{6}{7}\left(8^L - 1\right) \tag{2.6}$$

The proof of the above equation is given in Appendix A. Although this quantity is mentioned in the literature (e.g., in [144]), its exponential rise to extremely high values makes it impractical in 3D cases. Finally, the computation of integer powers of two in the preceding equations is implemented very efficiently by using the bitwise left shift operator, available in most programming languages.

2.2.3 Detection of the Immersed Geometry

The mesh adaptation to the solid bodies' surface requires the detection of the embedded geometry. To this end, cells cut by the solid surface should be identified employing a fast and robust tagging procedure based on the already developed tree data structure. The process starts by finding all triangular geometry facets contained to or intersected by the root cell. After its split, the same process is repeated for each one of its eight children. As the mesh subdivision continues, newly created cells inherit the triangle list of their parents. Thus, the list gets shorter after each successive subdivision increasing the efficiency of the cut-cells detection procedure.

The conditions responsible for testing the inclusion or intersection of a triangle by a Cartesian cell are checked remarkably often during the mesh generation and, therefore, should be carefully chosen. At this point, some necessary definitions are given. Firstly, the plane at which each face lies separates the 3D space into two subspaces. Let the one containing the entire cell be called internal and the other external. Moreover, the active area of a face is defined as the set of points satisfying the following conditions. Firstly, they belong to the external subspace of the face, and secondly, their projection to the face's plane belongs to the face.

The algorithm's structure consists of a number of consecutive geometric conditions arranged in ascending order in terms of computational effort. Hence, each criterion is checked only if its former is not satisfied. The conditions are:

1. If at least one triangle vertex is located inside the cell, return true.

2. If all triangle vertices are placed into the external subspace of any face, return false.

3. If two triangle vertices are located in the active area of two opposite faces, return true.

4. If the list of vertices resulting from the Sutherland–Hodgman algorithm, explained in section 2.4.1, is empty, return false. Else return true.

When the mesh generation ends, cells farthest down the hierarchy, called leaf cells, belong to different refinement levels and satisfy all the aforementioned geometric requirements. However, they do not constitute the final version of the mesh. Indeed, extra adjustments are needed to prepare the mesh for the flow simulation process because, until this step, the mesh connectivity is inaccessible by the flow solver, cut-cells have not been constructed yet, and cells that are covered entirely by the solid bodies are still part of the mesh. The following sections deal with these issues explaining the next steps of the mesh generation.

2.2.4 Pseudocode of the Octree Generation

To sum up, given a list of all triangular surface facets, i.e., "*surfaceTriangles*", the data provided and stored in memory by the process presented in this section are:

1. List "*cells*", which stores the integer coordinates for each cell belonging to the octree data structure.

2. List "*cellPosition*" mapping the integer coordinates of each cell with the cell's position in list *cells*.

3. List "*cellIsRefined*", which consists of a boolean variable for each cell that is true only for leaf cells. Cells numbered by list *cells* are in correspondence with those listed in "*cellIsRefined*".

4. List "*cellTriangles*" containing the list's "*surfaceTriangles*" position of tiangles enclosed or intersected by each cut-cell.

Finally, Algorithm 1 presents the mesh generation process in a pseudocode form.

Algorithm 1: Tree Data Structure Mesh Refinement (1)

```
input  : $surfaceTriangles$
output: $cells$, $cellPosition$, $cellIsRefined$, $cellTriangles$

Main Function meshGenerator()
    // create data for root cell
    $\vec{i} \leftarrow (1, 1, 1)$ // integer coordinates
    $cells[0] \leftarrow \vec{i}$
    $cellPosition[\vec{i}] \leftarrow 0$
    $cellIsRefined[0] \leftarrow false$
    $cellTriangles[0] \leftarrow$ intersectionOrInclusion($surfaceTriangles$)

    // create tree data structure
    $refinement \leftarrow true$
    $N_{min} \leftarrow 0$
    while $refinement$ is $true$ do
        $refinement \leftarrow false$
        $N_{max} \leftarrow$ totalNumberOfCells()
        foreach $cell\ c \in [N_{min}, N_{max})$ do
            if cellMustSplit($c$) then
                splitCellAndNeighbours($c$)
                $refinement \leftarrow true$
            end
        end
        $N_{min} \leftarrow N_{max}$
    end
return
```

The main function "meshGenerator" uses three functions, the purpose of which is further explained. The first one, called "intersectionOrInclusion", identifies the triangles of the given list that are totally or partly located in the cell's region. The used criteria are presented in subsection 2.2.3. Moreover, function "cellMustSplit" decides which cell is suitable for subdivision applying the already discussed user-defined criteria. Finally, function "splitCellAndNeighbours" is presented in Algorithm 2 and properly subdivides the given cell and its neighbors, ensuring that their difference in refinement level is at most equal to one. Moreover, its recursive behavior, supported by many programming languages, significantly boosts the mesh generation procedure.

Algorithm 2: Tree Data Structure Mesh Refinement (2)

```
Function splitCellAndNeighbours($c$)
    $\vec{i} \leftarrow cells[c]$
    foreach non-boundary face $f$ of cell $c$ do
        $\vec{i}_n \leftarrow$ findNeighbor($\vec{i}$, $f$) // add $\pm 1$ to one integer
            coordinate
        $\vec{i}_p \leftarrow$ findParentCell($\vec{i}_n$) // use eq. 2.2
        $c_p \leftarrow cellPosition[\vec{i}_p]$
        if not $cellIsRefined[c_p]$ then $neighbors \leftarrow$ addToList($c_p$)
    end
    splitCell($c$)
    foreach member $k$ of list $neighbors$ do
        $c_p \leftarrow neighbors[k]$
        splitCellAndNeighbours($c_p$) // recursion
    end
return

Function splitCell($c$)
    $\vec{i} \leftarrow cells[c]$
    $cellIsRefined[c] \leftarrow true$
    $N \leftarrow$ totalNumberOfCells()
    foreach child $k$ do
        $\vec{i}_c \leftarrow$ findChildCell($\vec{i}$, $k$) // use eq. 2.1; k from fig. 2.4
        $cells[N+k] \leftarrow \vec{i}_c$
        $cellPosition[\vec{i}_c] \leftarrow N+k$
        $cellIsRefined[N+k] \leftarrow false$
        $cellTriangles[N+k] \leftarrow$ intersectionOrInclusion($cellTriangles[i]$)
    end
return
```

The mesh generator's speed depends on the size of the corresponding tree. For example, the wall-clock time for the mesh generation around the ONERA M6 wing, used for an inviscid flow simulation, is approximately $0.25\dot{1}0^6$ leaf cells/min. In contrast, the generation of meshes for internal aerodynamics is faster, exceeding the $2.5\dot{1}0^6$ leaf cells/min. because a reduced variation in cell size is expected. Software's efficiency is comparable with other Cartesian mesh generators found in literature, e.g., 10^6 cells/min. in [6]. Finally, fig. 2.5 shows the resulted mesh around an isolated airfoil. On its left, the shape of the airfoil and the given domain are presented, while on its right the final mesh after the implementation of Algorithm 1 is depicted.

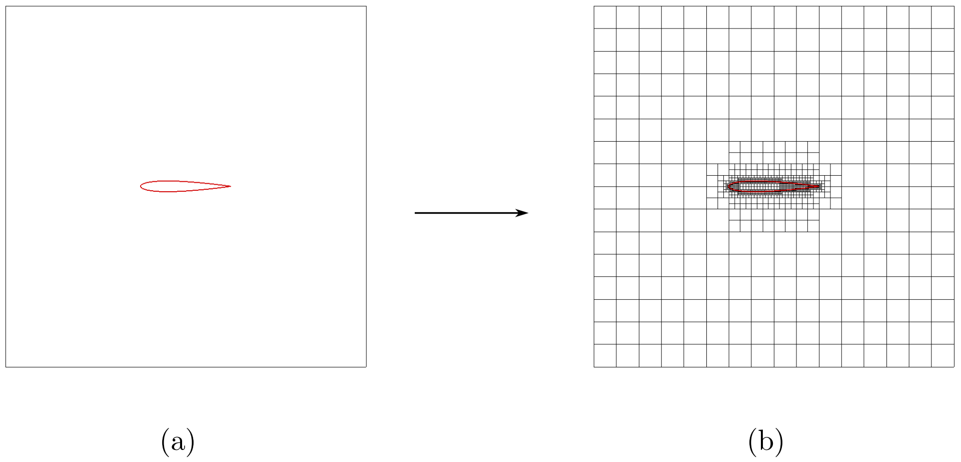


Figure 2.5: (a) An isolated airfoil located within a squared computational box. (b) The resulted mesh after the implementation of Algorithm 1.

2.3 Mesh Smoothing and Flow Adaptation

This section highlights two essential functionalities of regular body-conforming methods and proposes alternatives that mimic their behavior in the Cartesian mesh environment. Indeed, they increase the quality of the mesh by smoothing out the refinement level variations between different mesh regions and applying flow-field adaptation whenever considered necessary.

One major drawback of the mesh produced by Algorithm 1 is the abrupt cell's size growth with the distance from the wall, approximating the rate of a geometric progression law with a scale factor of 8 (or 4 in 2D). However, these sharp coarsening progressions adversely affect the convergence and accuracy of the flow solver, causing significant errors, especially in the proximity of large flow gradients. In contrast, conventional body-fitted mesh generators impose smooth cell size changes, especially in the development of boundary layers. Therefore, a smoothing is performed in the already generated mesh, reducing the differences between refined zones and increasing its quality.

The developed smoother applies Algorithm 1 to all leaf cells, taking advantage of its recursive nature to increase the procedure's efficiency. However, the criterion for the cell subdivision used in function "cellMustSplit" differs detecting cells larger than a predefined volume limit V_l, which are consecutively split until the condition is met. The criterion is applied to all non-cut-cells being closer to the geometry than a fixed value r_0. Moreover, volume V_l differs between cells, smoothly progressing from the size of cut-cells V_c to the maximum allowed cell size V_{max} according to the formula

$$V_l = fV_{max} + (1-f)V_c$$

where

$$f = -2\left(\frac{r}{r_0}\right)^3 + 3\left(\frac{r}{r_0}\right)^2, \quad r < r_0$$

Variables V_c and r represent each cell's distance from the wall and the volume of the closest cut-cell, respectively. However, the computation of their exact values is challenging. Several approaches have been proposed for the distance field computation such as the solution of the eikonal equation [281] or other attempts of equivalently high cost. Instead, a more efficient approximation method is preferred, explained in the following steps.

Firstly, it is considered that the distance of each cut-cell centroid from the closest wall is zero in the absence of more information at this stage of the mesh generation. On the contrary, the distance between cells is easily computed and temporarily stored in list "*distanceFromNeighbors*" applying eq. 2.5. Using this information, cells in the first layer away from the wall detect and store the closest neighboring cut-cell and the corresponding distance into lists "*closestCutCells*" and "*distanceFromWall*", respectively. Then, the process is repeated between cells of the first and second layers. A cell of the second layer approximates its wall distance by adding its distance from the closest neighbor of the first layer to the latter's already computed wall distance. The process gradually propagates to successive layers until all cells are reached. Algorithm 3 depicts the above method in a pseudocode form.

Two lines of Algorithm 3 need further explanation. Firstly, line 4 creates a list data structure with just one member deleting any previously stored information. Secondly, line 8 uses dynamical memory allocation to add new members to the list.

Finally, the body of function "minimumDistanceFromNeighboursFound" is shown in Algorithm 4.

Algorithm 3: Mesh Smoother (1)

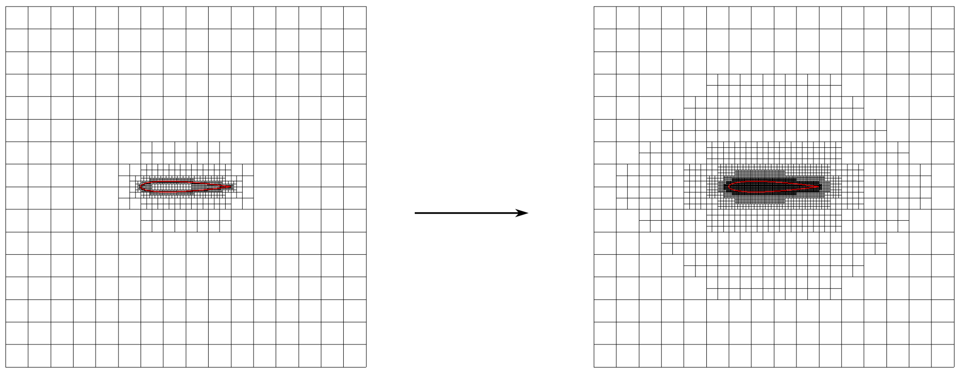
```
input  : $distanceFromNeighbors$
output: $distanceFromWall$, $closestCutCell$

Main Function meshSmoother()
    foreach intersected leaf cell $c$ do
        foreach neighbor $c_n$ of cell $c$ do
            $list \leftarrow$ createListWithOneMember($c_n$)
            foreach member $k$ of the list do
                $c_k \leftarrow list[k]$
                if minimumDistanceFromNeighborsFound($c_k$) then
                    $list \leftarrow$ addToListAllCellNeighbors($c_k$)
                end
            end
        end
    end
return
```

Algorithm 4: Mesh Smoother (2)

```
Function minimumDistanceFromNeighboursFound($c$)
    $cellDistanceIsRecomputed \leftarrow false$
    foreach neighbor $c_n$ of cell $c$ do
        $d \leftarrow distanceFromWall[c_n] + distanceFromNeighbours[c][c_n]$
        if $distanceFromWall[c] > d$ then
            $distanceFromWall[c] \leftarrow d$
            $closestCutCell[c] \leftarrow closestCutCell[c_n]$
            $cellDistanceIsRecomputed \leftarrow true$
        end
    end
return $cellDistanceIsRecomputed$
```

Fig. 2.6 shows the modification caused by Algorithm 3 to the mesh of fig. 2.5b.



(a) (b)

Figure 2.6: (a) The mesh around an isolated airfoil originated from fig. 2.5b. (b) Resulting mesh smoothened by Algorithms 1 and 3.

Another essential feature of mesh generators is their ability to perform local mesh enrichment triggered by characteristics of the current flow solution. Consequently, cells are added to regions where an increased resolution is required, improving the simulation's accuracy. Contrary to body-fitted meshes, solution-based refinement in the proposed mesh generator is straightforward by taking advantage of the tree data structure. In particular, after specifying a region of high interest, cells are further subdivided by simply creating new sub-branches in the already developed tree.

Hence, in addition to geometric refinement, described in section 2.2, flow-field refinement is also possible by further exploiting Algorithm 1 supported with different subdivision criteria. The new conditions employ flow sensors to detect specific physical flow phenomena and activate cell refinement. Mesh adaptation is performed several times during the flow simulation. Each attempt takes place only after the solution is sufficiently converged, providing a trustworthy approximation of the flow solution. After the mesh enrichment is completed, the flow field is transferred to the newly adapted mesh, and the solution process continues until convergence criterion is met.

The sensors developed in this thesis detect shock waves and viscous wakes. Grid adaptation along discontinuities is firstly discussed. A cell is flagged for refinement under the condition

$$\left|\frac{\vec{v}\cdot\vec{n}}{c}-1\right|<\epsilon$$

It detects cells placed in a zone of width ϵ close to the normal Mach number's iso-surface of unity computed along the perpendicular to the shock direction, expressed by the unitary vector $\vec{n}$ [187]. This vector is parallel to the local pressure gradient, which implies

$$\vec{n}=\frac{\nabla p}{|\nabla p|}$$

The notation of the above flow quantities is explained in chapter 3. Due to the absence of the converged flow field, errors in the pressure gradient computation lead to false shock wave detection. To overcome this issue, the filtering

$$|\nabla p|<\omega|\nabla p|_{max} \tag{2.7}$$

is applied, where ω is a user-defined factor [348]. Additionally, in unsteady flows, a correction is needed to detect the transient shock wave. Therefore, as proposed by [187], an additional term is introduced to the aforementioned relation,

$$\left|\frac{1}{c}\frac{1}{|\nabla p|}\frac{\partial p}{\partial t}+\frac{\vec{v}\cdot\vec{n}}{c}-1\right|<\epsilon$$

The detection of the viscous wake, characterized by the presence of flow recirculation, is more complicated. Many sensors have been proposed, like the helicity method [178], the λ_2 criterion [143], or the vorticity indicator [83]. An overview of existing detection methods is given in [146]. However, this study uses a simpler and more efficient criterion, in terms of computational cost, based on the total pressure (p_t) drop in regions with high viscous effects [214]. It is expressed as

$$p_t<\eta p_{t_\infty}$$

where p_{t_∞} is a user-defined threshold usually chosen equal to the far-field total pressure. Filter 2.7 is additionally used. Moreover, the reduction in η at each successive adaptation improves the sensor's behavior.

2.4 The Cut-Cell Generation

It has already been emphasized that the great advantage of the cut-cell method is the automatic and fast Cartesian mesh generation. Indeed, the process developed in the previous sections is very efficient and independent of the geometry's complexity having also low memory requirements. However, the cost to pay for its simplicity is the computation of the intersection between the solid surface and the Cartesian background mesh, making the cut-cells construction the most crucial part of the mesh generation. However, robust algorithms available in the literature deal efficiently with this challenging task, simplifying and automating the intersection procedure.

Initially, the appropriate terminology adopted in this thesis is introduced starting from clarifying the term "Cartesian cell". This cell is part of the background mesh and has the shape of a rectangular cuboid. Whenever intersected by the solid surface, its fluid part is occupied by the corresponding cut-cell. Fig. 2.10d illustrates an arbitrary cut-cell consisting of Cartesian-directed faces and boundary cut-faces colored blue and red, respectively. The first set's faces are part of the Cartesian black-colored background mesh and are called fluid faces. On the other hand, the faces of the second class are part of the triangulated surface and are mentioned as solid faces. Since both the Cartesian cell and the triangles are convex, their intersection produces convex solid faces. In contrast, fluid faces may be convex or concave.

Similarly, the nodes forming the cut-cell are split into two categories. Those belonging to the initial hexahedral cell are denoted as fluid nodes and are highlighted in fig. 2.10c. The rest are marked in blue in fig. 2.10a and are called solid nodes. The latter are subdivided further into two categories, the external nodes, which are part of at least one fluid face, and the internal nodes, which are located in the interior of the Cartesian cell. Finally, the consecutive line segments linking external nodes of the same external face comprise a fluid polyline. Fig. 2.10b depicts a cut-cell with four fluid polylines plotted with blue.

In the developed mesh generation method, the already constructed background mesh facilitates the cutting process. In particular, the tree data structure provides each cut-cell with the geometry facets needed for the intersection computation. Moreover, for simplicity reasons, each cut-cell is generated separately from the rest, paying the extra cost of determining twice the shape of faces between neighboring cut-cells. Hence, the developed algorithm focuses on the intersection of a single Cartesian cell by an arbitrary triangulated surface. Its purpose is to construct the resulting polyhedron and compute all the necessary topological information for the flow simulation.

Although the concept of such an intersection algorithm may be straightforward, its implementation is delicate. In a general case, the topology of a cut-cell may become very complex, containing dozens of nodes and faces. Thus, the proposition of a robust algorithm considering all possible geometric cases, including the hexahedron's separation into more than one discrete finite volume, is of paramount importance. The proposed method is partly based on techniques from the field of computer graphics [226], and its consecutive steps are described in the following subsections. Finally, the introduction of some assumptions considerably simplifies the cut-cell construction. Under this point of view, a second algorithm has been developed, which requires low computational and memory resources, presented in Appendix B.

2.4.1 The Construction of Solid Faces

The first step in the cut-cell construction algorithm is forming the solid faces by cutting off the surface triangles' parts that extend beyond the Cartesian cell's volume. The cutting procedure implemented in this thesis is based upon the concept of polygon clipping [6], which in 2D indicates the process where a square-shaped Cartesian cell acts as a window, and its target is to compute the visible parts of a polygon through it. Among various alternatives found in the literature [180], [340], this section applies a robust and straightforward algorithm proposed by Sutherland and Hodgman [296], which has the attractive property that the returning polygon keeps the initial order of its vertices. Moreover, its only requirement is for the clip window to be convex, making it suitable for trimming the solid triangular facets protruding from the cell's region.

The procedure will be firstly explained in 2D and then extended to the more complicated 3D case for the reader's convenience. The algorithm's key feature is the division of the clipping operation into a sequence of simpler problems by implementing a loop over the edges of the rectangular cell, focusing each time on the relative position between the polygon and the Cartesian edge. Fig. 2.7 illustrates the iterative process through four successive steps. During each step, the corresponding

Cartesian edge is extended infinitely in both directions, splitting the 2D space into two areas, the visible and the invisible one. Then a nested loop sweeps over all segments of the polygon. If a segment crosses the extended edge, the intersection's point will be added as a new vertex to the polygon, but if a segment lies entirely in the invisible area, then it will be discarded. Therefore, a new polygon is created after each iteration, which is then imported to the next one. Finally, once all edges have been processed, the resulting polygon will entirely be placed inside the Cartesian cell.

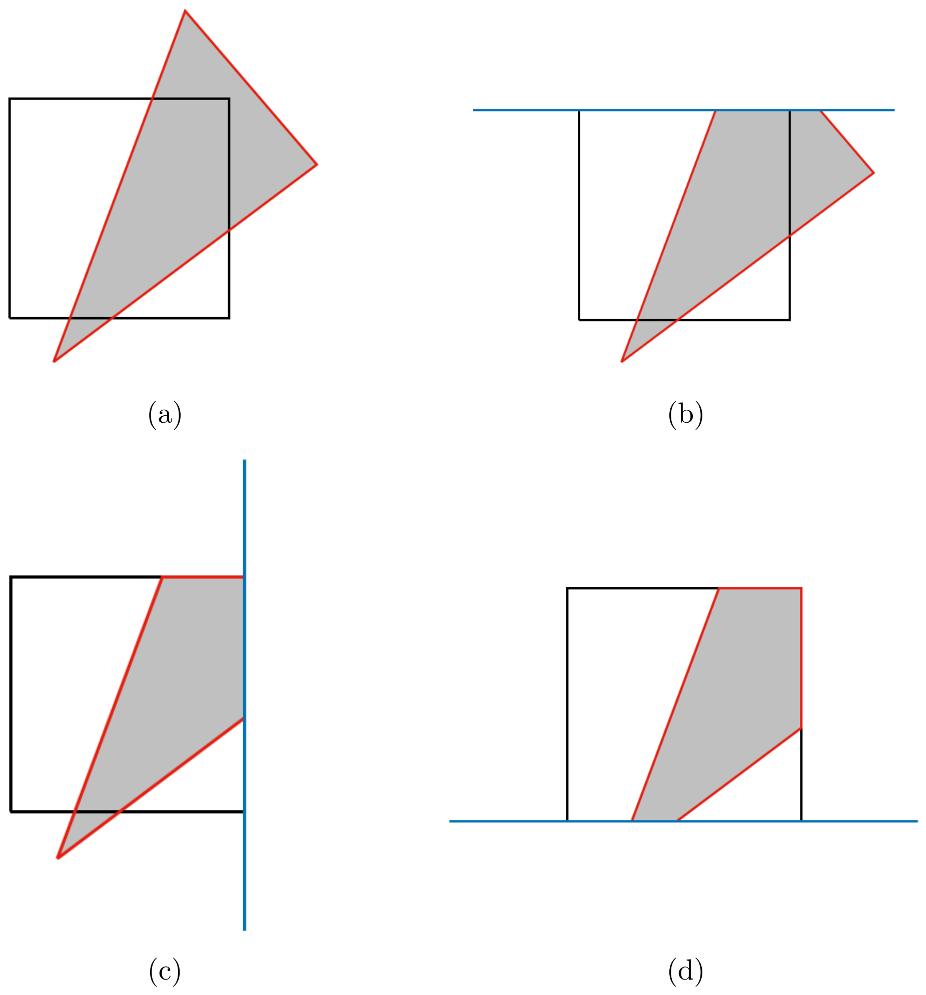


Figure 2.7: The geometric representation of the Sutherland and Hodgman polygon clipping algorithm. Each edge of the black square is infinitely extended and the non visible part of the triangle is cropped.

Algorithm 5 presents the generalization of the previously described procedure in 3D. It is adjusted in the case where the viewing window is a rectangular cuboid. Function "vertexIsVisible" checks the relative position of the polygon's vertices against the infinite extension of face f. Thus, fast spatial comparison operators are introduced to increase the algorithm's efficiency. These are based on outcode flags associated with each vertex location with respect to the hexahedron. The outcodes' definition is inspired by the study of crystalline structures and is discussed in detail in [6]. However, these operations are prone to roundoff errors, especially for vertices placed very close to the face's plane. Therefore, the plane is slightly displaced towards the invisible subspace to encompass the questionable vertices to the visible area making the algorithm less dependent on the computers's precision.

Algorithm 5: Sutherland-Hodgman Clipping Algorithm

```
input  : $polygon$
output: $polygon$

// every enumeration starts from zero
1  foreach face $f$ of the rectangular cuboid do
2      foreach vertex $v_1$ of polygon do
3          $N \leftarrow$ totalNumberOfVertices($polygon$)
4          $v_2 = (v_1 + 1)$ mod $N$ // next vertex of $v_1$
5          if vertexIsVisible($f$, $v_1$, $polygon$) then
6              $polygonNew \leftarrow$ addVertexToPolygon($v_1$, $polygon$)
7              if not vertexIsVisible($f$, $v_2$, $polygon$) then
8                  $p \leftarrow$ findIntersection($f$, $v_1$, $v_2$, $polygon$)
9                  $polygonNew \leftarrow$ addVertexToPolygon($p$)
10             end
11         else if vertexIsVisible($f$, $v_2$, $polygon$) then
12             $p \leftarrow$ findIntersection($f$, $v_1$, $v_2$, $polygon$)
13             $polygonNew \leftarrow$ addVertexToPolygon($p$)
14         end
15     end
16     $polygon \leftarrow polygonNew$
17     $polygonNew \leftarrow$ clearMemory()
18 end
```

2.4.2 The Construction of Fluid Faces

The construction of the solid faces, shown in subsection 2.4.1, is only a part of the cut-cell creation process. The reason is that although Algorithm 5 sufficiently computes the vertices' coordinates of each solid face in the correct order, it does not give any other connectivity information. Hence, the target of the present subsection is the computation of the resulting polyhedron's topological features, which are summarized as:

1. A vertex-coordinates mapping, which connects each vertex index with its spatial coordinates avoiding the existence of coincident vertices
2. A face-vertices mapping, which describes the faces by their vertices in a counterclockwise order
3. A face-cells mapping, which declares the cell in which each face belongs

The last mapping is useful only when a Cartesian cell is split into more than one cut-cells.

The developed method initially focuses on solid faces. Firstly, it separates the internal and external nodes by comparing the coordinates of each solid node, computed by Algorithm 5, with the coordinates of the hexahedron's faces given by eqs. 2.4 and 2.5. A useful sideproduct of the comparison is identifying the face on which each external node is laid. These belonging to more than one faces are located at the edges of the Cartesian cell and flagged as edge nodes to be easily accessed in a subsequent step of the algorithm.

Next step deals with the multiple storage of each solid node's coordinates resulting from the clipping algorithm's implementation in each triangle separately. Thus, each node is stored in memory as many times as the number of solid faces it belongs to. The unification of internal nodes is straightforward because they are usually numbered by the surface's data structure on which they belong. On the other hand, all external nodes arise from the intersection of a surface triangle with a Cartesian cell, and their unique identification is not straightforward.

To this end, a loop over the solid faces is implemented, searching for edges placed on the faces of the hexahedron. Each one of these edges represents a directional line segment being part of a fluid polyline. Fig. 2.8a depicts a Cartesian face, which is cut by an arbitrary solid surface. The intersection results in edges computed from Algorithm 5 as parts of the solid faces and represented by red segments directed from the empty to the filled cycle. Then, the edges are linked to construct the fluid polylines for each face. The process successively connects the ending node of each edge with the beginning of the next one by detecting coincident nodes, fig. 2.8b. The comparison of nodes' coordinates is prone to roundoff errors, and thus, avoided. Instead, identical nodes are verified by checking the identification integer number of the surface mesh's edge to which each node belongs. The formation of each fluid polyline starts and ends to a node located at an edge of the hexahedron. By the end of this process, all double-stored external nodes have been discarded.

Until now, the solid faces have been formed, and all internal and external nodes are uniquely stored in memory, resulting in the vertex-coordinates mapping. Next, the fluid faces are formed. The first step is to put all nodes located at the boundary, i.e., the edge nodes, of each Cartesian face in the correct order forming a list starting from a node that indicates the beginning of a fluid polyline. The order follows the orientation of the face's boundary which is defined by the right-hand rule such that the normal to the face points always outwards. In the example of fig. 2.8c, the boundary's orientation is counterclockwise, represented by black arrows. Moreover, the correct numbering of the edge nodes is also shown in this figure.

Next, the fluid nodes of each Cartesian face are detected. It is done by sweeping over the members of the before-mentioned list and classifying the four corner nodes by changing the status from solid (represented as a yellow dot) to fluid (represented as a blue dot) each time a solid node is met. For example, in fig. 2.8c, the process starts from node 1. If the next node in the list is one of the square's four corners, it will be labeled solid. Instead, node 2 is detected, changing the status from solid to fluid which means that node 3 is a fluid node.

Then, the formation of fluid faces is possible by iterating over the edge nodes and creating polygons by using the fluid nodes and fluid polylines. In fig. 2.8c, the process starts from node 1. The red arrows guide the algorithm through the fluid polyline to node 7. Then, the black arrow indicates the next node identified by index 8. The process continues till node 1 is met. The set of nodes defining this route forms a fluid face of the cut-cell. The process continues with the next edge node that is not included in the already defined fluid face, i.e., node 2. The two routes are presented with blue in fig. 2.8d. Finally, the detected faces are blue-colored in fig. 2.8e. By the end of this stage, the face-vertices mapping is completed for all faces of the cut-cell.

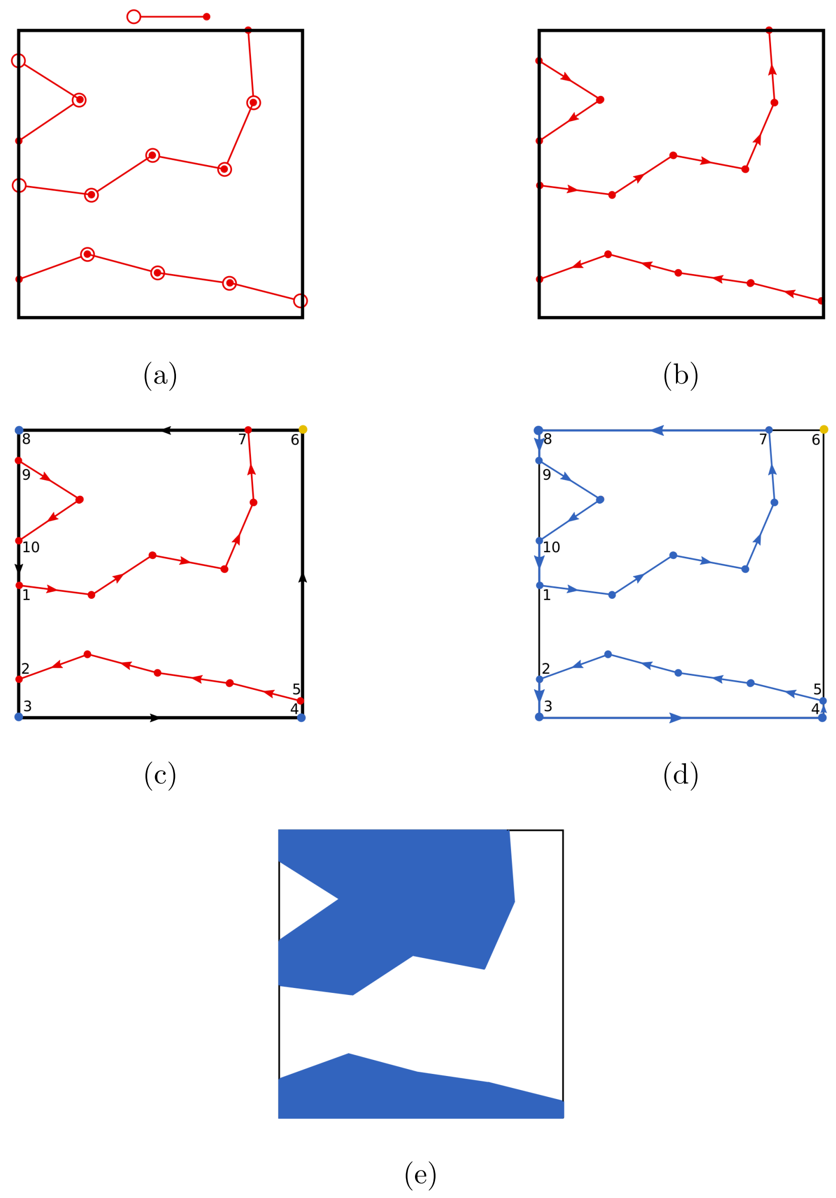


Figure 2.8: Construction of fluid faces in a Cartesian face (black square). The resulted intersection of the Cartesian face with a single triangle of the solid surface is represented by a red line connecting an empty with a filled cycle and is illustrated in (a). The process starts from (a) by linking consecutive red edges and ends to (e), showing the resulting faces with blue.

Subsequently, the face-cells mapping is created when the intersection between the Cartesian cell and the wall results in multiple cut-cells. Firstly, the solid nodes constructing each cut-cell are identified. The process starts by considering that an arbitrary solid node is part of the first cut-cell. Then, all the solid nodes connected to it belong to the same cell. The process continues by marking the neighbors of the just specified nodes until no other node can be added to this set. Next, a non-identified arbitrary node is considered part of the second cell, and the nodes' designation continues until all solid nodes are classified. Afterward, fluid nodes are labeled, which is done by visiting fluid faces containing at least one solid node and listing the rest of the nodes to the same cell. This method also corrects the label of solid nodes which are part of the same cell without sharing the same face. After classifying the solid and fluid nodes to the corresponding cells, each face is linked to the cell in which its nodes belong.

2.4.3 Illustration of the Cut-Cell Construction

This subsection discusses some examples of the cut-cell procedure offering a practical view of the methods explained in the previous subsections. Firstly, the intersection between a cube and a triangulated surface, shown in fig. 2.9a, is studied. The Sutherland–Hodgman algorithm is implemented for each red triangle to crop their part located outside the black cube. Different stages of the process are presented in fig. 2.9 describing the trimming of each triangle from each cube face. The final surface shape, shown in fig. 2.9p, is the input of the algorithm explained in subsection 2.4.2. Its four distinct steps are presented in fig. 2.10. The first step, fig. 2.10a, discards the duplicated internal nodes, marked in blue, while the second step, fig. 2.10b, generates the fluid polylines, highlighted with blue, resulting in a uniform node numbering. The eight vertices of the cube are labeled as fluid or solid, which are indicated with blue or yellow, respectively, in fig. 2.10c. Finally, the fluid faces of the cut-cell are defined, which are colored in blue in fig. 2.10d. The resulted polyhedron is appropriately shaded in fig. 2.11, giving a better perspective in its 3D shape.

Subsequently, the developed method's capabilities are tested on various demanding cases, where the resulted cut-cells are considered invalid by most mesh generators. This is because these cells usually cause complexities in their construction and difficulties in ascertaining their neighbors. Therefore, most software avoid dealing with these cases by applying local mesh refinement. However, this treatment increases the flow simulation cost without guaranteeing the problem solution because degenerate cells may still exist even in very fine meshes. In contrast, the developed cutting process successfully handles these odd cases, avoids any further mesh refinement, and ensures the mesh generation's robustness.

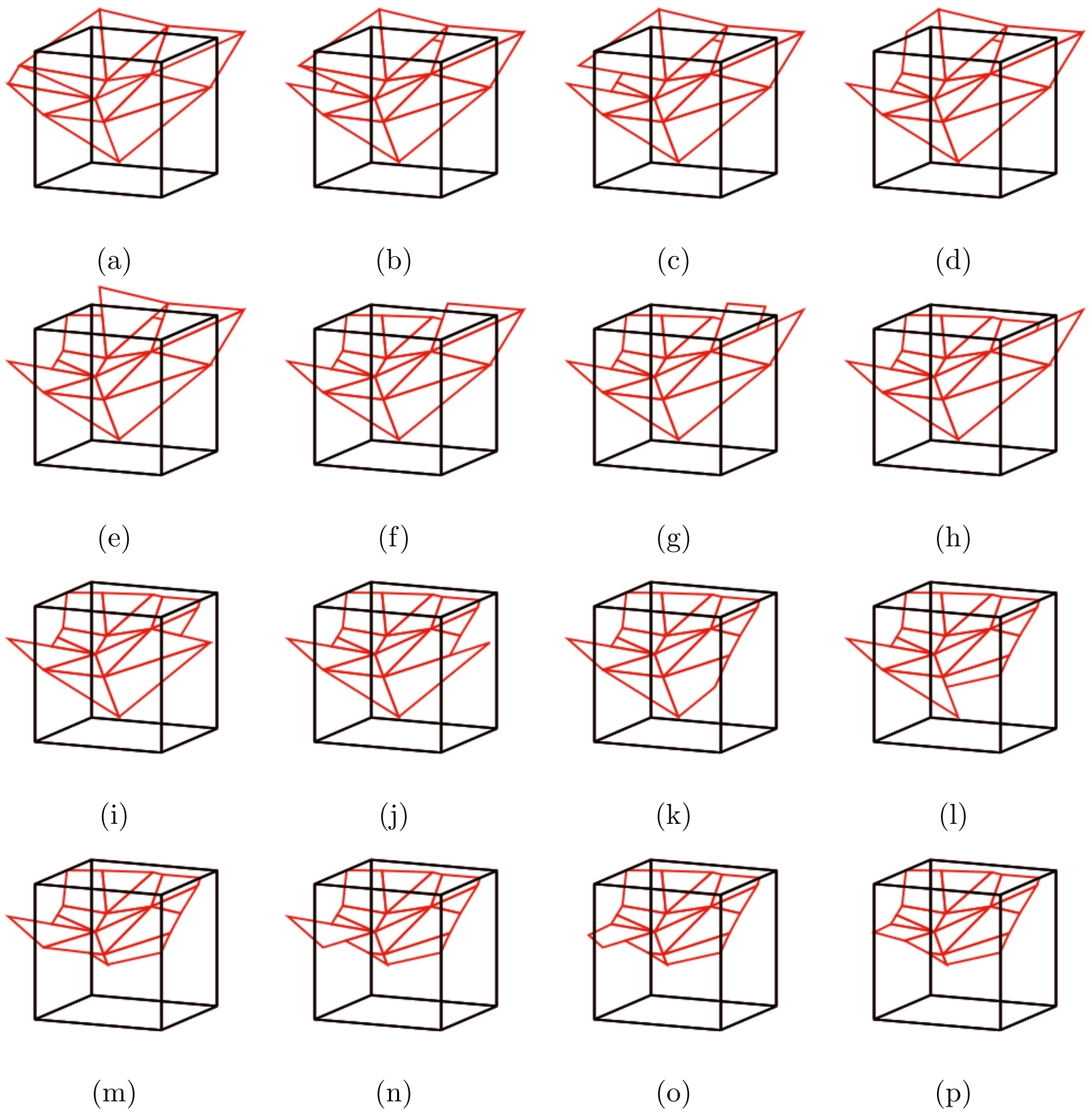


Figure 2.9: The red triangulated surface is clipped into the black cube by implementing the Sutherland–Hodgman algorithm in each triangle.

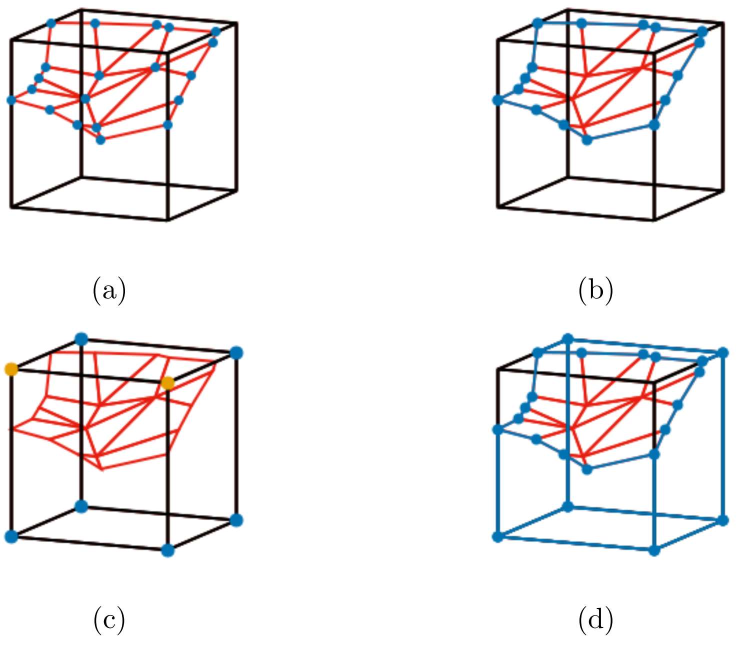


Figure 2.10: Four stages of the cut-cell construction starting from the result of the clipping algorithm shown in fig. 2.9p.

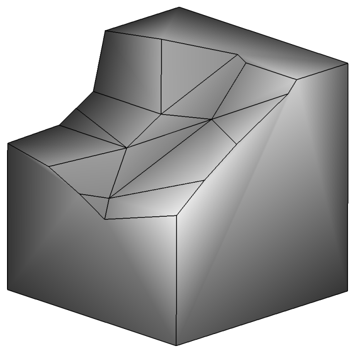
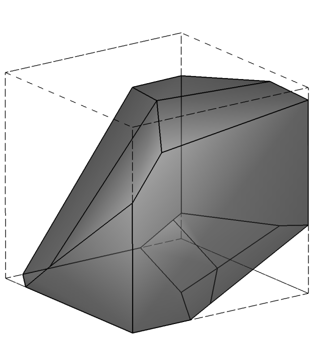
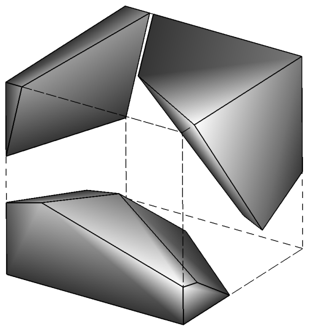
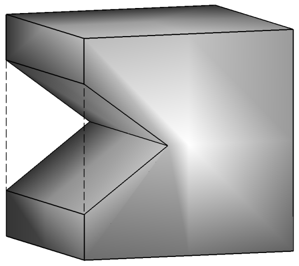


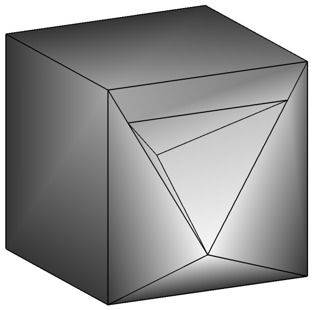
Figure 2.11: The appropriately shaded cut-cell of fig. 2.10d.

The presented cases are collected in fig. 2.12. In the first case, fig. 2.12a, the embedded boundary intersects the Cartesian cell at two regions. It usually happens around solid boundaries of high curvature or when two bodies tend to touch each other. On the other hand, fig. 2.12b shows a Cartesian cell divided into three separate cut-cells. This case usually appears at the trailing edge of a wing, where a hexahedron splits into two polyhedral. The appearance of cut-cells like the one depicted in fig. 2.12c is also typical in the vicinity of the trailing edge. A 3D version of a wedge penetrating the cube from the front face is illustrated in fig. 2.12d. In this case, a hole is formed in the middle of the face, which may hinder the convergence of the flow solver. Therefore, it is split into four convex sub-faces. Finally, fig. 2.12e shows an exotic geometric construction, where a wedge passes through the entire cell. It may happen close to sharp nibs.


(a)


(b)


(c)


(d)

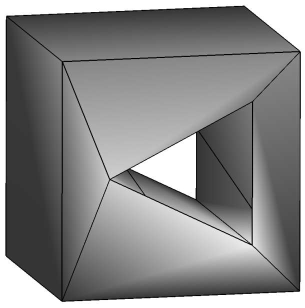

(e)

Figure 2.12: Demanding cases successfully treated by the developed cut-cell generator. (a) A cell is cut into two different regions. (b) Three cut-cells are originated from a single Cartesian cell. (c) A wedge cuts the left-hand side of a cube. (d) The corner of a tetrahedron penetrates the front face of a cube. (e) A solid body passes through the entire cube.

2.5 The Face-Based Mesh Data Structure

The developed tree data structure described in section 2.2 is convenient for the mesh's generation, but its practicality during the flow solution is questionable due to its inefficient mesh connectivity computation. Therefore, another alternative is adopted, which combines the flexibility of a face-based data structure, commonly used in unstructured meshes, with the special nature of Cartesian meshes leading to a highly compact data set.

This section presents the development of the face-based data structure starting from the storage of the cell connectivity derived from the tree data structure. Additionally, an algorithm computing the face-nodes and face-cells mapping considering all cells of the mesh is proposed. This information enables the detection of cells entirely covered by the solid geometries, which are discarded from the domain. Finally, a collection of geometric quantities, necessary for the performance of the flow-solver and the post-processor, is computed, taking advantage of the hexahedral shape of most cells.

2.5.1 The Cell-to-Cell Connectivity

The discussion below presents the algorithm detecting each cell's neighbors. Storing the tree in the manner described in section 2.2 gives important information for obtaining cell connectivity. Furthermore, provided that the difference in refinement levels between neighbors is at most one, the searching algorithm avoids the time-consuming traversing of the entire tree.

For the sake of clarity, the proposed algorithm is exemplified using a case study, where the neighbors along the positive half x-axis of an arbitrary cell (i, j, k) are requested. Firstly, the algorithm looks for a neighbor at the same refinement level, indexed as $(i{+}1, j, k)$. If the cell does not exist, its parent $(int[(i{+}1)/2], int[j/2], int[k/2])$, eq. 2.2, is the requested neighbor. On the other hand, if cell $(i+1, j, k)$ exists, "*cellPosition*" computes the entry key to list "*cellIsRefined*", computed by Algorithm 1, giving information about its refinement status. The cell is a neighbor only if not further subdivided. Otherwise, four of its eight children are adjacent to cell (i, j, k), which are identified by function "child", presented below. The whole process is carried out in Algorithm 6.

Algorithm 6: Neighbors Detector (1)

```
input  : cell $c$, face $f$
output: neighbors $c_p$ (coarser level) or $c_n$ (same level) or $c_c$ (finer level)

1  Main Function neighborDetector($c$, $f$)
       // $c$:  position of cell in list ‘‘cells’’
       // $f$:  index declaring one of the 6 faces of c, $f \in [0,5]$
2      if face $f$ is part of the mesh boundary then return Boundary Flag

3      $\vec{i} \leftarrow cells[c]$
4      $\vec{i}_n \leftarrow$ neighbors($\vec{i}, f$)
5      $\vec{i}_p \leftarrow$ findParentCell($\vec{i}_n$) // use eq. 2.2
6      $c_p \leftarrow cellPosition[\vec{i}_p]$
7      if $cellIsRefined[c_p]$ is false then return $c_p$

8      $c_n \leftarrow cellPosition[\vec{i}_n]$
9      if $cellIsRefined[c_n]$ is false then return $c_n$

10     $f_{op} \leftarrow oppositeFace[f]$
11     foreach face segment $f_s \in [0,3]$ of $f$ do
12         $\vec{i}_c \leftarrow$ child($\vec{i}, f_{op}, f_s$)
13         $c_c[f_s] \leftarrow cellPosition[\vec{i}_c]$
14     end
15 return $c_c$
```

Below, some details of Algorithm 6 are further explained. For this purpose, specific names are given to each of the 6 faces of a Cartesian cell. The two faces normal to the Cartesian x-axis are called F_{west} and F_{east}, where the abscissa of the first is smaller than that of the latter. Similarly, F_{south} is opposite to F_{north} along the y-direction, and F_{bottom} is opposite to F_{top} and both are normal to the z-axis.

Initially, the algorithm checks the existence of a neighboring cell in line 2. In other words, it confirms that the cell indexed as (i, j, k) is not at the edge of the mesh bounding box (B). These cells are detected by the following conditions,

$$\begin{gathered} i = i_{min} \Leftrightarrow F_{west} \in B, \quad i = i_{max} \Leftrightarrow F_{east} \in B, \\ j = j_{min} \Leftrightarrow F_{south} \in B, \quad j = j_{max} \Leftrightarrow F_{north} \in B, \\ k = k_{min} \Leftrightarrow F_{bottom} \in B, \quad k = k_{max} \Leftrightarrow F_{top} \in B \end{gathered}$$

where

$$i_{min} = j_{min} = k_{min} = 2^L$$
$$i_{max} = j_{max} = k_{max} = 2^{L+1} - 1$$

Subsequently, Algorithm 7 presents the two functions used in lines 4 and 12.

Algorithm 7: Neighbors Detector (2)

```
1  Function neighbors($\vec{i}$, $f$)
2      $dim \leftarrow faceDimension[f]$
3      $dir \leftarrow faceDirection[f]$
4      $\vec{i}_n \leftarrow \vec{i}$
5      $i_n[dim] \leftarrow i_n[dim] + dir$
6  return $i_n$
7  Function child($\vec{i}$, $f$, $f_s$)
8      $c \leftarrow childrenInEveryDirection[f][f_s]$
9      foreach dimension $d \in [0, 2]$ do
10         $m \leftarrow childIdentification[c][d]$
11         $i_c[d] \leftarrow 2i[d] + m$
12     end
13 return $\vec{i}_c$
```

Finally, the used matrices are

$$faceDimension = [1 \;\; 1 \;\; 0 \;\; 0 \;\; 2 \;\; 2]$$
$$faceDirection = [+1 \;\; -1 \;\; +1 \;\; -1 \;\; +1 \;\; -1]$$
$$oppositeFace = [1 \;\; 0 \;\; 3 \;\; 2 \;\; 5 \;\; 4]$$

and

$$childrenInEveryDirection = \begin{bmatrix} 2 & 3 & 6 & 7 \\ 0 & 1 & 4 & 5 \\ 3 & 1 & 7 & 5 \\ 2 & 0 & 6 & 4 \\ 6 & 7 & 4 & 5 \\ 2 & 3 & 0 & 1 \end{bmatrix}, \quad childIdentification = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.8}$$

The children's numbering, used in matrix "$childrenInEveryDirection$", follows the instructions of fig. 2.4.

2.5.2 Numbering of Nodes and Faces

At this point of the mesh generation, the shape of each cell is well-defined, nodes and faces determining each cell are locally numbered, and neighbors of each cell have been stored. However, the global specification of each face through the face-cells mapping is fundamental for the developed flow solver's functionality. Moreover, the global numbering of the mesh nodes is necessary for post-processing purposes. This being said, the global numbering of faces and nodes is studied below. It concerns the fluid and solid part of the mesh and is applied only to leaf cells. Considering the high complexity of the developed algorithm, only its basic structure is explained for the reader's convenience.

For the sake of the following discussion, the concept of hanging nodes, edges, and faces is introduced. They appear when a cell c is adjacent to four smaller in size cells. The area between them, shown in brown in fig. 2.13a, is described by one or four rectangular faces depending on the viewpoint of each side. The face belonging to c, occupying the whole brown area, will be called hanging face. Moreover, the term "hanging node" refers to the common node of the four neighboring cells lying on the hanging face. An example of such a node is shown with blue in fig. 2.13b. "Hanging edge" is called a segment on a hanging face defined by two nodes, one of which is hanging. Fig. 2.13b illustrates the point by plotting four hanging edges in red.

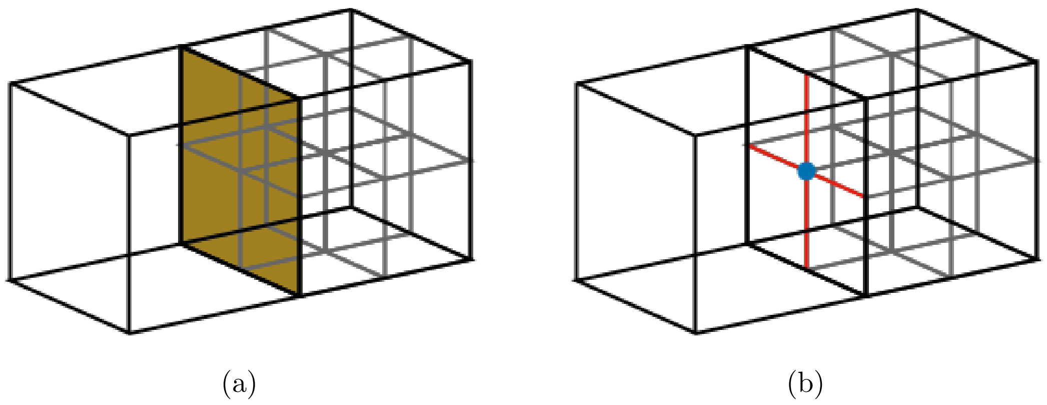


Figure 2.13: (a) The large cell on the left is adjacent to four smaller cells on the right side. The hanging face in brown belongs to the large cell. (b) A "hanging node" (blue) and four "hanging edges" (red) are formed in the border between the two sides.

The numbering process starts from the nodes. Firstly, the fluid nodes are numbered by looping over the faces of each Cartesian cell and giving the same identification number to coincident nodes of neighboring cells. Special treatment is needed for each hanging node which is never part of both neighboring cells of different sizes. Moreover, a repetitive process is necessary for common nodes between non-neighboring cells since the only available information concerns the direct neighbors. After that, the external solid nodes are numbered. They always belong to the border between two cut-cells. A typical neighborhood of cut-cells is shown in fig. 2.14. The identical nodes are detected without comparing their spatial coordinates, which could affect the mesh generator's robustness. Instead, each node placed in the interior of a cell's face is represented by the index number of the solid surface's edge at which it is located. On the other hand, two nodes placed at the same Cartesian edge are compared by using the index number of the surface triangle, at which they belong. Finally, numbering the solid nodes is straightforward since no comparison between nodes is needed.

Thereafter, the numbering of faces follows, giving rise to the face-vertices and face-cells mapping. Faces belonging to the bounding box of the mesh and the solid faces are numbered in a straightforward manner. However, this is not the case for internal faces, which are classified into three categories, depending on the nature of the adjacent cells. In the first case, faces between Cartesian cells are always rectangular and easily detected by using list "*neighbors*" created by Algorithm 6. Moreover, each hanging face is rejected and replaced by the four equally sized faces of the neighboring cells.

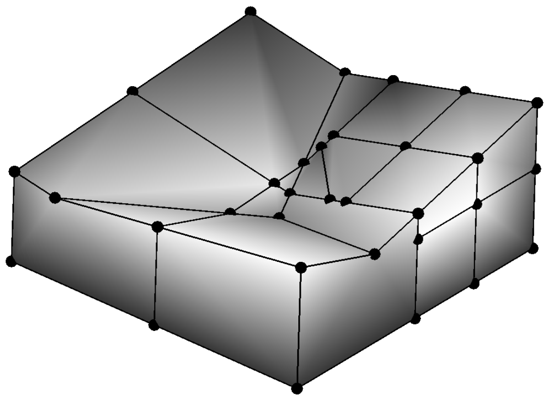


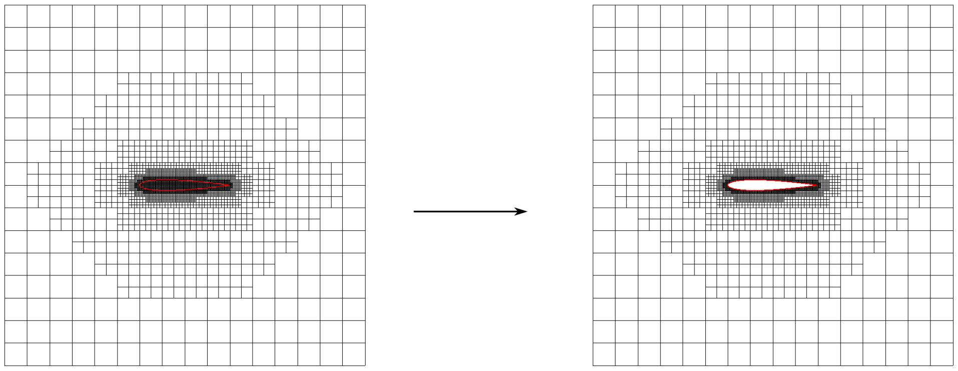
Figure 2.14: Neighborhood of 11 cut-cells of different shapes and refinement levels. Solid nodes shape the upper side of the cells.

The second category refers to faces between cut and non-cut Cartesian cells. The shape of these faces is still rectangular, but attention should be paid to cases where the intersected Cartesian cell is separated into more than one cut-cells. Then the face-cells mapping, presented in subsection 2.4.2, suggests in which of them the specific face belongs. Finally, the last category contains all the intersected by the solid surface faces. Such an example is shown in fig. 2.8e. The two neighboring cut-cells share all faces plotted in blue, which are matched by comparing them node-by-node. The process becomes more complicated when the adjacent Cartesian cells are of different refinement levels. Then, hanging nodes and solid nodes located along handing edges are only part of one of the two identical shapes, and thus, should be excluded from the face comparison procedure.

2.5.3 Detection of fluid cells

The cells of a Cartesian mesh belong either into the fluid or the solid region. Depending on the application, the solid part may also participate in the simulation process. For example, in applications involving Fluid-Structure Interaction (FSI), the body's deformation is computed by solving the appropriate PDEs in the solid part of the mesh. In particular, the direct collaboration of the solid and fluid regions through a sharp interface in a single fixed reference mesh makes the cut-cell method a valuable tool for simulating such phenomena. An extensive overview of IB approaches used to solve FSI problems can be found in [289]. Another application that requires the fluid-solid coupling is the numerical simulation of the Conjugate Heat Transfer (CHT) problem which consists of the heat conduction on a solid body, the heat convection in the surrounded fluid, and their thermal interaction. A solution to this problem based on the cut-cell method is explained in [230].

Moreover, in unsteady phenomena around moving bodies, which are also studied in the current thesis, the geometry covers and uncovers cells modifying their nature from solid to fluid and vice versa. Therefore, the solid part of the mesh is still necessary for the extrapolation of the flow field between successive time steps. A detailed algorithm dealing with these cases is described in [277]. However, the governing equations are solved only in the fluid part of the mesh, and thus, in this thesis, the solid part is excluded from the domain, erasing the corresponding part of the computer memory. The method described below aims to detect the fluid cells by implying a moving-front method. Fig. 2.15 shows the method's effect on a mesh around an isolated airfoil.



(a) (b)

Figure 2.15: (a) The mesh around an isolated airfoil, shown also in fig. 2.6b, is generated by Algorithm 3. (b) Resulting mesh after the rejection of its solid part.

A Cartesian cell is identified as solid when it lies completely inside the grid's solid region. The rest comprise the fluid cells classified as cut-cells and Cartesian cells entirely included in the fluid region. The next process marks all fluid cells starting from the geometry's boundary and then propagating to the interior. Firstly, a front of Cartesian cells, which lie in the fluid region and are neighbors of cut-cells, is formulated. This is possible by looping over the fluid faces of each cut-cell and checking their vertices. If a vertex is labeled “fluid” from the algorithm of subsection 2.4.3 and illustrated in fig. 2.10c, the neighboring Cartesian cell adjacent to the face is added to the front. Then, the front is moving to its new position comprised of the neighboring Cartesian cells of the first front cells. The process continues until the front’s size becomes zero. Finally, Algorithm 8 presents the method in a pseudocode form.

Algorithm 8: Detector of Cartesian Fluid Cells

```
input  : mesh topology
output: fluid cells

// initialize front
1  $front$ ← createEmptyList()
2  foreach $cut$-$cell$ $c$ do
3      foreach $face$ $f$ $of$ $c$ do
4          if $face$ $has$ $at$ $least$ $one$ $fluid$ $vertex$ then
5              $c_n$ ← neighborOfCell($c$, $f$)
6              $front$ ← addCellToFront($c_n$)
7              flagFluidCell($c_n$)
8          end
9      end
10 end
// move front
11 while $size$ $of$ $front$ $is$ $greater$ $than$ $zero$ do
12     $frontNew$ ← createEmptyList()
13     foreach $member$ $k$ $of$ $the$ $front$ do
14         $c$ ← $front[k]$
15         foreach $neighbor$ $c_n$ $of$ $c$ do
16             if $c_n$ $has$ $not$ $been$ $already$ $flagged$ then
17                 $frontNew$ ← addCellToFront($c_n$)
18                 flagFluidCell($c_n$)
19             end
20         end
21     end
22     $front$ ← $frontNew$
23 end
```

2.5.4 Computation of the Finite Volume's Geometric Quantities

The computation of the finite volumes' geometric quantities completes the creation of the mesh data structure. The term "geometric quantities" encapsulates all entities participating in the flow equations' discretization scheme, which are the area A_f, unit normal vector $\hat{\vec{n}}^f$, and centroid $\vec{x}^f$ of each face f as well as the volume Ω_c and centroid $\vec{x}^c$ of each cell c. Details about the used discretization method can be found in chapter 3.

The decision to store the above information, instead of computing it whenever needed, is based on a tradeoff between memory usage and computational efficiency. The software's ability to adjust its memory storage depending on each case is a middle-ground option, followed in this thesis. For example, if the memory resources are limited, the geometric quantities for uncut cells can be computed on the fly by eqs. 2.4 and 2.5, without any significant damage to the software's efficiency.

Subsequently, the computation of the aforementioned quantities for cut-cells, considered as arbitrary polyhedra, is presented. Firstly, the area and unit normal vector of each face are computed. The vector's direction is defined by the orbit of the face vertices $\vec{x}^i$, $i=1,\cdots,N$. The components of $\hat{\vec{n}}^f$ can be derived by the cross-product of the unit tangent vectors along two arbitrary edges of the face boundary. Although this method is efficient enough, the result is prone to roundoff errors, especially in very small faces. Thus, an alternative is preferred, which divides the face into N smaller triangles, each of them defined by an arbitrary point $\vec{c}$ on the plane of the face and two successive vertices (i.e., $\vec{x}^1$ and $\vec{x}^2$). Then, vectors $\vec{t}^1=\vec{x}^1-\vec{c}$ and $\vec{t}^2=\vec{x}^2-\vec{c}$, tangent to the face, are defined. The normal to the triangle's plane is

$$\vec{n}^t=\frac{1}{2}\left(\vec{t}^1\times\vec{t}^2\right)$$

Its magnitude is equal to the triangle's area A_t. Then, vector $\vec{n}^f$, defined as $\vec{n}^f=\hat{\vec{n}}^f A_f$, is computed as

$$\vec{n}^f=\sum_{t=1}^{N}\vec{n}^t$$

Finally, $A_f=|\vec{n}^f|$ and $\hat{\vec{n}}^f=\vec{n}^f/A_f$. The result is independent of $\vec{c}$, but numerical experiments show that the roundoff error is reduced by choosing the arithmetic mean $\bar{\vec{x}}^f$ of the face vertices.

The same geometric construction is used for the computation of $\vec{x}^f$, which is defined as

$$\vec{x}^f = \frac{1}{A_f}\int_{A_f}\vec{x}dA = \frac{1}{A_f}\sum_{t=1}^{N} A_t\vec{x}^t \tag{2.9}$$

where

$$\vec{x}^t = \frac{1}{A_t}\int_{A_t}\vec{x}dA = \frac{1}{3}\left(\vec{x}^1 + \vec{x}^2 + \bar{\vec{x}}^f\right)$$

is the centroid of each triangle, and A_t is its signed area given by

$$A_t = \vec{n}^t \cdot \hat{\vec{n}}^f$$

It is worth saying that $\vec{x}_f$ always lies on f because the developed mesh generator always provides planar faces.

Hereafter, the computation of the volume of each cut-cell is presented. Let F be the number of faces of each cut-cell. The widely used formula

$$\Omega_c = \frac{1}{3}\sum_{f=1}^{F}\left(\vec{x}^f \cdot \vec{n}^f\right)$$

is avoided because it is prone to roundoff errors and consequently fails to compute accurately the volume of very small cut-cells. On the contrary, pyramids are formed by connecting all vertices of the cut-cell with an arbitrary point $\vec{r}$. The base of each pyramid is the corresponding face f, and its apex is point $\vec{r}$. The summation of the signed volume Ω_f of all pyramids equals the total volume of the polyhedron. Volume Ω_f is

$$\Omega_f = \frac{1}{3}A_f h_f$$

where h_f is the height of each pyramid and is computed as

$$h_f = \left(\vec{x}^f - \vec{r}\right)\cdot\hat{\vec{n}}^f$$

Point $\vec{r}$ is set equal to the arithmetic mean $\bar{\vec{x}}^c$ of all cell vertices. This choice is of essential importance for the correct computation of the centroid in very small cut-cells. Furthermore, the cell centroid is computed as the volume-weighted average of the pyramid centroids,

$$\vec{x}^c = \frac{1}{\Omega_c}\int_{\Omega_c}\vec{x}d\Omega = \frac{1}{\Omega_c}\sum_{f=1}^{F}\Omega_f\vec{x}^{pf}$$

where $\vec{x}^{pf}$ is the centroid of the pyramid corresponding to face f. The latter is located on the line segment connecting $\vec{x}^f$ and $\vec{\bar{x}}^c$ being at a distance from $\vec{x}^f$ equal to one-quarter of its length, which implies

$$\vec{x}^c = \frac{3}{4\Omega_c} \sum_{f=1}^{F} \Omega_f \left(\vec{x}^f - \vec{\bar{x}}^c \right) + \vec{\bar{x}}^c$$

Finally, it is clarified that whenever a Cartesian cell is divided into multiple distinct cut-cells, the above geometric quantities are computed separately for each one of them.

2.6 Cell Merging

This section deals with a common issue in cut-cell methods called the "small cell problem". Generally, the intersection between the surface of a solid body and the Cartesian mesh creates cut-cells of arbitrary shapes and sizes. As a result, extremely small cell fragments most probably coexist next to regularly sized cells. However, small cut-cells adversely affect the flow solver's efficiency. Especially in explicit methods, the global time step's maximum value is drastically limited due to stability criteria, significantly delaying the simulation's wall-clock time [145]. On the other hand, in implicit methods, small cells increase the stiffness of the discretized equations' system, leading to stability and convergence issues [165].

The previously described problem is common to all cut-cell-based immersed boundary schemes and has been addressed using several approaches. Various researchers [340], [59], [338] have proposed a conservative approach according to which small cells can be eliminated by geometrically merging them with their neighbors. However, this process introduces additional complexities to the governing equations' discretization since the computational stencil becomes different for the merged cell and its neighbors. The problem is more profound when structured codes are used. Moreover, choosing the appropriate neighbors for merging is non-trivial in 3D meshes [218].

A way to overcome these issues is by applying the cell linking approach [165], where a small cell is linked with an adjacent bigger master cell to form a master-slave pair. This method retains both the master and the slave cell, avoiding their geometrical unification. Thus, the governing equations are discretized separately for both cells, applying the same procedure followed for the rest cells. Another method, which also avoids the topological changes on the mesh, is the cell mixing procedure [132]. According to that, fluid portion of a target cell is transferred to small cells maintaining the conservation of the flow equations. Finally, some other mixed procedures can be found in [124], [54].

In this thesis, a cell merging approach is presented based on [145] that circumvents the problems mentioned above. According to this approach, each small cell is geometrically merged with a bigger neighbor creating a hyper-cell, which substitutes both the small and bigger cells in the mesh data structure. Therefore, a new centroid and volume are computed and the list of neighbors is updated for the new cell and all its neighbors. Finally, nodes, faces, and cells of the mesh are re-numbered. Since these changes are based on the already developed data structure, the new modifications are implemented with a negligible computational cost. The developed method is exemplified in fig. 2.16.

The great advantage of this process is that the governing equations' spatial discretization remains unchanged throughout all mesh cells. Consequently, the flow solver does not treat the new merged cells differently, simplifying the software's structure. More specifically, the flow variables are stored at the merged cut-cell centroid, avoiding small cells' participation in the discretization scheme and, thus, preventing convergence issues they cause. Finally, although the proposed approach is more complicated, it is more natural in the sense that it preserves the conservative nature of the flow equations and avoids any artificial interference to the discretization scheme.

The algorithm responsible for the cell merging is then presented. The first step implements the definition of a "small cell", adopted in this thesis, to tag the proper cells for merging. A cut-cell is considered small when attached to a cell with a volume at least 20 times bigger. Then, a neighboring cell is chosen to be merged with each tagged cell. The criteria used to determine the proper neighbor aim to avoid creating stretched merged cells that may harm the flow simulation. Moreover, the algorithm is flexible enough to allow more than two cells to form a hyper-cell, which is usually needed in regions where the curvature of the wall is considerably high. Fig. 2.17 illustrates such a case.

The next step categorizes the tagged small cells into different zones. This is necessary because different merging criteria are implemented into different zones. Also, it has been observed that the treatment of small cells by zone results in a better quality mesh. In the first zone belong all small cells, which have at least one regular-sized (non-small) neighbor. The second zone contains the small cells which are attached to cells of the first zone. The next zones are formed following the same rule. The merging process starts from the cells of the first zone. Then, each of them is merged by discarding the largest face that connects it with a regular-sized neighbor. Next, the cells of the second zone are merged with the neighboring regular-sized or already merged cell of the smallest total volume. The process continues with the rest zones by using the volume criterion until all small cells disappear.

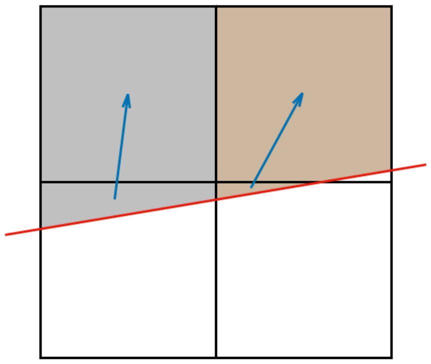


Figure 2.16: Two small cells are merged with their regular-sized neighbors, creating two new colored cells. A blue arrow starts from each small cell showing the regular-sized cell which is chosen for merging.

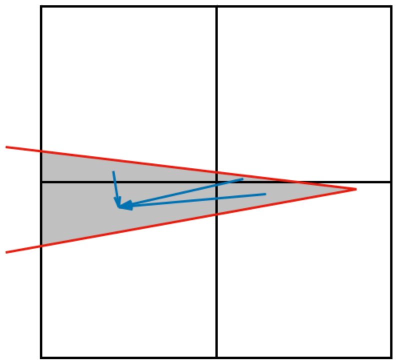


Figure 2.17: Three small cells are merged with a bigger one, creating a new cell shadowed with gray. A blue arrow starts from each small cell showing the chosen neighbor for merging.

2.7 Mesh Partitioning

The flow simulation of real-life applications requires the development of a flow solver adjusted in a parallel programming environment. This work implements a parallel non-shared memory programming method based on the Message Passing Interface (MPI). The multi-block decomposition of the domain is a prerequisite for the implementation of this strategy. The mesh-partitioning can be obtained in different ways. One option is to apply a commercial software package like METIS [158] or SCOTCH [58]. However, an attractive alternative arises from exploiting the nature of Cartesian meshes. This thesis develops a partitioner based on the Space-Filling Curve (SFC) concept [22], widely employed among various Cartesian methods [51].

The family of space-filling curves has been developed as a mapping method rather than a partitioning technique. The curves are characterized by their ability to pass through every point of a multi-dimensional space, hence the name “space-filling”. Peano first defined the concept of these curves in 1890 [241]. Since then, numerous types have been created (i.e., Hilvert [128] or Morton [217] curves) and applied in different fields, including mathematics [50], computational physics [271], geographical information systems [3], image processing [293], databases [45], algebraic multigrid [115], and mesh generation [30]. The developed partitioner presented in this subsection considers the Hilbert SFC approach due to the unique properties it offers. Fig. 2.18b shows the curve visiting each cell of the 4×4 block. A comparative study of various partitioners using different SFCs can be found in [7].

There are three essential properties associated with the Hilbert SFC making it attractive as a mesh partitioner [246],[4], namely,

1. Mapping: it provides a unique mapping from the 2D or 3D space to a 1D space.
2. Locality: two cells adjacent on the curve remain neighbors in the 2D or 3D mesh.
3. Compactness: it requires only local information (cell’s integer coordinates) for the mapping construction, disregarding the connectivity of the mesh.

Once the curve has passed through each Cartesian cell, the first two properties allow for a fast and straightforward mesh decomposition. The only operation needed is to divide the curve into equal segments and map the corresponding mesh domains to each processor. Subsequently, an example of a mesh of 10K cells is given, which should be decomposed into four processors. Firstly, the Hilbert SFC renumbers all cells giving each of them a unique ID. Then, cells with ID smaller or equal to 2500 belong to the first processor, cells with ID between 2501 and 5000 belong to the second one, and so on.

One advantage of this approach is that once the mapping from the SFC is computed and stored, the mesh can be repartitioned into any number of processors without paying almost any extra computational cost. This practice is very beneficial when a simulation should be restarted on a different number of processors. Alternatively, extra flexibility is given to users of a time-sharing workstation cluster environment, where the number of available processors may not be known a priori.

Moreover, locality ensures that the divided mesh domains are simply connected spaces. In other words, each domain consists of one piece without holes passing through it. This property reduces the computational cost caused by interprocessor communications. In fact, the quality of the mesh decomposition resulted from the SFC curves is quite competitive compared to other popular partitioners [246]. Furthermore, close distant cells are most likely to preserve their locality in the curve's access pattern. Thus, the discretization of the flow equations based on the presented renumbering results in a linear system with more nonzero elements close to the matrix diagonal. Overall, the reordering improves the conditioning of the matrix facilitating the system's solution process and reducing the simulation's wall clock time [274]. Finally, compactness is also important because it supports the parallel construction of the curve, avoiding the frequent communication between processors.

The Hilbert SFC is defined recursively in self-similar levels. Each subsequent level adds segments to the previous level's curve filling more of the space. Due to the special nature of this curve, the partitioning is applied to the Cartesian mesh before the removal of its solid part. The curve is constructed in parallel with the mesh generation procedure based on the tree data structure explained in section 2.2. The curve passes through all cells of the mesh, constructed at each level, using information from the curve's shape of the previous level. In the first level, the mesh consists of only one cell, and thus the curve degenerates to a point. Fig. 2.18a shows the Hilbert curve in the mesh of the second level forming the characteristic 'U' shape. This curve is called primitive and represents the simplest Hilbert curve [147]. It starts from the bottom left corner of the square and ends at the bottom right corner. Different orientations of the 'U' shape lead to four in total undirected primitive curves in 2D. Each of them can be traversed in two directions, defining eight primitive curves.

Fig. 2.18b presents the curve at the next level, which has been enriched exclusively by primitive curves. At each consecutive level, the curve is formed by appropriately connecting multiple primitive curves, figs. 2.18c and 2.18d. However, in an adapted Cartesian mesh, the curve is not entirely shaped. Fig. 2.18e demonstrates a non-uniform mesh around an airfoil and the corresponding Hilbert SFC. The subdivision of the curve into four equivalent parts leads to the mesh decomposition into four regions colored differently in fig. 2.18f.

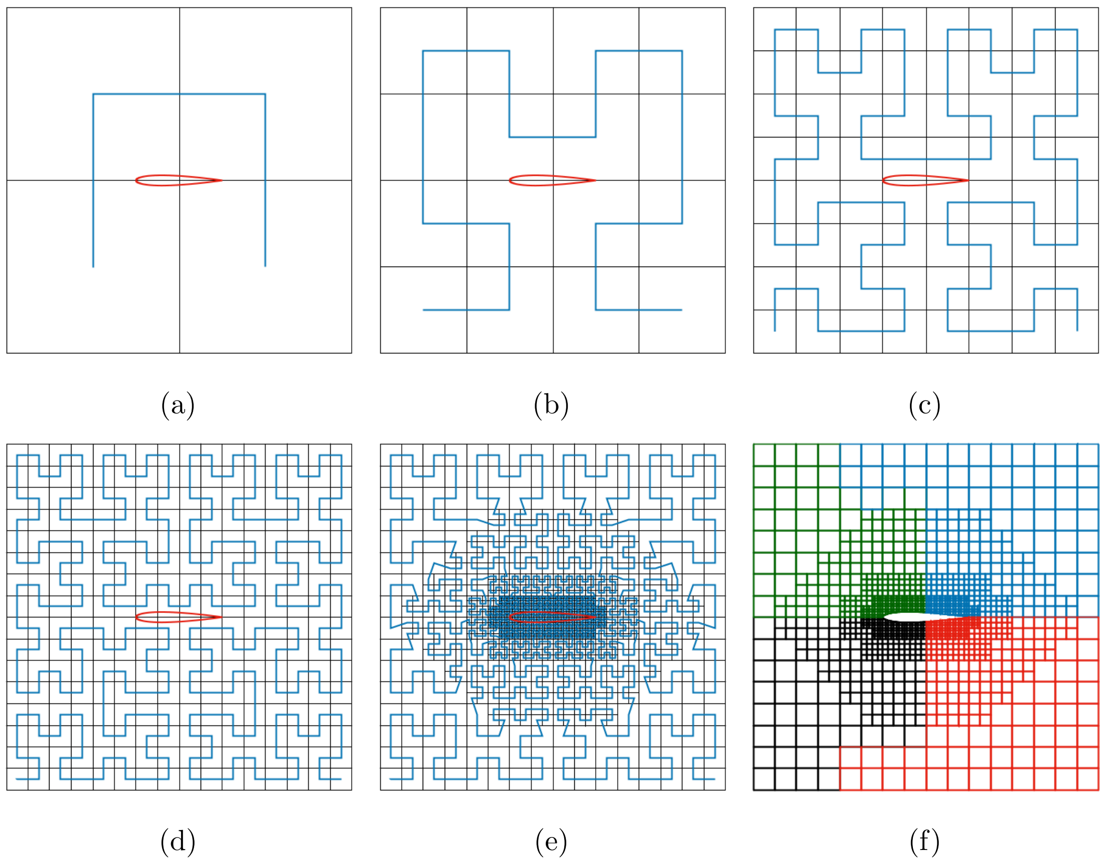


Figure 2.18: (a-e) Different levels during the mesh generation process around an airfoil. At each level, the Hilbert SFC (blue line) visits every Cartesian cell. Then, it is used to decompose the final mesh into four parts shown in (f) with different colors.

The extension of the Hilbert SFC in 3D is possible by defining new primitive curves. The 3D curve follows the same basic construction rules, but its shape is more complicated, allowing the curve to turn into multiple directions to pass through each cell of a 3D mesh. An example of a curve in a $4 \times 4 \times 4$ block is shown in fig. 2.19. The definition of the primitive curves is more sophisticated than in the 2D case, leading to 48 directed primitive curves embedded inside a $2 \times 2 \times 2$ block. Moreover, the 3D primitive curves are not uniquely defined, allowing different Hilbert curves to occupy a given mesh. A complete discussion about the different curves of the Hilbert family is presented in [126]. A comparison of partitioners in terms of efficiency is demonstrated as well.

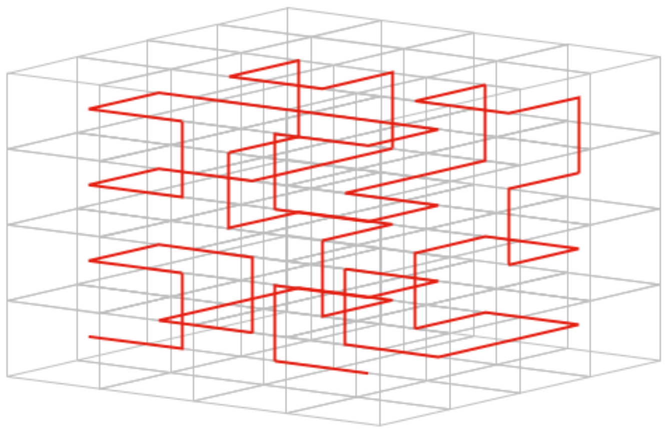


Figure 2.19: An example of a 3D Hilbert SFC passing through all cells of a $4 \times 4 \times 4$ mesh.

Subsequently, the shape of the 2D primitive curves is defined using the children's numbering shown in fig. 2.4. Each primitive curve visits the four cells of a 2×2 block in a different order. For example, the 'U' curve, shown in fig. 2.18a, follows the sequence 0, 2, 3, 1, where each number corresponds to a child cell of the block. The following matrix, called "PrimitiveCurves", contains the characteristic sequence of each primitive curve in each of its 8 rows

$$PrimitiveCurves = \begin{bmatrix} 0 & 3 & 2 & 1 & 1 & 2 & 0 & 3 \\ 2 & 1 & 3 & 0 & 3 & 0 & 1 & 2 \\ 3 & 0 & 1 & 2 & 2 & 1 & 3 & 0 \\ 1 & 2 & 0 & 3 & 0 & 3 & 2 & 1 \end{bmatrix}^T$$

The row number names each curve starting from 0, and thus the '0' curve is the one shown in fig. 2.18a.

The curve of the next level emerges by substituting each of the 4 cells with a new 2×2 block traversed by a primitive curve [184]. Therefore, a recipe is needed to determine which primitive curve replaces each cell. For example, in curve 0, primitive curves 6, 0, 0, and 7 replace children 0, 1, 2, and 3, leading to the curve of fig. 2.18b. Equivalently, the rest primitive curves from 1 to 7 correspond to a different sequence of 4 numbers. Matrix "CurveReplacement" contains such a sequence in each of its 8 rows.

$$CurveReplacement = \begin{bmatrix} 6 & 7 & 5 & 4 & 3 & 2 & 0 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 7 & 6 & 4 & 5 & 2 & 3 & 1 & 0 \end{bmatrix}^T$$

The corresponding matrices "PrimitiveCurves" (PC) and "CurveReplacement" (CR) for the 3D Hilbert SFC are

$$PC = \begin{bmatrix}
0 & 5 & 1 & 4 & 6 & 3 & 2 & 7 & 4 & 2 & 0 & 6 & 1 & 7 & 3 & 5 & 0 & 3 & 2 & 1 & 5 & 6 & 4 & 7 & \cdots \\
2 & 7 & 3 & 6 & 4 & 1 & 0 & 5 & 5 & 3 & 1 & 7 & 0 & 6 & 2 & 4 & 4 & 7 & 6 & 5 & 1 & 2 & 0 & 3 & \\
4 & 1 & 0 & 5 & 2 & 7 & 3 & 6 & 0 & 6 & 2 & 4 & 5 & 3 & 1 & 7 & 1 & 2 & 0 & 3 & 4 & 7 & 6 & 5 & \cdots \\
6 & 3 & 2 & 7 & 0 & 5 & 1 & 4 & 1 & 7 & 3 & 5 & 4 & 2 & 0 & 6 & 5 & 6 & 4 & 7 & 0 & 3 & 2 & 1 & \\
5 & 0 & 4 & 1 & 3 & 6 & 7 & 2 & 2 & 4 & 6 & 0 & 7 & 1 & 5 & 3 & 3 & 0 & 1 & 2 & 6 & 5 & 7 & 4 & \cdots \\
7 & 2 & 6 & 3 & 1 & 4 & 5 & 0 & 3 & 5 & 7 & 1 & 6 & 0 & 4 & 2 & 7 & 4 & 5 & 6 & 2 & 1 & 3 & 0 & \\
1 & 4 & 5 & 0 & 7 & 2 & 6 & 3 & 6 & 0 & 4 & 2 & 3 & 5 & 7 & 1 & 2 & 1 & 3 & 0 & 7 & 4 & 5 & 6 & \cdots \\
3 & 6 & 7 & 2 & 5 & 0 & 4 & 1 & 7 & 1 & 5 & 3 & 2 & 4 & 6 & 0 & 6 & 5 & 7 & 4 & 3 & 0 & 1 & 2 & \\
3 & 6 & 7 & 2 & 5 & 0 & 4 & 1 & 7 & 1 & 5 & 3 & 2 & 4 & 6 & 0 & 6 & 5 & 7 & 4 & 3 & 0 & 1 & 2 & \cdots \\
1 & 4 & 5 & 0 & 7 & 2 & 6 & 3 & 6 & 0 & 4 & 2 & 3 & 5 & 7 & 1 & 2 & 1 & 3 & 0 & 7 & 4 & 5 & 6 & \\
7 & 2 & 6 & 3 & 1 & 4 & 5 & 0 & 3 & 5 & 7 & 1 & 6 & 0 & 4 & 2 & 7 & 4 & 5 & 6 & 2 & 1 & 3 & 0 & \cdots \\
5 & 0 & 4 & 1 & 3 & 6 & 7 & 2 & 2 & 4 & 6 & 0 & 7 & 1 & 5 & 3 & 3 & 0 & 1 & 2 & 6 & 5 & 7 & 4 & \\
6 & 3 & 2 & 7 & 0 & 5 & 1 & 4 & 1 & 7 & 3 & 5 & 4 & 2 & 0 & 6 & 5 & 6 & 4 & 7 & 0 & 3 & 2 & 1 & \cdots \\
4 & 1 & 0 & 5 & 2 & 7 & 3 & 6 & 0 & 6 & 2 & 4 & 5 & 3 & 1 & 7 & 1 & 2 & 0 & 3 & 4 & 7 & 6 & 5 & \\
2 & 7 & 3 & 6 & 4 & 1 & 0 & 5 & 5 & 3 & 1 & 7 & 0 & 6 & 2 & 4 & 4 & 7 & 6 & 5 & 1 & 2 & 0 & 3 & \cdots \\
0 & 5 & 1 & 4 & 6 & 3 & 2 & 7 & 4 & 2 & 0 & 6 & 1 & 7 & 3 & 5 & 0 & 3 & 2 & 1 & 5 & 6 & 4 & 7 &
\end{bmatrix}^T$$

$$CR = \begin{bmatrix}
46 & 43 & 34 & 39 & 42 & 47 & 38 & 35 & 40 & 45 & 30 & 27 & 44 & 41 & 26 & 31 & 36 & 33 & 24 & 29 & 32 & 37 & 28 & 25 & \cdots \\
18 & 23 & 14 & 11 & 22 & 19 & 10 & 15 & 20 & 17 & 2 & 7 & 16 & 21 & 6 & 3 & 8 & 13 & 4 & 1 & 12 & 9 & 0 & 5 & \\
36 & 32 & 44 & 40 & 37 & 33 & 45 & 41 & 28 & 24 & 46 & 42 & 29 & 25 & 47 & 43 & 30 & 26 & 38 & 34 & 31 & 27 & 39 & 35 & \cdots \\
9 & 13 & 17 & 21 & 8 & 12 & 16 & 20 & 1 & 5 & 19 & 23 & 0 & 4 & 18 & 22 & 3 & 7 & 11 & 15 & 2 & 6 & 10 & 14 & \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & \cdots \\
24 & 25 & 26 & 27 & 28 & 29 & 30 & 31 & 32 & 33 & 34 & 35 & 36 & 37 & 38 & 39 & 40 & 41 & 42 & 43 & 44 & 45 & 46 & 47 & \\
11 & 14 & 18 & 23 & 10 & 15 & 19 & 22 & 2 & 7 & 17 & 20 & 3 & 6 & 16 & 21 & 1 & 4 & 8 & 13 & 0 & 5 & 9 & 12 & \cdots \\
39 & 34 & 46 & 43 & 38 & 35 & 47 & 42 & 30 & 27 & 45 & 40 & 31 & 26 & 44 & 41 & 29 & 24 & 36 & 33 & 28 & 25 & 37 & 32 & \\
15 & 10 & 22 & 19 & 14 & 11 & 23 & 18 & 6 & 3 & 21 & 16 & 7 & 2 & 20 & 17 & 5 & 0 & 12 & 9 & 4 & 1 & 13 & 8 & \cdots \\
35 & 38 & 42 & 47 & 34 & 39 & 43 & 46 & 26 & 31 & 41 & 44 & 27 & 30 & 40 & 45 & 25 & 28 & 32 & 37 & 24 & 29 & 33 & 36 & \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & \cdots \\
24 & 25 & 26 & 27 & 28 & 29 & 30 & 31 & 32 & 33 & 34 & 35 & 36 & 37 & 38 & 39 & 40 & 41 & 42 & 43 & 44 & 45 & 46 & 47 & \\
33 & 37 & 41 & 45 & 32 & 36 & 40 & 44 & 25 & 29 & 43 & 47 & 24 & 28 & 42 & 46 & 27 & 31 & 35 & 39 & 26 & 30 & 34 & 38 & \cdots \\
12 & 8 & 20 & 16 & 13 & 9 & 21 & 17 & 4 & 0 & 22 & 18 & 5 & 1 & 23 & 19 & 6 & 2 & 14 & 10 & 7 & 3 & 15 & 11 & \\
42 & 47 & 38 & 35 & 46 & 43 & 34 & 39 & 44 & 41 & 26 & 31 & 40 & 45 & 30 & 27 & 32 & 37 & 28 & 25 & 36 & 33 & 24 & 29 & \cdots \\
22 & 19 & 10 & 15 & 18 & 23 & 14 & 11 & 16 & 21 & 6 & 3 & 20 & 17 & 2 & 7 & 12 & 9 & 0 & 5 & 8 & 13 & 4 & 1 &
\end{bmatrix}^T$$

where the dots indicate the continuation of the row to the next line. Algorithm 9 maps each cell c to its new identification number represented by index c_h such that $c{=}hilbertCurve[c_h]$, where $hilbertCurve$ stores the new numbering of cells. A similar approach is discussed in [147].

Algorithm 9: Hilbert Curve Generator (1)

```
input  : mesh topology
output: $hilbertCurve$

Main Function createHilbertCurve()
    // PCI: primitive curve index
    $c \leftarrow 0$
    $PCI \leftarrow 0$ // it can be any number between 0 and 7
    $c_h \leftarrow 0$
    createNextLevelOfHilbertCurve($c$, $PCI$, $c_h$)
return
```

Function "createNextLevelOfHilbertCurve" is presented in Algorithm 10.

Algorithm 10: Hilbert Curve Generator (2)

```
Function createNextLevelOfHilbertCurve($c$, $PCI$, $c_h$)
    if not $cellIsRefined[c]$ then
        $hilbertCurve[c_h] \leftarrow c$
        $c_h \leftarrow c_h + 1$
        return
    end

    $\vec{i} \leftarrow cells[c]$
    foreach $child\ k$ do
        // $k \in [0,7]$ or $k \in [0,3]$ for 2D or 3D, respectively
        $j \leftarrow PrimitiveCurves[PCI][k]$
        $PCI_c \leftarrow CurveReplacement[PCI][k]$
        foreach $dimension\ d \in [0,2]\ or\ d \in [0,1]$ do
            $m \leftarrow childIdentification[j][d]$
            $i_c[d] \leftarrow 2i[d] + m$
        end
        $c_c \leftarrow cellPosition[\vec{i_c}]$
        createNextLevelOfHilbertCurve($c_c$, $PCI_c$, $c_h$) // recursion
    end
return
```

Matrices $cells$, $cellPosition$, and $cellIsRefined$ used above are defined in subsection 2.2.4. Finally, matrix $childIdentification$ is given by eq. 2.8.

2.8 Mesh with Moving Boundaries

This section deals with moving impermeable boundaries within a fixed Cartesian mesh. The prescribed motion of a solid body, immersed in a fixed Cartesian mesh, causes extra complexities, absent from the mesh generation process around static geometries demonstrated in the previous sections. The presented method facilitates the flow solver maintaining the conservation of the flow even for large boundary displacements and retaining its efficiency and robustness.

At each time step, the geometry changes its position, and the mesh is re-adapted to the updated solid wall keeping track of its motion. Thus, regions close to the geometry's previous position are coarsened, and cells in the vicinity of the displaced boundary are split anew, increasing the flow simulation's accuracy. A method for the field's extrapolation at each time step to the subsequent mesh is presented in subsection 2.8.1. Although the status of most cells remains unchanged over a time step, the cells within the region swept by the moving boundary modify their shape and nature. In particular, the displaced fluid-solid interface may cover fluid cells changing their nature from fluid to solid. These solidified cells, called covered cells, disappear from the domain and are no longer used during the flow simulation. In contrast, the geometry's motion can reveal solid cells, called uncovered cells, which suddenly appear as newborn cells in the fluid part of the mesh.

These transitions cause abrupt modifications in the governing equations' discretization, which act like spurious sources or sinks, generating artificial oscillations traveling throughout the flow field and deteriorating the flow solution. The cause of these abnormalities has been studied extensively by numerous researchers. For example, according to [279], unphysical pressure oscillations are induced by violating the geometric conservation law due to the cells' sudden appearance or disappearance. Moreover, [189] mentions that the abrupt change in the numerical scheme's stencil may also cause such oscillations. Finally, the influence of the time step's size and the mesh width on the oscillations' generation has been investigated by [137]. Therefore, additional modifications are required to the numerical scheme to eliminate the truncation error and maintain strict conservation. Existing approaches involve merging newborn cells with their neighbors [340], use ghost-cells to provide continuation in time to the conservative variables [339], or implement Lagrangian interpolation to estimate the velocities of the uncovered cells [344]. This thesis mimics the merging technique and improves it by developing a cell linking method, described in subsections 2.8.2 and 2.8.3, ensuring flow conservation without restricting the maximum displacement of the immersed geometry at each time step.

2.8.1 Mapping Between Subsequent Meshes

The discussion starts with the study of the continuous mesh adaptation to the moving boundary. Fig. 2.20 shows an airfoil performing an upward translational motion followed by the corresponding adapted mesh. The mesh generation at each time step can be initialized by the mesh of the previous time instant and then readjust the final mesh performing the appropriate coarsening and refining operations. However, generating a mesh from scratch is preferred because it simplifies the corresponding algorithm without damaging its efficiency. Indeed, both strategies were developed and compared without noticing any significant difference in their efficiency.

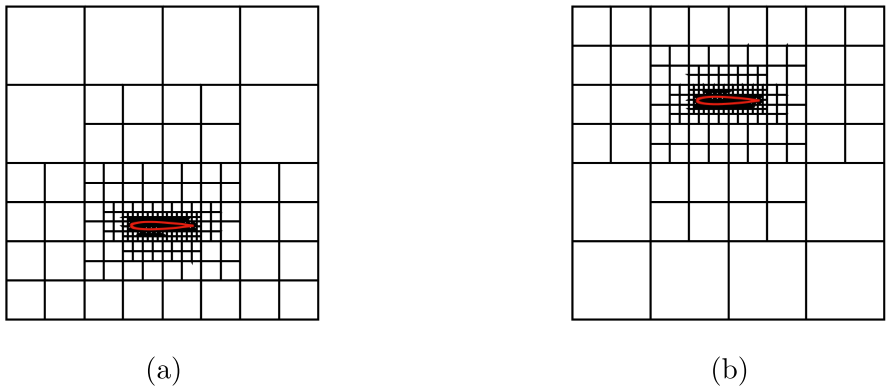


Figure 2.20: An airfoil performs a translational motion. The mesh is dynamically adapted to the solid boundary (red) at each time step.

The mapping between two successive meshes supports the flow solver with the necessary time history for each cell. In general, two successive meshes do not have a cell-to-cell match, and 3D interpolation of the flow field is required. However, finding interpolants by searching over all cells of both meshes is very expensive and impractical. Nevertheless, the tree data structure, presented in section 2.2, allows for an efficient way to detect the cells of the new mesh (e.g., B) contained in each cell of the old mesh (e.g., A) and the other way around. Fig. 2.21 shows a cell of A mapped onto various cells of B belonging to different refinement levels. The left-right arrow represents the opposite case, where a group of cells corresponds to a larger one. The developed algorithm starts from the root cell in both meshes and visits all tree levels cell-by-cell until a leaf cell appears, e.g., in A. If its counterpart in tree B is further subdivided, the algorithm traverses the rest of tree B, searching for its offspring leaf cells, which are included by definition in the cell of A. Fig. 2.22 proves that both cases represented by the left-right arrow in fig. 2.21 can be present concurrently. Blue arrows show the mapping between the black and the red mesh. Finally, section 3.3 presents the mathematical formulation for transferring the flow variables to the new mesh.

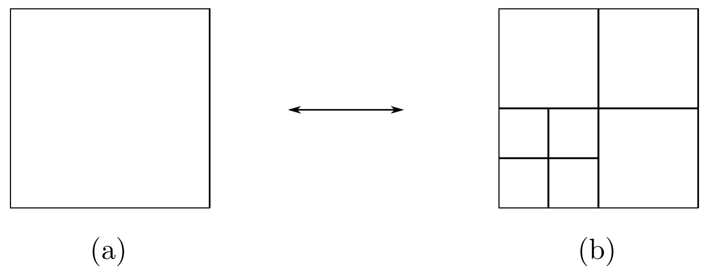


Figure 2.21: Two meshes corresponding to successive time steps are shown. The cell on the left is mapped onto several smaller cells on the right. Inversely, various cells on the right are mapped onto only one cell on the left.

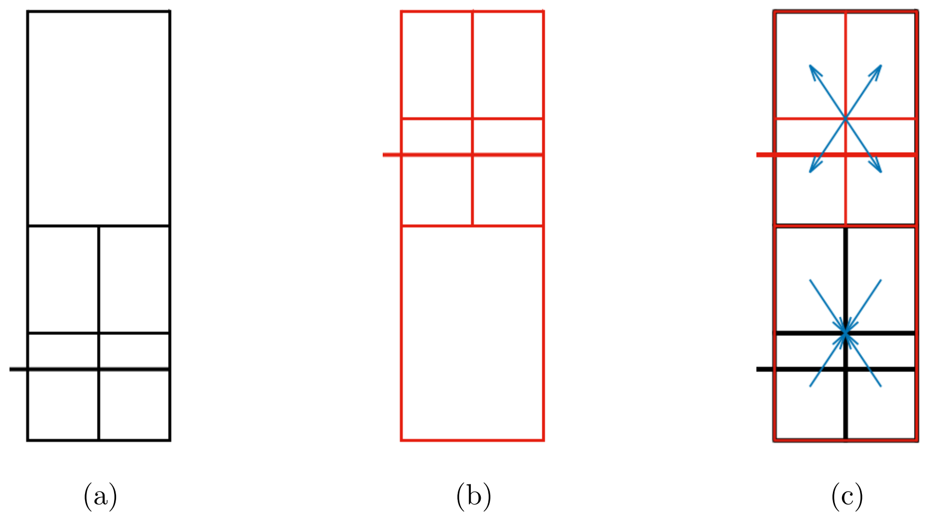


Figure 2.22: (a, b) Two meshes at successive time steps are presented in different colors (black, red). The bold horizontal line indicates the fluid-solid interface. (c) Blue arrows depict the mapping between the two meshes plotted on top of each other.

2.8.2 Covered and Uncovered Cells

Firstly, the treatment of covered cells by the solid geometry is discussed. Fig. 2.23c shows a cluster of two cells, the lower of which becomes solid due to the boundary's upward motion. The loss of conservative variables stored in this cell is avoided by merging it with its neighbor before the surface displacement. Figs. 2.23b and 2.23a represent the past and present states of the merged fluid cell. Consequently, although the cut-cell finally disappears from the fluid domain, its contribution to the conserved variables is transferred into its neighbor.

However, the cell merging technique may lead to very complicated structures, especially in large surface displacements. Thus, a virtual linking between the brown cell of fig. 2.23c and the gray cell of fig. 2.23a is preferred, avoiding the formulation of the merged cell in fig. 2.23b. Consequently, the linking method corrects the flow variables, which are stored in the gray cell defining its time history. A similar strategy is shown in [277], where the numerical error is redistributed from the disappeared cut-cell to the surrounding cells.

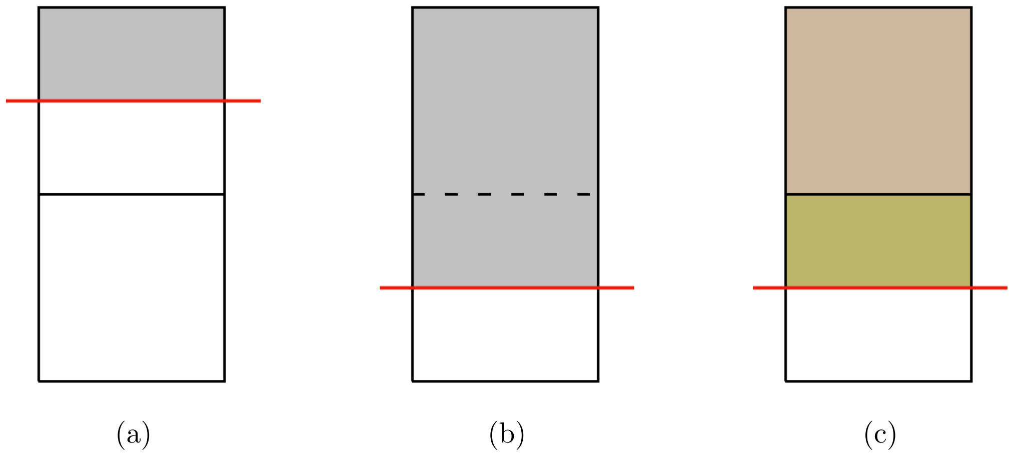


Figure 2.23: From left to right, the boundary moves downwards and reveals a newborn cell. In the reversed order, a fluid cell is covered by the solid body. Colors correspond to the mesh' fluid part and indicate the proper cell merging.

On the other hand, following the subfigures from subfig. 2.23a to subfig. 2.23c, the fluid-solid interface moves downwards, uncovering the lower cell. Such cells emerge into the fluid with no flow solution history, and thus their time integration is meaningless. This issue is resolved by temporarily linking the newborn cells with an adequately chosen neighbor in a similar manner to the solidification approach. Such corrections ensure that all newborn cells are initialized with consistent flow values at the cost of a more complex algorithm and implementation. The following subsection presents an algorithm that, under specific criteria, determines the linking between cells.

2.8.3 Cell Linking

The correct linking between cells is far from trivial due to various geometric conditions needed to be satisfied. Firstly, each cluster of connected cells should define a simply connected space. Secondly, neighboring clusters of linked cells that differ much in volume shall be avoided because they damage the robustness of the flow solver. Finally, it is recommended to formulate linked cells in the direction of the surface velocity vector. This practice is consistent with the concept of the grid velocity used in body-fitted meshes.

Algorithm 11 determines the correct cell linking, accomplishing all the aforementioned requirements. Its input is the data structure of two successive meshes, A and B. If A stands for the old mesh and B for the new one, the developed method handles covered cells linking each of them with a fluid neighbor, both located in B. On the other hand, the algorithm deals with uncovered cells if A and B correspond to the new and old mesh, respectively. For the sake of brevity, the subsequent analysis refers only to the cells' solidifications.

The algorithm's first step is implemented by function "disappearedCell" shown in Algorithm 12, which detects disappeared cells from the domain of B. Then an iteration over these cells is carried out, looking over the proper linking among their first neighbors. Only fluid neighbors c_n in a requested direction are allowed to be linked with each disappeared cell c. At this point, angle ϕ is introduced to determine this direction. It is defined as the angle between the vector connecting the centroids of c and c_n and a vector defined as the arithmetic mean of velocity vectors stored in the solid faces of c. Then, the proper neighbor for linking is the one that forms the smallest angle ϕ, which should also be less than $90°$ degrees (condition in line 15 of Algorithm 12). Function "connectFirstCategoryCells" in Algorithm 12 is responsible for these computations.

The necessary velocity on each face centroid ($\vec{x}^f$) is computed anew at each time instant since the solid faces may not be continuously present during the unsteady flow simulation. Therefore, its value emerges by interpolating the velocities of the solid surface's vertices. Considering that $\vec{x}^f$ lies on an arbitrary triangle of the solid surface defined by vertices $\vec{x}^1$, $\vec{x}^2$, and $\vec{x}^3$ provided with velocities $\vec{v}^1$, $\vec{v}^2$, and $\vec{v}^3$, respectively, the extrapolated velocity $\vec{v}^f$ is computed as

$$\vec{v}^f = \frac{1}{A_t}\left(A_{23}\vec{v}^1 + A_{31}\vec{v}^2 + A_{12}\vec{v}^3\right)$$

where A_t is the triangle's area and A_{ij} is the area of a triangle defined by $\vec{x}^f$ and vertices $\vec{x}^i$ and $\vec{x}^j$.

However, not all disappeared cells are able to find a linking satisfying the previously mentioned criterion. The rest are linked by function "connectSecondCategoryCells", also presented in Algorithm 12. According to that, each cell c looks for the proper link in the group of fluid cells that have already been linked with the neighbors of c. The fluid cell forming the smallest angle ϕ is preferred.

The final step prevents the creation of large clusters of linked cells. Thus, each covered cell is allowed to be linked with multiple fluid cells distributing its flow variables to a broader mesh area. This new kind of linking is achieved under the following rules. The first one refers to each covered cell c, already connected to a fluid cell c_n of higher refinement level, which also happens to be its neighbor. This cell should also be connected with the common neighbor c'_n of c and c_n under the condition that c'_n is a fluid cell. Fig. 2.24a exemplifies the case by plotting a red mesh, on which the horizontal fluid-solid interface moves upwards. As a result, the covered large cell at the bottom is linked to two neighboring cells of smaller size.

A second rule also allows for multiple linking and refers only to each solid cell c surrounded exclusively by solid neighbors. Let c_n be a smaller in size neighbor, which is connected to a fluid cell c_f. Then, c can also be connected with c_f. An example is shown in fig. 2.24b. Each of the 4 small disappeared cells is connected with its fluid neighbor of the same size. Then, each of the two larger disappeared cells is connected with two of these fluid neighbors. Finally, both rules mentioned before are imposed by function "distributeConnection" used in Algorithm 11. Fig. 2.24c demonstrates the case shown in fig. 2.22c after the implementation of this function.

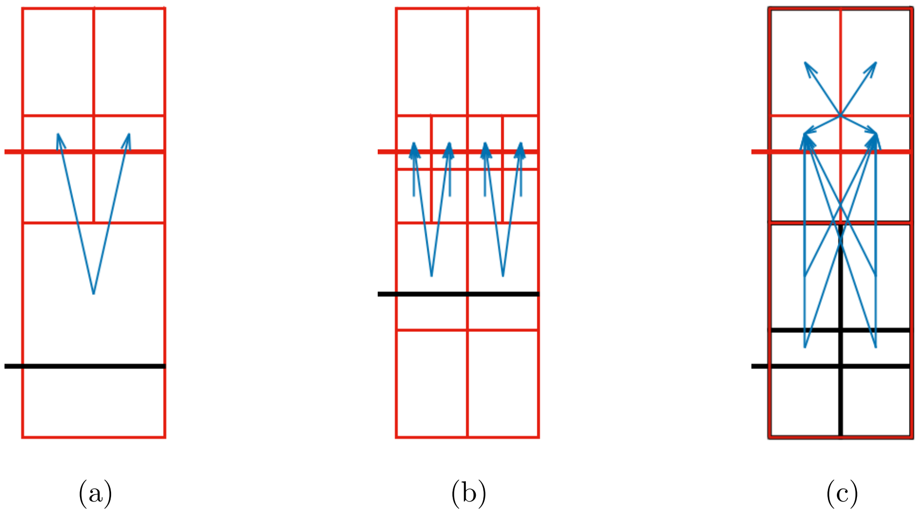


Figure 2.24: (a) The solid surface, depicted by a bold horizontal line, moves from its black to the red position over an unmodified, in time, mesh. (a) Two blue vectors show the contribution of the covered large cell to two smaller cells. (b) Each one of the 4 small covered cells is linked with its fluid neighbor. The larger covered cells use this information to be linked with the same fluid cells. (c) The final cell linking applied in the example of fig. 2.22c after imposing function "distributeConnection". The 12 vectors indicate the linking between the two subsequent in time meshes.

Algorithm 11: Cell Linking in Unsteady Cases (1)

```
  input  : mesh $A$, mesh $B$
  output: list $connection$

1 Main Function linkCells()
      // create list $DC$ of disappeared cells
2     foreach cell $c_B$ of mesh $B$ do
3         if disappearedCell($c_B$) then $DC \leftarrow$addToList($c_B$)
4     end
      // fill list $connection$
5     connectFirstCategoryCells()
6     connectSecondCategoryCells()
7     distributeConnection()
8 return
```

Algorithm 12: Cell Linking in Unsteady Cases (2)

```
Function disappearedCell($c_B$)
    if $c_B$ is fluid cell then return false
    foreach cell $c_A$ of mesh $A$ mapped onto $c_B$ do
        if $c_A$ is fluid cell then return true
    end
return false

Function connectFirstCategoryCells()
    foreach cell $c$ of list $DC$ do
        foreach neighbor $c_n$ of $c$ do
            if cellIsFluid($c_n$) and not disappearedCell($c_n$) then
                $neighbors \leftarrow$ addToList($c_n$)
            end
        end
        $c_n \leftarrow$ findProperCell($c$, $neighbors$, $correctDirection$)
        if $correctDirection$ then $connection[c] \leftarrow c_n$
    end
return

Function connectSecondCategoryCells()
    $correction \leftarrow true$
    while $correction$ do
        $correction \leftarrow false$
        foreach second category cell $c$ do
            foreach connected neighbor $c_n$ of $c$ do
                $neighbors \leftarrow$ addToList($connection[c_n]$)
            end
            $c'_n \leftarrow$ findProperCell($c$, $neighbors$, $correctDirection$)
            if $connection[c] \neq c'_n$ then
                $connection[c] \leftarrow c'_n$
                $correction \leftarrow true$
            end
        end
    end
return
```

2.9 Mesh Differentiation

A significant part of this thesis deals with gradient-based shape optimization methods applied in complex geometric structures of industrial interest. A continuous or discrete adjoint approach supports the optimization algorithm computing the required shape sensitivities on a Cartesian mesh. The corresponding mathematical development is presented in section 7.4 and indicates the need for the mesh's differentiation. The term "mesh differentiation" refers to the computation of the variation of every geometric quantity included in the flow equations' discrete form caused by the infinitesimal change of the geometry's shape. The quantities of interest are the unit normal vector $\hat{\vec{n}}^f$, surface area A_f, and centroid $\vec{x}^f$ of each face f, as well as the volume Ω_c and centroid $\vec{x}^c$ of each cell c comprising the members of the generalized vector $\vec{G}$.

Moreover, using a parameterization tool to update the geometry's shape during the optimization loop is out of this thesis discussion. Thus, the geometry's modification is expressed by changing the coordinates of vertices $\vec{v}^k$ constituting the triangulated geometry's surface, where k indicates the serial number of each node. Furthermore, the set of vertices $\vec{v}^k$ is the only input of the mesh generator that varies during the optimization process, implying that $\vec{G}$ is a function of $\vec{v}^k$. The computation of its derivative $\partial\vec{G}/\partial\vec{v}^k$ is the target of this section.

While this approach is well understood and widely used in implementations based on body-fitted meshes, an alternative approach is required for the cut-cell method. The difference originates from the special readjustment of the Cartesian mesh to the modified geometry at each optimization step. In particular, if the geometry's deformation is infinitesimally small, only the shape of the cut-cells is affected, preserving the mesh's topological characteristics. In other words, the face-vertices and face-cells mappings remain unchanged. Considering also that the mathematical formulas describing the shape of each face or cell are differentiable w.r.t. $\vec{v}^k$, one concludes that the derivative $\partial\vec{G}/\partial\vec{v}^k$ is well defined.

The finite difference method (FD) is a straightforward way to approximate these derivatives. Although it has widely been used in body-fitted meshes, its application is questionable in Cartesian meshes due to the great effect the step size value has on the final result. The chosen value should vary within limits that provide a sufficiently accurate gradient approximation, avoiding round-off errors. However, any value in this range may cause a wide enough displacement of the geometry's boundary to cover or uncover mesh cells. In such a case, the one-to-one mapping of faces and cells before and after the solid surface's displacement is meaningless, making the gradient's approximation through FD impractical.

Therefore, the mathematical differentiation of $\vec{G}$ is a reasonable alternative. In the method presented below, no assumptions are made regarding the mesh and geometry intersection, providing accurate expressions for the computation of $\partial\vec{G}/\partial\vec{v}^k$. Moreover, the developed software is robust and insensitive to the complexity of the geometry's shape. Finally, the computational cost of this approach is negligible since, contrary to the body-fitted meshes, the derivative is non-zero only to cut-cells.

The process followed for the differentiation of $\vec{G}$ is better explained by the example below. Fig. 2.25a shows a cube with dashed lines intersected by a solid surface consisting of 4 red-colored triangles. The resulted cut-cell is depicted in blue, and one of its solid faces is gray shaded. A top view, shown in fig. 2.25b, offers a better perspective of the shaded face. Its shape is determined by the relative position of the cube and the bolded triangle described by vertices $\vec{v}^1$, $\vec{v}^2$, and $\vec{v}^3$. Their intersection gives rise to the four blue points noted by $\vec{x}^l$, $l=1,\cdots,4$, the coordinates of which are computed by the Sutherland-Hodgman algorithm, section 2.4.1.

The coordinates of the centroid $\vec{x}^c$, shown in black, are exclusively defined by the location of these blue points. Therefore, the centroid and generally all members of vector $\vec{G}$ depend on the solid vertices $\vec{x}^l$ of the mesh, which are functions of the geometry's vertices $\vec{v}^k$. Thus,

$$\frac{\partial\vec{G}}{\partial\vec{v}^k}=\sum_{l=1}^{L}\frac{\partial\vec{G}}{\partial\vec{x}^l}\frac{\partial\vec{x}^l}{\partial\vec{v}^k} \tag{2.10}$$

where L is the number of vertices comprising the solid boundary of the mesh. Term $\partial\vec{x}^l/\partial\vec{v}^k$ is of major importance for the cut-cell method because it encapsulates the adjustment of the stationary Cartesian mesh on a continuously modified geometry. Thus, it represents the main difference between the present method and the conventional body-fitted approaches.

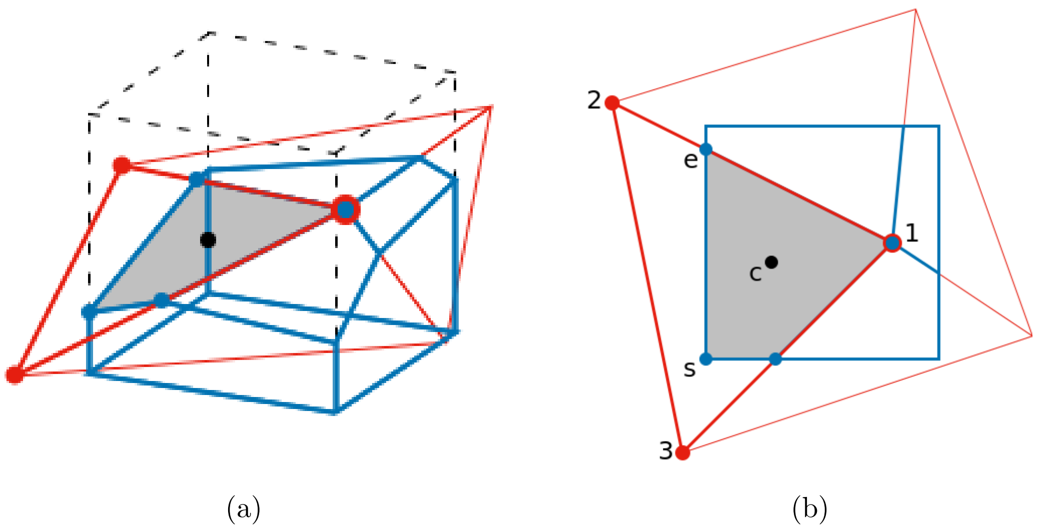


Figure 2.25: (a) The intersection of the red triangles and the cube, plotted with dashed black lines, results in the blue cut-cell. (b) Top view of the cut-cell. The blue points define a solid gray shaded face. They are located on the interior ($\vec{x}^s$), the edges ($\vec{x}^e$), or the vertices ($\vec{v}^1$) of the bolded triangle. The centroid $\vec{x}^c$ of the shaded face is presented with a black dot.

2.9.1 Differentiation of the Mesh Solid Boundary

Subsequently, the mathematical development of the second term on the r.h.s. of eq. 2.10 is examined. Three different approaches are followed based on the location of $\vec{x}^l$ on the triangulated solid surface. In the first case, the mesh node coincides with a vertex of the geometry. An example is given in fig. 2.25b, where $\vec{v}^1$ is part of both the blue face and the red triangle. Secondly, nodes such as $\vec{v}^e$ are located on the edge of a triangle. These nodes are originated from the intersection of this edge with a face of the Cartesian mesh. Finally, nodes that lie on the interior of the triangles' surface, e.g., $\vec{x}^s$, emerge when a Cartesian edge pierces the solid body's boundary.

The first case is trivial and reads,

$$\frac{\partial x_i^l}{\partial v_j^k} = \begin{cases} \delta_{ij}, & \vec{x}^l = \vec{v}^k \\ 0, & \text{otherwise} \end{cases}$$

where i, j indicate the Cartesian coordinates of each vertex and δ_{ij} is the Kronecker delta.

The differentiation of vertices located on the edge is presented via the example of $\vec{x}^{e}$. Its position changes due to the displacement of $\vec{v}^{1}$ and $\vec{v}^{2}$ always remaining on the cube's face. Therefore, one of its coordinates, expressed by index d, is constant and equal to c. In total, the vertex's position is expressed as

$$x_i^e = \begin{cases} \lambda\left(v_i^2 - v_i^1\right) + v_i^1, & i = \{1,2,3\} \setminus d \\ c, & i = d \end{cases}$$

where

$$\lambda = \frac{x_d^e - v_d^1}{v_d^2 - v_d^1}$$

Its derivatives are computed as

$$\begin{aligned} \frac{\partial x_i^e}{\partial v_j^1} &= (1-\lambda) M_{ij} \\ \frac{\partial x_i^e}{\partial v_j^2} &= \lambda M_{ij} \end{aligned} \tag{2.11}$$

Matrix M is defined as $M = I + \Phi^d$, where I is the identity matrix and $\Phi_{ij}^d = \phi_i \delta_{jd}$. Finally,

$$\phi_i = \frac{v_i^2 - v_i^1}{v_d^2 - v_d^1}$$

The coordinates of the third set of vertices are differentiated considering the arbitrary case of an intersection point $\vec{x}^{s}$ located in the interior of a triangle defined by vertices $\vec{v}^{l_1}$, $\vec{v}^{l_2}$, and $\vec{v}^{k}$. The derivative $\partial \vec{x}^{s} / \partial \vec{v}^{k}$ is computed under the condition that the intersection point remains aligned to the cube's edge after the infinitesimal displacement of $\vec{v}^{k}$. Therefore, the derivative is parallel to the edge's Cartesian direction, denoted by d, implying that

$$\frac{\partial \vec{x}^{s}}{\partial v_j^k} = a_j^k \vec{e}^{d} \tag{2.12}$$

where $\vec{e}^{d}$ represents the unit vector along the d axis. The unknown a_j^k is computed by considering that $\vec{x}^{s}$ lies on the triangle's plane,

$$\left(\vec{x}^{s} - \vec{v}^{l_1}\right) \cdot \vec{n}^{t} = 0 \tag{2.13}$$

where $\vec{n}^{\,t}$ is normal to the plane, and its magnitude equals the triangle's area,

$$\vec{n}^{\,t} = \frac{1}{2}\left(\vec{v}^{\,l_2} - \vec{v}^{\,l_1}\right) \times \left(\vec{v}^{\,k} - \vec{v}^{\,l_1}\right) \tag{2.14}$$

By differentiating eqs. 2.13 and 2.14 w.r.t. v_j^k one gets

$$\frac{\partial \vec{x}^{\,s}}{\partial v_j^k} \cdot \vec{n}^{\,t} + \left(\vec{x}^{\,s} - \vec{v}^{\,l_1}\right) \cdot \frac{\partial \vec{n}^{\,t}}{\partial v_j^k} = 0 \Leftrightarrow a_j^k \vec{e}^{\,d} \cdot \vec{n}^{\,t} - \left(\vec{v}^{\,l_1} - \vec{x}^{\,s}\right) \cdot \frac{\partial \vec{n}^{\,t}}{\partial v_j^k} = 0$$

and

$$\frac{\partial \vec{n}^{\,t}}{\partial v_j^k} = \frac{1}{2}\left(\vec{v}^{\,l_2} - \vec{v}^{\,l_1}\right) \times \frac{\partial \vec{v}^{\,k}}{\partial v_j^k} \Leftrightarrow \frac{\partial \vec{n}^{\,t}}{\partial v_j^k} = \frac{1}{2}\left(\vec{v}^{\,l_2} - \vec{v}^{\,l_1}\right) \times \vec{e}^{\,j}$$

Their combination leads to

$$\begin{aligned}
&a_j^k \vec{e}^{\,d} \cdot \vec{n}^{\,t} - \left(\vec{v}^{\,l_1} - \vec{x}^{\,s}\right) \cdot \frac{1}{2}\left[\left(\vec{v}^{\,l_2} - \vec{v}^{\,l_1}\right) \times \vec{e}^{\,j}\right] = 0 \Leftrightarrow \\
&a_j^k \vec{e}^{\,d} \cdot \vec{n}^{\,t} - \left(\vec{v}^{\,l_1} - \vec{x}^{\,s}\right) \cdot \frac{1}{2}\left[\left(\vec{v}^{\,l_2} - \vec{x}^{\,s}\right) \times \vec{e}^{\,j} - \left(\vec{v}^{\,l_1} - \vec{x}^{\,s}\right) \times \vec{e}^{\,j}\right] = 0 \Leftrightarrow \\
&a_j^k \vec{e}^{\,d} \cdot \vec{n}^{\,t} - \frac{1}{2}\left(\vec{v}^{\,l_1} - \vec{x}^{\,s}\right) \cdot \left[\left(\vec{v}^{\,l_2} - \vec{x}^{\,s}\right) \times \vec{e}^{\,j}\right] = 0 \Leftrightarrow \\
&a_j^k \vec{e}^{\,d} \cdot \vec{n}^{\,t} - \frac{1}{2}\vec{e}^{\,j} \cdot \left[\left(\vec{v}^{\,l_1} - \vec{x}^{\,s}\right) \times \left(\vec{v}^{\,l_2} - \vec{x}^{\,s}\right)\right] = 0 \Leftrightarrow \\
&a_j^k A_d^t - \vec{e}^{\,j} \cdot \hat{\vec{n}}^t A_k^t = 0 \Leftrightarrow a_j^k = \frac{A_k^t}{A_d^t} \hat{n}_j^t
\end{aligned}$$

where A_d^t represents the triangle's area projection to the d-direction, and A_k^t is the triangle's sub-area defined by $\vec{x}^{\,s}$, $\vec{v}^{\,l_1}$, and $\vec{v}^{\,l_2}$. Moreover, $\hat{\vec{n}}^t$ is the unit vector, normal to the triangle's plane. Finally, eq. 2.12 becomes

$$\frac{\partial x_i^s}{\partial v_j^k} = \frac{A_k^t}{A_d^t} \hat{n}_j^t \delta_{id} \tag{2.15}$$

2.9.2 Differentiation of Face and Cell Geometric Quantities

The computation of term $\partial \vec{G}/\partial \vec{x}^{\,l}$, eq. 2.10, for each member of $\vec{G}$ is presented below using the same notation and definitions introduced in subsection 2.5.4. According to that, the normal face vector is

$$\vec{n}^{\,f} = \frac{1}{2}\sum_{n=1}^{N} \vec{t}^{\,n} \times \vec{t}^{\,n'}$$

where n and n' are two subsequent face vertices. Its derivative w.r.t. the face vertex $\vec{x}^l$ is

$$\frac{\partial \vec{n}^f}{\partial \vec{x}^l} = \begin{bmatrix} 0 & \Delta t_3 & -\Delta t_2 \\ -\Delta t_3 & 0 & \Delta t_1 \\ \Delta t_2 & -\Delta t_1 & 0 \end{bmatrix} \tag{2.16}$$

where $\Delta\vec{t} = \vec{t}^s - \vec{t}^p$. Indices s, p identify the subsequent and previous vertices of $\vec{x}^l$. Moreover, $A_f = |\vec{n}^f|$, which implies

$$\frac{\partial A_f}{\partial x_j^l} = \hat{\vec{n}}^f \cdot \frac{\partial \vec{n}^f}{\partial x_j^l} \tag{2.17}$$

According to eq. 2.9, the face centroid is

$$\vec{x}^f = \frac{1}{A_f} \sum_{n=1}^{N} \vec{x}^{nn'} A_{nn'}$$

where $A_{nn'}$ and $\vec{x}^{nn'}$ are the area and centroid of the triangle defined by $\vec{x}^f$ and vertices n and n'. Its differentiation concludes to

$$\frac{\partial x_i^f}{\partial x_j^l} = \frac{1}{A_f} \left[\frac{\bar{A}_f + A_{pl} + A_{ln}}{3} \delta_{ij} + \sum_{n=1}^{N} \left(x_i^{nn'} - x_i^f \right) \frac{\partial A_{nn'}}{\partial x_j^l} \right] \tag{2.18}$$

where $\bar{A}_f = A_f/N$. Therefore, the computation of $\partial A_{nn'}/\partial x_j^l$ is needed. Terms $\partial A_{pl}/\partial x_j^l$ and $\partial A_{ls}/\partial x_j^l$ are separately discussed. The first of them is defined as

$$A_{pl} = \frac{1}{2} \left(\vec{t}^p \times \vec{t}^l \right) \cdot \hat{\vec{n}}^f$$

and its derivative is

$$\frac{\partial A_{pl}}{\partial x_j^l} = \frac{1}{2} \frac{\partial}{\partial x_j^l} \left(\vec{t}^p \times \vec{t}^l \right) \cdot \hat{\vec{n}}^f + \frac{1}{2} \left(\vec{t}^p \times \vec{t}^l \right) \cdot \frac{\partial \hat{\vec{n}}^f}{\partial x_j^l}$$

However, the derivative of the normal unit vector is represented by a vector lying on the face, and thus, the second addend is zero. After the proper mathematical development of the remaining term, the derivative is expressed as

$$\frac{\partial A_{pl}}{\partial x_j^l} = \frac{1}{2} \hat{\vec{n}}^f \times \left[\vec{t}^p + \frac{1}{N} \left(\vec{t}^l - \vec{t}^p \right) \right]$$

Similarly,

$$\frac{\partial A_{ls}}{\partial x_j^l} = -\frac{1}{2}\hat{\vec{n}}^f \times \left[\vec{t}^{\,s} + \frac{1}{N}\left(\vec{t}^{\,l} - \vec{t}^{\,s}\right)\right]$$

and

$$\frac{\partial A_{nn'}}{\partial x_j^l} = \frac{1}{2N}\hat{\vec{n}}^f \times \left(\vec{t}^{\,n'} - \vec{t}^{\,n}\right), \quad \forall\, n, n' \neq l$$

According to subsection 2.5.4, the volume of a cell is

$$\Omega_c = \sum_{f=1}^{F} \Omega_f$$

where

$$\Omega_f = \frac{1}{3}\left(\vec{x}^{\,f} - \vec{r}\right) \cdot \vec{n}^{\,f}$$

is the volume of each pyramid to which the cell is divided. The cell's volume is independent of $\vec{r}$, and thus, it will be considered constant during the differentiation. Therefore,

$$\frac{\partial \Omega_c}{\partial x_j^l} = \sum_{f=1}^{\tilde{F}} \frac{\partial \Omega_f}{\partial x_j^l} \tag{2.19}$$

and

$$\frac{\partial \Omega_f}{\partial x_j^l} = \frac{1}{3}\left[\frac{\partial \vec{x}^{\,f}}{\partial x_j^l} \cdot \vec{n}^{\,f} + \left(\vec{x}^{\,f} - \vec{r}\right) \cdot \frac{\partial \vec{n}^{\,f}}{\partial x_j^l}\right] \tag{2.20}$$

The sum on the r.h.s. of eq. 2.19 consists exclusively of $\tilde{F}$ in number pyramids containing vertex $\vec{x}^{\,l}$. The derivatives of $\vec{n}^{\,f}$ and $\vec{x}^{\,f}$ on the r.h.s. of eq. 2.20 are computed by eqs. 2.16 and 2.18, respectively. Additionally, the arithmetic mean of cell vertices can be used to compute $\vec{r}$.

Finally, based on subsection 2.5.4, the cell's centroid is

$$\vec{x}^{\,c} = \frac{3}{4\Omega_c}\sum_{f=1}^{F}\left(\vec{x}^{\,f} - \vec{r}\right)\Omega_f + \vec{r}$$

Once again, the centroid's value is independent of $\vec{r}$, and thus, it is considered constant during the differentiation process. Hence,

$$\frac{\partial \vec{x}^{\,c}}{\partial x_j^l} = \frac{3}{4\Omega_c}\sum_{f=1}^{\tilde{F}}\left[\frac{\partial \vec{x}^{\,f}}{\partial x_j^l}\Omega_f + \left(\vec{x}^{\,f} - \vec{r}\right)\frac{\partial \Omega_f}{\partial x_j^l}\right] - \frac{1}{\Omega_c}\left(\vec{x}^{\,c} - \vec{r}\right)\frac{\partial \Omega_c}{\partial x_j^l} \tag{2.21}$$

The derivatives of $\vec{x}^f$, Ω_f, and Ω_c on the r.h.s. are computed from eqs. 2.18, 2.20, and 2.19, respectively.

Chapter 3

Numerical Discretization of the Navier-Stokes Equations

This chapter presents the mathematical formulation of the flow equations, their discretization, and a numerical method to solve the resulting algebraic system. The examined flow model consists of the steady or unsteady Navier-Stokes equations for either compressible or incompressible, inviscid or laminar flows. The artificial compressibility method is applied to solve the incompressible flow equations. The presented discretization scheme takes advantage of the Cartesian mesh data structure dictated by the cut-cell method. It is based on a cell-centered, second-order finite volume method employing the MUSCL scheme. The Roe's approximate Riemann solver is used to compute inviscid fluxes, and second-order accuracy is attained by reconstructing the flow variables at mesh faces based on a Taylor series expansion. The required gradients of flow field variables are computed through the least squares technique. Orthogonal correction is utilized to compute the velocity and temperature gradients used for the viscous flux discretization. In unsteady simulations, the temporal term is discretized by applying a first-order Euler scheme, and time marching is based on a dual time-stepping method. Special treatment is made for cells that appear or disappear from the fluid domain in case of flows involving moving bodies. The Newton-Raphson algorithm is applied for solving the resulting non-linear system. The last section describes the ghost-cell method for steady and unsteady flow simulations and provides details about the fluid-solid interface treatment.

3.1 Compressible Fluid Flow Model

The Navier-Stokes equations and the mass and energy conservation laws for a 3D unsteady compressible flow of a perfect gas can be expressed in a Cartesian coordinate system (x_1, x_2, x_3) as [173]

$$\frac{\partial U_i}{\partial t} + \frac{\partial f_{ik}^{inv}}{\partial x_k} - \frac{\partial f_{ik}^{vis}}{\partial x_k} = 0, \quad i = 1, \cdots, 5, \; k = 1, \cdots, 3 \tag{3.1}$$

where

$$\vec{U} = \begin{bmatrix} \rho \\ \rho v_1 \\ \rho v_2 \\ \rho v_3 \\ \rho E \end{bmatrix}, \quad \vec{f}_k^{inv} = \begin{bmatrix} \rho v_k \\ \rho v_k v_1 + \delta_{1k} p \\ \rho v_k v_2 + \delta_{2k} p \\ \rho v_k v_3 + \delta_{3k} p \\ \rho v_k h_t \end{bmatrix}, \quad \vec{f}_k^{vis} = \begin{bmatrix} 0 \\ \tau_{1k} \\ \tau_{2k} \\ \tau_{3k} \\ v_j \tau_{jk} + q_k \end{bmatrix}$$

The Einstein notation has been used, according to which repeated indices imply summation. $\vec{U}$, $\vec{f}_k^{inv}$ and $\vec{f}_k^{vis}$ are the conservative variables, inviscid flux and viscous flux vectors, respectively. Primitive variables ρ, v_i and p stand for the fluid's density, velocity components and pressure. The corresponding primitive variables' vector is defined as $\vec{V} = [\rho,\ v_1,\ v_2,\ v_3,\ p]$. E stands for the energy per unit mass and h_t is the total enthalpy. These are related through the equation

$$h_t = E + \frac{p}{\rho}$$

For a Newtonian fluid, the viscous stress tensor τ is expressed as

$$\tau_{ik} = \mu \left(\frac{\partial v_i}{\partial x_k} + \frac{\partial v_k}{\partial x_i} - \frac{2}{3} \delta_{ik} \frac{\partial v_m}{\partial x_m} \right)$$

with μ being the fluid's dynamic viscosity and δ_{ik} the Kronecker delta. Heat flux q_m is given by the Fourier's law

$$q_m = k \frac{\partial T}{\partial x_m}$$

where k and T stand for the fluid's thermal conductivity and temperature, respectively. The equation of state

$$p = \rho R T$$

completes the above system of PDEs, where R is the specific gas constant. Finally, for the conservative to primitive set of flow variables conversion,

$$\rho E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho v_i^2$$

where γ is the specific heat ratio, is used.

An appropriate set of boundary conditions accompanies the above system of PDEs. They depend on the type of each boundary and the physics behind every specific application. In external aerodynamics, such as the flow simulation around an isolated wing, far-field boundaries are positioned far away from the examined geometry. Far-field flow is considered constant, and five flow variables are set at ghost-nodes placed along the mesh boundary. On the other hand, in internal aerodynamics, such as flow simulation within ducts or pumps, a different set of flow quantities is imposed at the inlet and outlet. For subsonic inlet boundaries total pressure (p_t), total temperature (T_t) and two angles (α_{yaw}, α_{pitch}) are specified. Primitive variables are computed by interpolating the velocity magnitude ($|\vec{v}|$) from the flow domain interior. This process starts from the temperature computation $T = T_t - |\vec{v}|^2/(2c_p)$ at the boundary, where c_p is the specific heat capacity at constant pressure. Then, the primitive variables are computed as

$$\begin{aligned}
p^{BC} &= p_t \left(\frac{T}{T_t}\right)^{\frac{\gamma}{\gamma-1}} \\
\rho^{BC} &= \frac{p^{BC}}{RT} \\
v_1^{BC} &= |\vec{v}| sin(\alpha_{pitch}) cos(\alpha_{yaw}) \\
v_2^{BC} &= |\vec{v}| sin(\alpha_{pitch}) sin(\alpha_{yaw}) \\
v_3^{BC} &= |\vec{v}| cos(\alpha_{pitch})
\end{aligned}$$

An alternative is the imposition of entropy (s) and the three velocity components. Density is extrapolated from the interior and, then pressure is computed as $p = s\rho^\gamma$. However, in real-world applications, inlet entropy is usually unknown. Its value can be found by the user-defined density (ρ_0) and pressure (p_0) approximations, so that $s = p_0/\rho_0^\gamma$. For supersonic inlet flow boundaries, all primitive variables are user-defined. At the outlet, only pressure is specified in the case of a subsonic flow. The other four primitive variables are extrapolated from the interior. On the other hand, in a supersonic outlet case, no boundary conditions are imposed at the exit. Wall boundary conditions depend on the used flow model. For inviscid flows, the no-penetration condition applies, and thus the flow and wall's normal velocity components (v_i^w) are equal,

$$v_i n_i = v_i^w n_i$$

where n_i is the normal to the wall component along the x_i direction. In a viscous flow case, the no-slip condition is employed as well, namely the flow and wall velocity vectors are equal,

$$v_i = v_i^w$$

Moreover, solid walls are adiabatic, thus

$$q_k n_k = 0$$

Finally, boundary conditions along a symmetry plane are

$$\begin{aligned} \left.\frac{\partial \rho}{\partial n}\right|_{BC} &= 0 \\ \left.\frac{\partial p}{\partial n}\right|_{BC} &= 0 \\ v_i^{BC} &= v_i - 2v_k n_k n_i \end{aligned}$$

where $\vec{n}$ is the perpendicular to the symmetry plane unit vector, $\vec{v}$ is the fluid's velocity, and $\vec{v}^{BC}$ is the modified velocity at the boundary.

3.2 Discretization of the Steady Compressible Laminar Equations

This section deals with the discretization of the governing eqs. 3.1 in steady flows. Its structure is separated into smaller parts discussing specific details of the discretization scheme. Firstly, the finite volume method implementation is presented. The next subsection focuses on the flux discretization, the Riemann problem and its approximate solution by the Roe scheme. Then, the second-order MUSCL method is demonstrated and the limiter's use is briefly explained. Limiters are presented in detail in the following subsection. A separate subsection deals with the flow variables gradient computation by using the least squares method. After that, the flux computation over the boundary faces is examined. Thereafter, the discretization of viscous terms is presented. Finally, stability and convergence issues are discussed.

3.2.1 The Finite Volume Method

Over the years, numerous methods have been developed to discretize hyperbolic systems of PDEs. Therefore, the steady flow equations are reformulated as a hyperbolic system to take advantage of such accurate solvers [306]. Accordingly, a pseudo-time derivative of the conservative variables is added to the governing equations. Their final form is

$$\frac{\partial U_i}{\partial \tau} + \frac{\partial f_{ik}}{\partial x_k} = 0, \quad i = 1, \cdots, 5, \; k = 1, \cdots, 3$$

where $\vec{f}_k = \vec{f}_k^{inv} - \vec{f}_k^{vis}$ and pseudo-time is denoted by τ to be distinguished from the real-time t. The above modification does not affect the flow solution because the extra term vanishes after convergence is achieved. The flow equations discretization is based on the cell-centered finite volume method [177]. On a cut-cell basis, each finite volume (Ω) is considered a region placed entirely inside the mesh fluid part, restricted by solid boundaries and internal mesh faces. It can be an internal orthogonal polyhedron or a boundary cut-cell of an arbitrary polyhedral shape. Fig. 3.1 shows a mesh detail with two differently colored cells. Conservative variables ($\vec{U}_P$, $\vec{U}_Q$) are stored at their centroids. Integrating the above equation over $\Omega \times \Delta\tau$ and applying the Green-Gauss theorem in the temporal and spatial terms, one gets

$$\int_{\tau}^{\tau+\Delta\tau} \int_{\Omega} \frac{\partial U_i}{\partial \tau} d\Omega d\tau + \int_{\tau}^{\tau+\Delta\tau} \int_{\Omega} \frac{\partial f_{ik}}{\partial x_k} d\Omega d\tau = 0 \Leftrightarrow$$
$$\int_{\Omega} U_i d\Omega \bigg|_{\tau+\Delta\tau} - \int_{\Omega} U_i d\Omega \bigg|_{\tau} + \int_{\tau}^{\tau+\Delta\tau} \int_{S} f_{ik} n_k dS d\tau = 0$$

where $\int_\Omega$ and $\int_S$ are shortcuts for the triple volume and double surface integrals, $\Delta\tau$ is the pseudo-time discretization step, S is the finite volume boundary surface and $\vec{n}$ is the outward pointing surface unit normal vector. The surface integral can be written as a summation of integrals over all edges or faces of a polygonal or polyhedral finite volume in a 2D or 3D case. Each of them is represented by the averaged flux vector $\vec{\bar{f}}_k^m$ leading to the following formula

$$\int_{\Omega} U_i d\Omega \bigg|_{\tau+\Delta\tau} - \int_{\Omega} U_i d\Omega \bigg|_{\tau} + \int_{\tau}^{\tau+\Delta\tau} \left(\sum_{m=1}^{M} \bar{f}_{ik}^m n_k^m \Delta S^m \right) d\tau = 0$$

where

$$\bar{f}_{ik}^{m} = \frac{\int_{S_m} f_{ik} dS}{\Delta S^m}$$

and M is the number of finite volume's faces. By defining the mean values

$$\bar{U}_i^n = \frac{\int_\Omega U_i d\Omega\big|_\tau}{\Omega}, \text{ and } \bar{U}_i^{n+1} = \frac{\int_\Omega U_i d\Omega\big|_{\tau+\Delta\tau}}{\Omega}$$

the equation is written as

$$\frac{\bar{U}_i^{n+1} - \bar{U}_i^n}{\Delta\tau}\Omega + \sum_{m=1}^{M}\left(\frac{1}{\Delta\tau}\int_\tau^{\tau+\Delta\tau} \bar{f}_{ik}^{m} n_k^m d\tau \Delta S^m\right) = 0 \tag{3.2}$$

The pseudo-time integral computation is quite different for the inviscid and viscous flux. Therefore, it is described separately in subsections 3.2.2 and 3.2.7.

3.2.2 Convective Flux Discretization Scheme

The computation of the integral in eq. 3.2 along every mesh face is challenging since two neighboring finite volumes are met, creating a discontinuity in the flow field. For example, fig. 3.1 shows a 2D mesh detail, where an edge is separating finite volumes P and Q. On its left-hand side, flow variables are equal to $\vec{U}^L$, being different than the right-hand values $\vec{U}^R$. Generally, $\vec{U}^L$ and $\vec{U}^R$ may differ from $\vec{U}^P$ and $\vec{U}^Q$. Their values depend on the assumption on the rules the flow distribution follows inside the finite volume. The discontinuity problem is physically well represented by the 1D so-called shock-tube problem along the segment connecting the two centroids, where two stationary gases of different pressure and density are placed in a tube separated by a diaphragm. The diaphragm removal generates a nearly centered wave system that typically consists of a rarefaction wave, a contact discontinuity, and a shock wave. The general case, in which the gases are initially allowed to move, having different velocities $\vec{v}^L$ and $\vec{v}^R$, is the renowned Riemann problem. Its analytical solution allows for the exact computation of the above integral. Assuming that the chosen time step $\Delta\tau$ is small enough and, thus, prevents the wave interaction between the finite volume faces, the Riemann problem local solution suggests constant flow values $\vec{U}^{nm}$ along $\Delta\tau$, where n is the pseudo-time step counter. Thus,

$$\sum_{m=1}^{M}\frac{1}{\Delta\tau}\int_\tau^{\tau+\Delta\tau} \bar{f}_{ik}^{inv,m} n_k^m d\tau \Delta S^m = \sum_{m=1}^{M} \bar{f}_{ik}^{inv,m}(\vec{U}^{nm}) n_k^m \Delta S^m$$

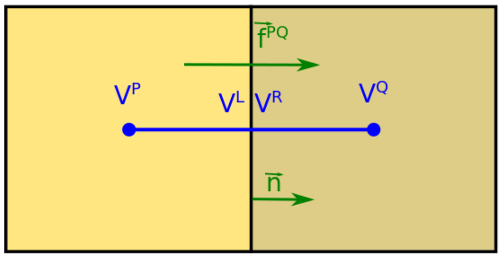


Figure 3.1: Mesh detail of two neighboring cells P and Q. Flux $\vec{f}^{PQ}$ is computed on their common edge. A discontinuity appears along the edge separating the $\vec{V}^L$ and $\vec{V}^R$ flow conditions. Edge normal unit vector $\vec{n}$ is also shown.

The discretization presented above is called the Godunov method [109]. Its integral form is easily implemented in an unstructured mesh, allowing for discontinuous solutions to appear. However, it suffers from two weaknesses. Firstly, like most discretization schemes, $\Delta\tau$ is restricted by an upper limit relying on the maximum absolute wave velocity $S^n_{max} = |v^n| + c^n$ throughout the finite volume. For a 1D cell of length h, this upper limit is

$$\Delta\tau < \frac{h}{S^n_{max}} \tag{3.3}$$

The meaning of the above inequality is evident in fig. 3.2, where a 1D cross-section along the line connecting the two centroids, depicted in fig. 3.1, is shown in a space-time plane. The P finite volume is surrounded by waves created on its boundaries. Time step $\Delta\tau$ should be quite small to impede the interaction of the two groups of waves, allowing for the two Riemann problems separate treatment. This upper limit delays the convergence, increasing the overall computational cost. To overcome this drawback, $\vec{U}^{(n+1)m}$ can be used instead of $\vec{U}^{mn}$, giving rise to an implicit discretization scheme.

Secondly, the Riemann problem analytical solution is required. Although exact Riemann solvers are available, they involve iterative procedures increasing the computational cost. In practical applications, the Riemann problem should be solved millions of times, making it impossible to implement an exact Riemann solver. On the other hand, approximate, non-iterative solvers have the potential to provide accurate enough numerical solutions at a reasonable computational cost.

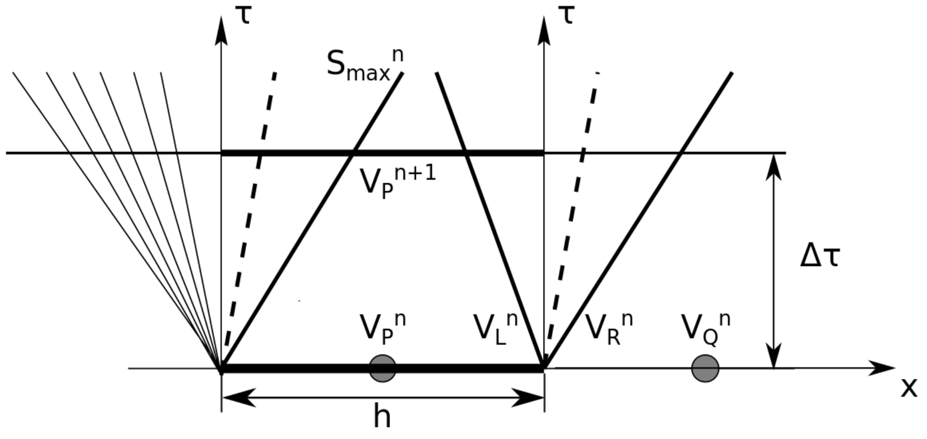


Figure 3.2: A wave system representation in a space-time plane generated by discontinuities appeared at both boundaries of the 1D finite volume P of length h. Dashed, bold, and fan-type lines represent the contact, shock and rarefaction waves, respectively. The neighboring finite volume Q is also shown. Solution of the local Riemann problem caused due to the V_L^n and V_R^n discontinuity is needed by the Godunov averaging method. Pseudo-time step $\Delta\tau$ is chosen such as the wave of maximum speed S_{max}^n within P is not affecting the Riemann solution of the right wave system. Post- processed figure taken from [306].

One of the most well–known approximate Riemann solvers was introduced by Roe in 1981 [262]. Many refinements and corrections have been presented, such as the Roe-Pike approach [263] or the Glaister extension for the time-dependent Euler equations [106]. Roe's approach has been applied to a great variety of physical problems proving its reliability and robustness. According to the Roe's approximate Riemann solver, the flux is computed by using one of the two following alternatives [306], [309],

$$\bar{f}_{ik}^{inv,m} n_k^{PQ} = \frac{1}{2}\left(f_{ik}^L + f_{ik}^R\right) n_k^{PQ} - \frac{1}{2}\left|\tilde{A}_{ijk} n_k^{PQ}\right| \left(U_j^R - U_j^L\right) \tag{3.4}$$

$$\bar{f}_{ik}^{inv,m} n_k^{PQ} = \frac{1}{2}\left(f_{ik}^P + f_{ik}^Q\right) n_k^{PQ} - \frac{1}{2}\left|\tilde{A}_{ijk} n_k^{PQ}\right| \left(U_j^R - U_j^L\right) \tag{3.5}$$

where $\vec{n}^{PQ}$ is the unit vector normal to the surface separating finite volumes P, Q, always directed from the L to the R side. A_k is the Jacobian matrix of the governing equations system along the k-direction,

$$A_{ijk} = \frac{\partial f_{ik}}{\partial U_j} \tag{3.6}$$

The Jacobian matrix projected to the normal direction leads to the diagonalizable matrix $A_k n_k$. Due to the hyperbolic nature of the governing PDEs, $A_k n_k$ has real eigenvalues. Its absolute counterpart is

$$|A_{ijk} n_k| = P_{il} \, |\Lambda_{lm}| \, P^{-1}_{mj} \tag{3.7}$$

Column vectors of P are the right eigenvectors of $A_k n_k$ and $|\Lambda| = diag(|\lambda_1|, |\lambda_2|, \cdots, |\lambda_5|)$ where $|\lambda_i|$ are the absolute eigenvalues of $A_k n_k$. The P and Λ analytical expressions can be found in [130] and Appendix D. Finally, the tilde symbol denotes the use of the Roe averages for the $\left|\tilde{A}_{ijk} n_k\right|$ computation. These are given as

$$\begin{aligned} \tilde{\rho} &= \sqrt{\rho^L \rho^R} \\ \tilde{v}_i &= \frac{\sqrt{\rho^L} v_i^L + \sqrt{\rho^R} v_i^R}{\sqrt{\rho^L} + \sqrt{\rho^R}} \\ \tilde{h}_t &= \frac{\sqrt{\rho^L} h_t^L + \sqrt{\rho^R} h_t^R}{\sqrt{\rho^L} + \sqrt{\rho^R}} \end{aligned} \tag{3.8}$$

Very useful is also the relation giving the Roe averaged sound speed,

$$\tilde{c} = \sqrt{(\gamma - 1)\tilde{h}_t - \frac{1}{2}\tilde{v}_i^2}$$

A detailed proof of eq. 3.4 is presented in Appendix E. All values being part of the Roe scheme discretization correspond to time $\tau + \Delta\tau$. Superscript $n+1$ is omitted for the sake of brevity. Eq. 3.4 is more accurate than eq. 3.5, but it may cause convergence issues, especially in transonic flows where strong shock waves occur. This behavior is avoided by using eq. 3.5, which is proved to also provide second-order accuracy results [18]. However, numerical experiments have shown that eq. 3.5 causes spurious pressure oscillations close to the solid boundary of a Cartesian mesh. Thus, its use should be avoided whenever possible.

3.2.3 The second-order MUSCL Method

Numerical simulations of real-world applications require at least second-order accuracy algorithms to preserve reliability. In this thesis, the Monotone Upstream–centered Scheme for Conservation Laws (MUSCL) introduced by van Leer [318] is applied to compute the left (L) and right (R) conditions that appeared in eqs. 3.4 and 3.5. However, according to Godunov's theorem [108], second or higher-order accurate linear schemes are prone to create spurious oscillations in the flow field, especially in the vicinity of large gradients. This condition is overcome by constructing non–linear, oscillation–free discretization methods, called Total Variation Diminishing (TVD) schemes, offering second-order accuracy in smooth parts of the solution. These methods try to mimic the exact solution of the scalar conservation laws by preventing the total variation increase in pseudo-time. Appendix G proves that monotone schemes, such as MUSCL, belong to the class of TVD schemes. Non-linearity in the MUSCL scheme is introduced by the use of slope limiters, which enforce monotonicity. Consequently, the MUSCL approach not only implies second-order spatial accuracy but also avoids the creation of unphysical oscillations during the simulation. It is essential to mention that the above theoretical basis is mathematically developed only for scalar 1D cases. However, experience over many decades shows that this theory serves well as a guideline for extending the above ideas in multidimensional applications.

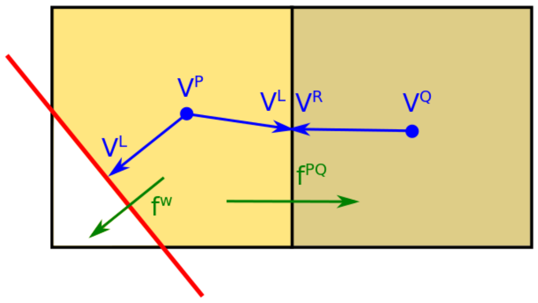
According to MUSCL, the primitive flow variables follow a linear distribution inside a finite volume as follows

$$V_i(\vec{x}) = V_i^c + \phi_i \sum_{j=1}^{2\,or\,3} \left.\frac{\partial V_i}{\partial x_j}\right|_c (x_j - x_j^c)$$

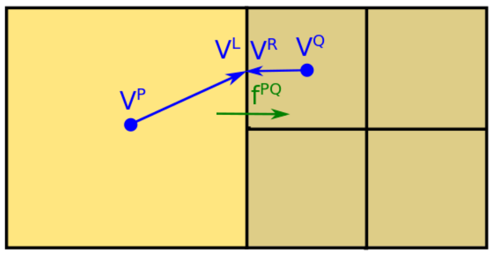
where ϕ_i is the slope limiter for each variable. Index c denotes the finite volume's centroid. Based on this assumption, flow values $\vec{U}^L$ and $\vec{U}^R$ correspond to the extrapolated variables $\vec{U}^P$ and $\vec{U}^Q$ at the two sides of each face.

$$\begin{aligned} V_i^L =& V_i^P + \phi_i^P \sum_{j=1}^{2\,or\,3} \left.\frac{\partial V_i}{\partial x_j}\right|_P (x_j^f - x_j^P) \\ V_i^R =& V_i^Q + \phi_i^Q \sum_{j=1}^{2\,or\,3} \left.\frac{\partial V_i}{\partial x_j}\right|_Q (x_j^f - x_j^Q) \end{aligned} \tag{3.9}$$

where $\vec{x}^f$ are the coordinates of the edge or face centroid for 2D or 3D cases, respectively. Fig. 3.3 depicts the extrapolation by using blue arrows in two different cases. Blue arrows geometrically present the extrapolation in fig. 3.3 in cases where centroids P and Q are not inlined due to geometrical intersections or cells surrounding the face are of different coarsening levels. Finally, an important fact is that, although the MUSCL procedure is second-order accurate, close to the solid boundary the limiter takes very low values at a significant number of cut-cells, reducing the discretization accuracy locally to first-order. The next subsections are dealing with the computation of the limiter function and the primitive variables' gradient used in eqs. 3.9.



(a)



(b)

Figure 3.3: Two cases of $\vec{V}^P$ and $\vec{V}^Q$ variables extrapolation to approximate $\vec{V}^L$ and $\vec{V}^R$ conditions used for flux computation at internal ($\vec{f}^{PQ}$) or boundary ($\vec{f}^{w}$) edges. P and Q centroids are not aligned due to (a) solid boundary intersections and (b) different cells' size.

3.2.4 Limiters

The benefits using limiter in eqs. 3.9 are already discussed in subsection 3.2.3. This subsection focuses on two ways to compute limiters. By definition, limiters guarantee that the extrapolated flow variables' magnitude does not exceed the adjacent neighbor's cell-centered solution. Various limiters have been proposed in the literature with different characteristics and behavior. The proper limiter choice depends on the case and is selected on a trial and error basis. Two kinds of limiters for unstructured meshes are used in this thesis. The first one (ϕ_{BJ}) is introduced by Barth and Jespersen [26] and uses the non-differential min function which may harm the convergence of non-linear systems. Although the limiter enforces monotonicity, it reduces the scheme's accuracy because it remains active in smooth regions of the solution. Limiter's ability to prevent the generation of new local extrema is presented in Appendix H. The second one (ϕ_V), proposed by Venkatakrishnan [321], is based on ϕ_{BJ} but is differentiable. It can also revert to a scheme without limiter in smooth regions, increasing the discretization accuracy, but it cannot maintain the solution's monotonicity. Both limiters' expressions are shown below

$$\phi_{BJ} = \begin{cases} min\left(1, \frac{\Delta_c}{\Delta_f}\right) & |\Delta_f| \geqslant eps \\ 1 & |\Delta_f| < eps \end{cases} \tag{3.10}$$

$$\phi_V = \begin{cases} \frac{1}{\Delta_f} \frac{\left(\Delta_c^2+\epsilon\right)\Delta_f + 2\Delta_f^2\Delta_c}{\Delta_c^2+2\Delta_f^2+\Delta_f\Delta_c+\epsilon} & |\Delta_f| \geqslant eps \\ 1 & |\Delta_f| < eps \end{cases} \tag{3.11}$$

where

$$\Delta_c = \begin{cases} V_{i_{max}} - V_i^c & \Delta_f \geqslant eps \\ V_{i_{min}} - V_i^c & \Delta_f < eps \end{cases}$$

$$\Delta_f = V_i^f - V_i^c = \left.\frac{\partial V_i}{\partial x_j}\right|_c (x_j^f - x_j^c)$$

and

$$\epsilon = (KD)^3$$

The user-defined *eps* value takes on a very small value (e.g., 10^{-14}), $K = 0.3$, and D is the cell's hydraulic diameter computed as $D = 6\Omega/S$. For each cell, flow variables $\vec{V}^c$ stored at the centroid $\vec{x}^c$ are extrapolated at each cell's face centroid $\vec{x}^f$ by using the first-order Taylor expansion. The resulting values $\vec{V}^f$ are used to compute Δ_f. Depending on Δ_f sign, Δ_c is computed by using $V_{i_{max}}$ or $V_{i_{min}}$ standing for the maximum or minimum value stored in neighboring cells' centroid. Neighboring cells are considered all cells that share a common face with the referred finite volume. In case a finite volume's face is part of a mesh boundary, a fake node is used as the corresponding $\vec{V}_i^c$ value. Its flow variables are computed by using the extrapolated on the face values $\vec{V}^f$ which are modified according to the corresponding boundary conditions presented in section 3.1. The procedure mentioned above is applied for each finite volume's face giving a different limiter's value, the minimum of which is its final finite volume's limiter value. The algorithm is repeated for each primitive variable.

3.2.5 Gradient Computation Using the Least Squares Method

Computation of primitive flow variables at the face's centroid via eq. 3.9 requires the cell-centered gradient $\left.\frac{\partial V_i}{\partial x_j}\right|_P$ evaluation in an arbitrary finite volume P. There are two common techniques for its computation, the Green–Gauss theorem or the least-squares approach. Despite its relative complexity, the second method is used in this study due to its higher accuracy [298]. It computes a gradient by best approximating the flow variables stored in the neighboring cells through the Taylor expansion. This request is obtained by minimizing a cost function for each primitive variable i given by

$$E_i = \sum_{m=1}^{M} w^m \left[V_i^m - V_i^P - \left.\frac{\partial V_i}{\partial x_j}\right|_P \left(x_j^m - x_j^p\right) \right]^2$$

where M is the number of cell's faces. If the face is internal, then $\vec{x}^m$ and $\vec{V}^m$ are the neighboring cell's centroid and flow variables, respectively. Otherwise, a fake node is defined, positioned at the face centroid. Firstly, $\vec{V}^P$ is copied to the fake node and, then, transformed by considering the appropriate boundary conditions giving rise to the corresponding $\vec{V}^m$ flow vector. The weight coefficient w^m places greater importance to the stencil of neighbors being nearby. It is usually set to $w^m = 1/\left|\vec{x}^m - \vec{x}^P\right|^2$. Numerical experiments have shown that $w^m = 1$ also gives accurate results. The cost function's minimum is computed by nullifying its derivatives w.r.t. the unknown gradient components leading to a 3×3 algebraic system $A\vec{x} = \vec{b}_i$ given by

$$\begin{bmatrix} \sum w^m \Delta x_1^m \Delta x_1^m & \sum w^m \Delta x_1^m \Delta x_2^m & \sum w^m \Delta x_1^m \Delta x_3^m \\ \sum w^m \Delta x_2^m \Delta x_1^m & \sum w^m \Delta x_2^m \Delta x_2^m & \sum w^m \Delta x_2^m \Delta x_3^m \\ \sum w^m \Delta x_3^m \Delta x_1^m & \sum w^m \Delta x_3^m \Delta x_2^m & \sum w^m \Delta x_3^m \Delta x_3^m \end{bmatrix} \begin{bmatrix} \frac{\partial V_i}{\partial x_1} \\ \frac{\partial V_i}{\partial x_2} \\ \frac{\partial V_i}{\partial x_3} \end{bmatrix} = \begin{bmatrix} \sum w^m \Delta x_1^m \Delta V_i^m \\ \sum w^m \Delta x_2^m \Delta V_i^m \\ \sum w^m \Delta x_3^m \Delta V_i^m \end{bmatrix}$$

where $\Delta x_i^m = x_i^m - x_i^P$ and $\Delta V_i^m = V_i^m - V_i^P$. The system has a unique solution provided that A is invertible. At the end of each pseudo-time step, flow variables are recomputed and their derivatives are given by solving the above system for each finite volume P and each primitive variable i. This procedure is computationally costly and can be avoided by differently formulating the system. Firstly, a new matrix and vector are defined as

$$B = \begin{bmatrix} w^1\Delta x_1^1 & w^2\Delta x_1^2 & \cdots & w^m\Delta x_1^m \\ w^1\Delta x_2^1 & w^2\Delta x_2^2 & \cdots & w^m\Delta x_2^m \\ w^1\Delta x_3^1 & w^2\Delta x_3^2 & \cdots & w^m\Delta x_3^m \end{bmatrix}, \quad \vec{c}_i = \begin{bmatrix} \Delta V_i^1 \\ \Delta V_i^2 \\ \vdots \\ \Delta V_i^m \end{bmatrix}$$

where the m index repetition in matrix B does not imply summation. By definition, $\vec{b}_i = B\vec{c}_i$ which yields

$$A\vec{x} = \vec{b}_i \Leftrightarrow A\vec{x} = B\vec{c}_i \Leftrightarrow \vec{x} = A^{-1}B\vec{c}_i \Leftrightarrow \vec{x} = W\vec{c}_i$$

where

$$W = A^{-1}B = \begin{bmatrix} \vec{W}_1 \\ \vec{W}_2 \\ \vec{W}_3 \end{bmatrix} \tag{3.12}$$

Then, the gradient is computed as

$$\frac{\partial V_i}{\partial x_j} = \vec{W}_j.\vec{c}_i \tag{3.13}$$

where $\vec{W}_j$ is a function of only geometrical quantities, and $\vec{c}_i$ contains exclusively flow variables. Therefore, $\vec{W}_j$ remains constant during the convergence procedure and can be computed at the beginning of the flow simulation. Then, at the end of each time step, $\vec{c}_i$ changes, and the variables' gradient is computed anew at the cost of an internal product (eq. 3.13) instead of a 3×3 system solution.

3.2.6 Flux Computation at the Boundary Faces

So far, the convection term discretization for internal faces has been presented in detail. The flux computation at the boundary faces differs, depending on the kind of the corresponding boundary conditions. For wall boundary faces, the Roe scheme is not used and, thus, additional dissipation is avoided. Wall fluxes ($\bar{f}_{ik}^{w,m} n_k$) are computed on the centroid of the face being part of the intersection between the solid bodies and the Cartesian mesh. Such a case is illustrated in fig. 3.3a. The presented discretization's capability indicates the cut-cell method's superiority compared to other immersed boundary methods. Wall flux expression on a stationary boundary is

$$\vec{\bar{f}}_{k}^{w,m} n_k = \begin{bmatrix} 0 \\ pn_1 \\ pn_2 \\ pn_3 \\ 0 \end{bmatrix} \tag{3.14}$$

Pressure is computed by extrapolating its value from the cell to the face centroid, fig. 3.3a. The exact wall boundary representation through the precise cut-cell construction described in chapter 2 increases the extrapolation accuracy and, therefore, the solid boundary conditions imposition. On the other hand, for the inlet, outlet, and symmetry conditions, the Roe scheme is preferred due to its ability to increase stability and drive the governing equations to convergence. Eq. 3.4 is transformed as

$$\bar{f}_{ik}^{inv,BC,m} n_k = \frac{1}{2}\left(f_{ik}^{L} + f_{ik}^{BC}\right) n_k - \frac{1}{2}\left|\tilde{A}_{ijk} n_k\right| \left(U_j^{BC} - U_j^{L}\right) \tag{3.15}$$

where left (L) variables are computed through extrapolation from the boundary cell's barycenter, and $\vec{U}^{BC}$ is computed by imposing the boundary conditions presented in section 3.1, on the $\vec{U}^{L}$ flow variables.

3.2.7 Diffusive Flux Discretization Scheme

This subsection deals with the discretization of the integral's viscous part appeared in eq. 3.2. More specifically,

$$\sum_{m=1}^{M}\left(\frac{1}{\Delta\tau}\int_{\tau}^{\tau+\Delta\tau} \bar{f}_{ik}^{vis,m} n_k^m d\tau \Delta S^m\right) = \sum_{m=1}^{M}\left(\bar{f}_{ik}^{vis,m}(\vec{U}^m, \left.\frac{\partial \vec{U}}{\partial \vec{x}}\right|_m) n_k^m \Delta S^m\right)$$

Superscript $n+1$ in terms $\vec{U}^m$ and $\left.\frac{\partial \vec{U}}{\partial \vec{x}}\right|_m$ is omitted for the sake of brevity. Based on eq. 3.1, flux $\bar{f}_{ik}^{vis,m} n_k$ is computed as

$$\bar{f}_{ik}^{vis,m} n_k = \begin{bmatrix} 0 \\ \tau_{1k} n_k \\ \tau_{2k} n_k \\ \tau_{3k} n_k \\ v_i \tau_{ik} n_k + q_k n_k \end{bmatrix}_m$$

Subscript m on the r.h.s. denotes that all primitive variables and their spatial derivatives should be computed at the m^{th} face centroid. Cells attached to the face are denoted as P and Q. Then, primitive variables are

$$\begin{aligned} \rho^m &= \frac{\rho^L + \rho^R}{2} \\ v_i^m &= \frac{\rho^L v_i^L + \rho^R v_i^R}{\rho^L + \rho^R} \\ p^m &= \frac{p^L + p^R}{2} \end{aligned} \tag{3.16}$$

The left (L) and right (R) states are given by eq. 3.9. The spatial derivative $\left.\frac{\partial V_i}{\partial x_k}\right|_m$ is computed by using an orthogonal correction scheme [141], which appropriately combines the already computed derivatives at P and Q centroids as described in subsection 3.2.5. It allows for the gradient's discretization for any angle formed between vector $\overrightarrow{PQ}$ and the face normal $\vec{n}$. Moreover, it smooths out any unphysical oscillations that may occur during the pseudo-time iteration process [78], [269]. Face and cell centroids are $\vec{x}^F$, $\vec{x}^P$ and $\vec{x}^Q$, respectively. The gradient is computed by the formula

$$\left.\frac{\partial V_i}{\partial x_k}\right|_m = \overline{\frac{\partial V_i}{\partial x_k}} - \left(\overline{\frac{\partial V_i}{\partial x_j}} \alpha_j - \left.\frac{\partial V_i}{\partial \alpha}\right|_m \right) \alpha_k \tag{3.17}$$

where

$$\vec{\alpha} = \frac{\vec{x}^Q - \vec{x}^P}{|\vec{x}^Q - \vec{x}^P|}$$

is a unit vector parallel to $\overrightarrow{PQ}$. Moreover, the mean derivative appearing in eq. 3.17 is

$$\overline{\frac{\partial V_i}{\partial x_k}} = \left.\frac{\partial V_i}{\partial x_k}\right|_P w + \left.\frac{\partial V_i}{\partial x_k}\right|_Q (1-w)$$

where

$$w = \frac{\left|\vec{x}^F - \vec{x}^Q\right|}{\left|\vec{x}^Q - \vec{x}^P\right|}$$

and the gradient along the $\vec{\alpha}$ direction is

$$\left.\frac{\partial V_i}{\partial \alpha}\right|_m = \frac{V_i^Q - V_i^P}{\left|\vec{x}^Q - \vec{x}^P\right|}$$

Eq. 3.17 is explained in detail in Appendix I. It is likely that, in a Cartesian mesh, cells P and Q are of the same size. Considering that $\vec{\alpha} = \vec{n}$ and $w = 0.5$ the aforementioned equations are simplified, which significantly reduces the gradient's computational cost. Regarding a boundary face, w is set to 1, which leads to

$$\overline{\frac{\partial V_i}{\partial x_k}} = \left.\frac{\partial V_i}{\partial x_k}\right|_P$$

Furthermore,

$$\left.\frac{\partial V_i}{\partial \alpha}\right|_m = \frac{V_i^{BC} - V_i^P}{\left|\vec{x}^F - \vec{x}^P\right|}$$

where V_i^{BC} computation is explained in subsection 3.2.6. Finally, for internal and boundary faces, the temperature gradient, needed for the heat flux computation, is

$$\left.\frac{\partial T}{\partial x_k}\right|_m = \left(\frac{1}{p^m}\left.\frac{\partial p}{\partial x_k}\right|_m - \frac{1}{\rho^m}\left.\frac{\partial \rho}{\partial x_k}\right|_m\right) T^m$$

where $T^m(\rho^m, p^m)$ is given by the equation of state presented in section 3.1.

3.2.8 Pseudo-Time Step Computation

The stability and convergence speed of the flow solver considerably depend on the pseudo-time step ($\Delta\tau$) choice [319]. One should choose the largest possible step size to accelerate the pseudo-time marching procedure and, consequently, reduce the total computational cost. However, stability analysis of 1D hyperbolic equations' explicit numerical solution scheme sets an upper bound on its value, eq. 3.3. Moreover, stability restriction for the 1D model diffusion equation $u_t = \nu u_{xx}$ suggests $\Delta\tau \leqslant 1/(2\nu\Delta x^2)$. Both criteria depend on local geometrical and flow quantities, which appear in a wide variety of scales along the flow field, and therefore, a local $\Delta\tau$ is adjusted at each finite volume [312], [249]. Due to the lack of similar theoretical analysis for the multi-dimensional Navier-Stokes equations, a combination of the two 1D restrictions is employed [148], though it does not ensure convergence. Specifically,

$$\Delta\tau = min(\Delta\tau_1, \Delta\tau_2, \Delta\tau_3) \tag{3.18}$$

where

$$\Delta\tau_i = CFL\frac{h_i}{|v_i| + c + T_i^{vis}}, \quad i = 1, \cdots, 3$$

and

$$T_i^{vis} = \frac{2\mu}{\rho h_i}$$

For inviscid flow applications T_i^{vis} is set to zero. In most CFD cases, the inviscid (Courant) stability restriction is stricter than the viscous limitation [148]. Variable h_i is the cell's height for each direction and CFL is the Courant-Friedrichs-Lewy number. Its value varies starting from a significantly small value and gradually increasing as the pseudo-time iterations (n) proceed,

$$CFL = \begin{cases} 10^{-3} + r^5(6-5r)CFL_{max}, & n < n_{max} \\ CFL_{max}, & n \geqslant n_{max} \end{cases}$$

where $r = n/n_{max}$. The user-defined variables CFL_{max} and n_{max} determine the first time step in which CFL takes its maximum value. The discretization presented in the previous subsections allows for $CFL_{max} > 1$.

3.3 Temporal Term Discretization of the Compressible Equations

Phenomena dealing with unsteady flows around moving geometries are of significant importance in real-world applications. However, they pose various challenges, such as the need for mesh deformation tools, which can inherently be met by IBMs. Despite their significant superiority in such cases over solvers using body-fitted meshes, they face difficulties retaining high simulation accuracy close to the moving solid bodies. Especially, the cut-cell method struggles to maintain conservation or provide physical solutions in large boundary displacements [219]. This section describes a discretization method developed within the current thesis which deals with these challenges. Firstly, the Arbitrary Lagrangian-Eulerian technique is briefly described, and flow equations are presented in a proper formulation. Thereafter, the dual time-stepping method is explained, and the corresponding algorithm for simulating unsteady flows is presented. Finally, treatments for the cells' abrupt appearance or disappearance within the flow domain, as well as the flow field projection from each mesh to the next one, are discussed in detail.

3.3.1 The Arbitrary Lagrangian Eulerian Technique

In flow simulations around moving bodies, the governing equations are integrated over a deforming finite volume provided with an arbitrary velocity distribution on its surface. Their discretization is based on the Arbitrary Lagrangian-Eulerian (ALE) technique [131] as a convenient way to face the Lagrangian motion of a body through an Eulerian flow field. The governing equations' integral form is

$$\frac{d}{dt}\int_{\Omega} U_i d\Omega - \int_{S} U_i v_k^g n_k dS + \int_{\Omega} \frac{\partial f_{ik}}{\partial x_k} d\Omega = 0 \tag{3.19}$$

where v_k^g is the solid body's surface velocity. When $\vec{v}^g = \vec{0}$, the equations' description is Eulerian, which allows for the simulation of a moving flow in a fixed computational mesh. On the contrary, when $\vec{v}^g$ is equal to the flow velocity, the equations' formulation turns into a Lagrangian one, meaning that mesh nodes follow the moving material particles. Therefore, ALE is a generalization of the two aforementioned flow field representations.

When a solid body performs an infinitesimally small displacement, all but mesh boundary faces stay unaffected, meaning that only boundary flux expressions are modified compared to the corresponding steady ones. Therefore, the steady equations' discretization method presented in section 3.2 remains almost intact. The following analysis aims to alleviate differences in moving wall fluxes by adjusting the equations' formulation. Without loss of generality, a finite volume is studied, constituted by three internal and one solid face, fig. 3.4. Although the three faces 1, 3, and w are modified due to the wall's motion, normal wall velocity $v_k^g n_k$ is non-zero only in face w, and therefore, it is the only face contributing to the surface integral of eq. 3.19,

$$\frac{d}{dt}\int_{\Omega} U_i d\Omega - U_i^w v_j^g n_k^w \Delta S^w + \sum_{m=1}^{3} f_{ik}^m n_k^m \Delta S^m + f_{ik}^w n_k^w \Delta S^w = 0 \tag{3.20}$$

The temporal term is discretized as

$$\frac{d}{dt}\int_{\Omega} U_i d\Omega = \frac{\Omega^{n+1}U_i^{n+1} - \Omega^{n+\frac{1}{2}}U_i^{n+\frac{1}{2}}}{\Delta t} \tag{3.21}$$

where Δt is the chosen physical time step. The computation of the conservative variables and volumes at the intermediate time step $(n+1/2)$ is discussed in subsection 3.3.3. Taking the no-penetration condition $v_i^w n_i^w = v_i^g n_i^w$ on a moving wall into account, the flux becomes

$$\vec{f}_k^w n_k^w = \begin{bmatrix} \rho^w v_i^g n_i^w \\ \rho^w v_1^w v_i^g n_i^w + p^w n_1^w - \tau_{1j}^w n_j^w \\ \rho^w v_2^w v_i^g n_i^w + p^w n_2^w - \tau_{2j}^w n_j^w \\ \rho^w v_3^w v_i^g n_i^w + p^w n_3^w - \tau_{3j}^w n_j^w \\ (\rho^w E^w + p^w) v_i^g n_i^w - v_i^g \tau_{ij}^w n_j^w \end{bmatrix} = \vec{U}^w v_i^g n_i^w + \underbrace{\begin{bmatrix} 0 \\ p^w n_1^w \\ p^w n_2^w \\ p^w n_3^w \\ p^w v_i^g n_i^w \end{bmatrix}}_{\vec{f}_k^{inv,w} n_k^w} - \underbrace{\begin{bmatrix} 0 \\ \tau_{1j}^w n_j^w \\ \tau_{2j}^w n_j^w \\ \tau_{3j}^w n_j^w \\ v_i^g \tau_{ij}^w n_j^w \end{bmatrix}}_{\vec{f}_k^{vis,w} n_k^w} \tag{3.22}$$

where the no-slip ($v_i^w = v_i^g$) and adiabatic wall ($q_k^w n_k = 0$) boundary conditions have also been taken into account in the viscous terms. Substituting the wall flux expression into eq. 3.20 one gets

$$\frac{\Omega^{n+1}U_i^{n+1} - \Omega^{n+\frac{1}{2}}U_i^{n+\frac{1}{2}}}{\Delta t} + \sum_{m=1}^{3} f_{ik}^m n_k^m \Delta S^m + f_{ik}^{inv,w} n_k^w \Delta S^w - f_{ik}^{vis,w} n_k^w \Delta S^w = 0 \tag{3.23}$$

Therefore, the term corresponding to the surface integral of eq. 3.19 vanishes and unsteady wall flux is almost the same compared to its steady variant, eq. 3.14. Their only difference appears in the energy equation, where $\vec{v}^g$ emerges, which should be computed so as to satisfy the Geometric Conservation Law (GCL) [176]. This ensures that no spurious mass, momentum or energy sinks appear in the flow domain. The GCL is a straight consequence of the Reynolds transport theorem [260], stated as

$$\frac{d}{dt}\int_{\Omega} d\Omega - \int_{S} v_k^g n_k dS = 0$$

A first-order forward-in-time finite difference scheme is used for its discretization

$$v_k^g n_k = \frac{\Omega^{n+1} - \Omega^{n+\frac{1}{2}}}{\Delta t \sum\limits_{\text{wall faces}} \Delta S^w}$$

The $\Omega^{n+\frac{1}{2}}$ computation does not guarantee that the resulting $v_k^g n_k$ matches the user-defined solid body's velocity $\hat{v}_k^g = \frac{dx_k^g}{dt}$. Numerical experiments have shown that the $\hat{v}_k^g$ choice over v_k^g facilitates convergence and results in a smoother flow field close to the moving boundaries, and thus, its use is preferred.

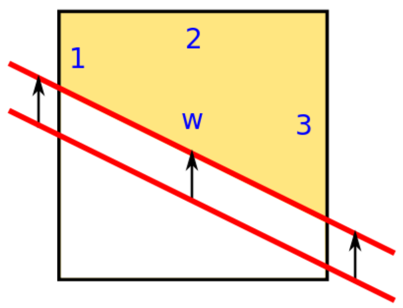


Figure 3.4: Cell intersected by a moving boundary wall (red line). Square's fluid part is colored. Vectors show the boundary's displacement direction. Blue numbers and letter 'w' (wall edge) name the four cut-cell edges.

3.3.2 Dual-Time Stepping

Moving boundaries flow applications have received an excessive amount of attention, leading to a substantial number of discretization methods in the recent literature. An implicit dual-time stepping method [202], firstly introduced by [179], is used for eq. 3.23 time evolution, allowing for an implicit temporal treatment that decouples the time step from the local mesh' scale, avoiding further stability restriction barriers. Consequently, time step choice is determined by the physics appropriate handling rather than numerical restrictions. By increasing the time step, the number of times the geometry and mesh intersection is detected over the whole unsteady simulation decreases, avoiding the frequent use of the most time-consuming part of the mesh generator. Moreover, the dual-time formulation supports the straightforward incorporation of the already developed steady-state flow solver infrastructure, discussed in section 3.2. The governing equations become

$$\frac{\partial U_i}{\partial t} + \frac{\partial U_i}{\partial \tau} + \frac{\partial f_{ik}}{\partial x_k} = 0$$

and their discretization is

$$\frac{\Omega^{n+1}U_i^{n+1,q+1} - \Omega^{n+\frac{1}{2}}U_i^{n+\frac{1}{2}}}{\Delta t} + \frac{\Omega^{n+1}U_i^{n+1,q+1} - \Omega^{n+1}U_i^{n+1,q}}{\Delta \tau} + \sum_{m=1}^{M} \left(f_{ik}^m n_k^m \Delta S^m\right)^{n+1,q+1} = 0$$

Recall that τ is referred to pseudo-time representing the internal iterative process, and t is the physical time. Contrary to section 3.2, indices n and q stand for the time and pseudo-time step counting, respectively. According to the developed algorithm, at each time-step, the geometry moves to its new position, the mesh is regenerated, and $\vec{U}^{n+\frac{1}{2}}$ are computed. Then, flow field $\vec{U}^{n+1}$ is initialized with $\vec{U}^{n+\frac{1}{2}}$ values, and a pseudo-time iteration process starts, until convergence is achieved, namely $|U_i^{n+1,q+1} - U_i^{n+1,q}| < eps$, where eps is a predefined small number. By the end of the current time step, the pseudo-time derivative term will vanish and, subsequently, the same procedure is repeated for the next time step.

3.3.3 Covered and Uncovered Cells Treatment

The main difficulty related to the flow simulation around moving boundaries using the cut-cell method and, generally, any IBM is the loss of mass, momentum, and energy conservation due to several cells' transition from solid to fluid and vice versa at each time step. Although this issue has already been discussed in section 2.8, where the cell-linking approach was introduced, this subsections deals with it from the perspective of the flow equations discretization. According to that, both transitions require delicate treatment leading to the $\vec{U}^{n+\frac{1}{2}}$ field computation used in the temporal term discretization presented in subsection 3.3.2.

Fig. 3.5 shows a fluid cell at time step n which is going to be covered by the solid region. Starting from fig. 3.5c, two finite volumes are depicted, the lower of which is going to disappear at the next time step. Indeed, the red boundary is lifted up totally covering the brown cell. The mesh detail after the boundary's motion at time step n+1 is presented in fig. 3.5a, where only the upper finite volume appears, deformed in shape. The cell's cover causes the disappearance of the conservative variables stored at its centroid. Within this context, an intermediate fake step (n+1/2) is defined, fig. 3.5b. The lower cell is combined with its upper neighbor, forming a merged hyper-cell. So, its contribution to the flow conservation variables will be transferred into the neighboring cell, which continues to exist at time step n+1. Flow variables at the intermediate step are computed ensuring conservation as

$$\begin{aligned} \Omega^{n+\frac{1}{2}} &= \sum_{m=1}^{M} \Omega^{mn} \\ U_i^{n+\frac{1}{2}} &= \frac{\sum\limits_{m=1}^{M} U_i^{mn} \Omega^{mn}}{\Omega^{n+\frac{1}{2}}} \end{aligned} \tag{3.24}$$

where M is the total number of incorporated cells forming a merged finite volume.

On the other hand, different complications arise when newborn cells emerge at a new time step. Their time integration by the governing equation is meaningless due to the absence of their conservative variables' time history. The case is also exemplified in fig. 3.5. Starting from fig. 3.5a at time step n, where only the upper cell belongs to the fluid domain, and following the red boundary lowering motion, a new cell appears in fig. 3.5c with no time history. Therefore, it is linked to its neighboring cell, yielding to the combined finite volume shown in fig. 3.5b. At time instant $n{+}1$, the cells are separated again and are treated independently by the flow solver. The volume and the conservative variables of each merged cell are updated as follows,

$$\begin{aligned} \Omega^{n+\frac{1}{2}} &= \sum_{m=1}^{M} \frac{1}{k^m} \Omega^{mn} \\ U_i^{n+\frac{1}{2}} &= \frac{\sum\limits_{m=1}^{M} U_i^{mn} \frac{1}{k^m} \Omega^{mn}}{\Omega^{n+\frac{1}{2}}} \end{aligned} \tag{3.25}$$

where M is the total number of cells combined to form a hyper-cell and k^m is the number of merged cells the m^{th} cell of the n^{th} time instant is part of. In the simple example illustrated in fig. 3.5, $M = 2$ and $k^m = 1$.

Consequently, spurious mass sources or sinks are avoided, and conservation is strictly maintained. Moreover, the developed method does not impose any restriction to the time step choice, allowing for large boundary displacements to occur covering or uncovering a great number of cells. Large intermediate merged cells close to the solid boundaries may reduce the simulation accuracy, but practice has shown that it does not affect the global accuracy of the discretization method [53], [68]. Section 2.8 defines the criteria used to form the merged cells at the $n{+}1/2$ time instant for appeared and disappeared cells.

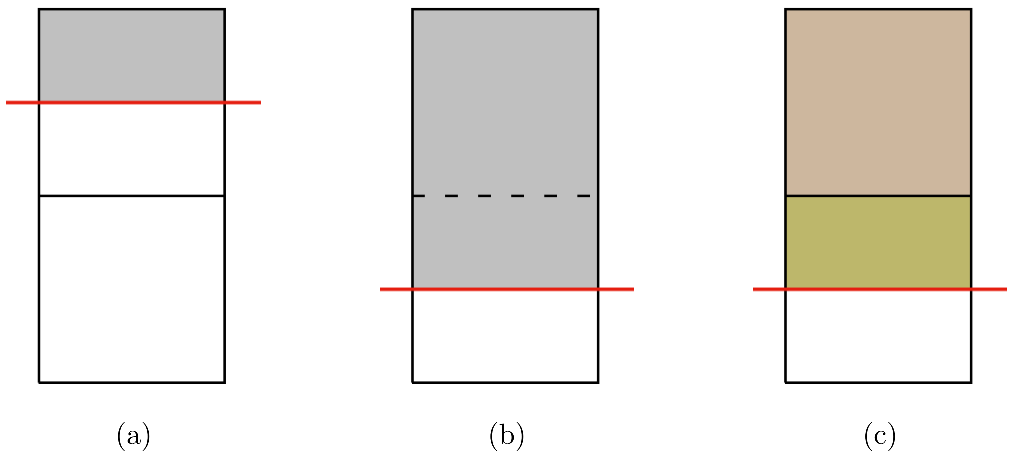


Figure 3.5: By arranging the figures from left to the right, a new cell appears to the fluid domain due to the red solid boundary's lowering motion. In the reversed order, a cell disappears from the fluid region. The squares' colored segment indicates the cells' fluid part.

Another factor affecting the intermediate step $n+1/2$ computation is mesh adaptation to the moving boundaries, which also affects cells far from the solid wall. Mesh is dynamically refined at each time step in the vicinity of the solid boundaries following their motion and maximizing the simulation's accuracy. According to subsection 2.8.1, this process embodies both coarsening and refinement tasks. Along with the mesh adaptation, interpolation should be employed to transfer the flow solution from the previous time step to the new one [273]. For the sake of clarity, two simple cases are demonstrated. Firstly, during the refinement process, the cell depicted in fig. 3.6a is subdivided into several cells of various sizes, fig. 3.6b. Flow variables $\vec{U}^{n+\frac{1}{2}}$ at the newly created cells are set equal to the initial cell's values $\vec{U}^n$. On the other hand, by reversing the figures' order, all cells, shown in fig. 3.6b, are merged to form a single hyper-cell, fig. 3.6a. The following formulas are used to attribute values to its centroid, ensuring the satisfaction of the conservation laws,

$$\begin{aligned} \Omega^{n+\frac{1}{2}} &= \Omega^{n+1} \\ U_i^{n+\frac{1}{2}} &= \frac{\sum\limits_{m=1}^{M} U_i^{mn} \Omega^{mn}}{\Omega^{n+\frac{1}{2}}} \end{aligned}$$

A matching process is required for both projections explained above, connecting cells from one mesh to the other. Such an algorithm is explained in subsection 2.8.1.

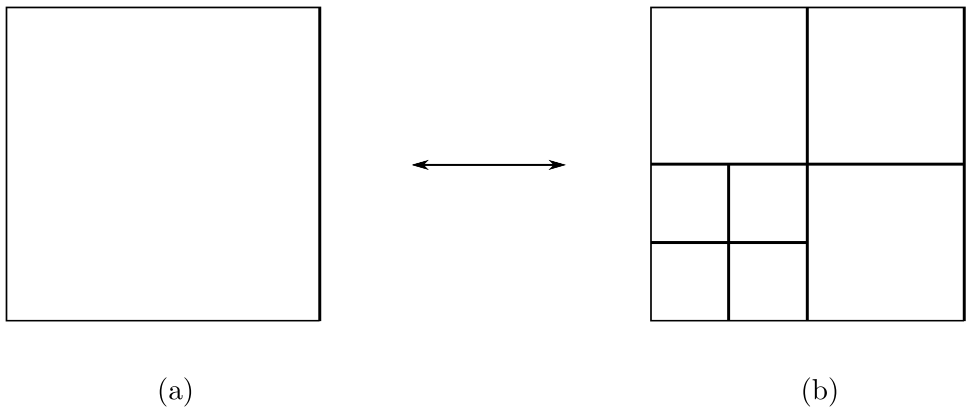


Figure 3.6: During the refining process, the cell on the left is subdivided into a number of smaller cells on the right. Inversely, on a coarsening process, neighboring cells on the right are merged to form the hyper-cell on the left.

3.4 Incompressible Fluid Flow Model

This section presents the equations governing incompressible flows. The governing equations are not a sub-case of the compressible equations presented in section 3.1 because the equation of state no longer holds. The density field is considered constant, causing the decoupling of the energy equation from the continuity and momentum PDEs. Heat transfer flow problems are beyond the thesis scope, and, therefore, the energy equation is not included in the mathematical model. The governing equations are

$$M_{ij}\frac{\partial V_j}{\partial t}+\frac{\partial f_{ik}^{inv}}{\partial x_k}-\frac{\partial f_{ik}^{vis}}{\partial x_k}=0, \quad i=1,\cdots,4,\ k=1,\cdots,3 \tag{3.26}$$

where

$$\vec{V}=\begin{bmatrix}p\\v_1\\v_2\\v_3\end{bmatrix},\quad \vec{f}_k^{inv}=\begin{bmatrix}v_k\\v_kv_1+\delta_{1k}p\\v_kv_2+\delta_{2k}p\\v_kv_3+\delta_{3k}p\end{bmatrix},\quad \vec{f}_k^{vis}=\begin{bmatrix}0\\\tau_{1k}\\\tau_{2k}\\\tau_{3k}\end{bmatrix},\quad M=\begin{bmatrix}0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

where p denotes the pressure divided by the constant fluid's density. The stress tensor τ is expressed as

$$\tau_{ik} = \nu \left(\frac{\partial v_i}{\partial x_k} + \frac{\partial v_k}{\partial x_i} \right)$$

with ν being the kinematic viscosity μ/ρ.

A set of boundary conditions completes the above system of PDEs. In external aerodynamics, boundaries are positioned far away from the examined geometry, and all flow variables' values are user-defined. The discretization scheme employed at the boundary is responsible for their selective implementation depending on the local flow conditions. On the other hand, in internal aerodynamics, a different set of flow quantities is imposed at the inlet and outlet. Three Dirichlet conditions are imposed at the inlet, two of which are the two angles identifying the velocity vector direction. The third quantity can either be the total pressure or the velocity magnitude. In the first case, velocity magnitude is extrapolated from the flow domain interior. The corresponding primitive variables are computed as

$$\begin{aligned}
p &= p_t - \frac{1}{2} v_i^2 \\
v_1 &= v_i^2 sin(\theta) cos(\phi) \\
v_2 &= v_i^2 sin(\theta) sin(\phi) \\
v_3 &= v_i^2 cos(\theta)
\end{aligned}$$

where p_t stands for the total pressure divided by density. The Bernoulli law is used for the total pressure definition. In the second set of boundary conditions, pressure is extrapolated from the flow domain. On the other hand, pressure is always imposed at the outlet, and the three velocity components are extrapolated from the flow domain.

Boundary conditions implied at the solid wall for inviscid or viscous flows are the same for incompressible and compressible flows, see section 3.1. Finally, boundary conditions on the symmetry plane are

$$\begin{aligned}
\frac{\partial p}{\partial n} &= 0 \\
v_i^{BC} &= v_i - 2 v_k n_k n_i
\end{aligned}$$

3.5 Discretization of the Steady Incompressible Laminar Equations

The study of eqs. 3.26 discretization starts by initially neglecting its viscous terms. Their numerical solution represents some significant difficulties since the corresponding system's Jacobian matrix does not provide real eigenvalues. Thus, the system of PDEs is not hyperbolic, and the techniques presented in section 3.2 are no more applicable for the incompressible equations. The artificial compressibility approach developed by Chorin [63] overcomes this difficulty by introducing an artificial density ($\hat{\rho}$) and the equation of state

$$p\rho = \beta^2 \hat{\rho}$$

where β is a positive real number called the artificial compressibility parameter and is assumed to be constant along the flow field. It is also reminded that p stands for the pressure divided by the constant fluid's density. By mimicking the compressible flow equations, eq. 3.1, the pseudo-temporal derivative of the density ($\frac{\partial \hat{\rho}}{\partial \tau}$) is added to the incompressible continuity equation. Thus, the steady variant of eqs. 3.26 becomes

$$\Gamma_{ij}^{-1} \frac{\partial V_j}{\partial \tau} + \frac{\partial f_{ik}^{inv}}{\partial x_k} = 0 \tag{3.27}$$

with Γ being the preconditioner matrix equal to

$$\Gamma = \begin{bmatrix} \beta^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The governing equations' multiplication with Γ modifies the inviscid flux as follows,

$$\vec{f}_k^{inv,\Gamma} = \Gamma \vec{f}_k^{inv} = \begin{bmatrix} \beta^2 v_k \\ v_1 v_k + \delta_{1k} p \\ v_2 v_k + \delta_{2k} p \\ v_3 v_k + \delta_{3k} p \end{bmatrix}$$

For further details on this method and its generalization, the reader is referred to [314], [315]. The flow equations' mathematical nature alteration leads to a hyperbolic system allowing for numerical techniques implementation, similar to those used for the compressible equations. The governing equations integration over a finite volume complies with the same analysis presented in 3.2.1. Moreover, Godunov discretization is also implemented in eps. 3.27. Therefore, a corresponding 1D Riemann problem should be solved to provide the appropriate flux expression [87]. The properly modified Roe's approximate Riemann solver gives

$$\bar{f}_{ik}^{inv,\Gamma,m} n_k^{PQ} = \frac{1}{2}\left(f_{ik}^{inv,\Gamma,L} + f_{ik}^{inv,\Gamma,R}\right) n_k^{PQ} - \frac{1}{2}\left|\tilde{A}_{ijk}^{\Gamma} n_k^{PQ}\right| \left(V_j^R - V_j^L\right) \tag{3.28}$$

where

$$A_{ijk}^{\Gamma} = \frac{\partial f_{ik}^{inv,\Gamma}}{\partial V_j} \tag{3.29}$$

The preconditioned Jacobian $A_k^{\Gamma} n_k$ is diagonalizable with real eigenvalues. Absolute Jacobian is defined as

$$|A_{ijk}^{\Gamma} n_k| = P_{il}^{\Gamma} |\Lambda_{lm}^{\Gamma}| P_{mj}^{\Gamma,-1} \tag{3.30}$$

The P^{Γ}, Λ^{Γ} and $P^{\Gamma,-1}$ expressions can be found in [283] and Appendix D. Matrix $|\tilde{A}_k^{\Gamma} n_k|$ is computed by using Roe averages, which are set to the mean quantities between the L and R states. It is shown in [302] that this algebraic average satisfies the Roe conditions,

$$\tilde{V}_i = \frac{V_i^L + V_i^R}{2} \tag{3.31}$$

Proof of eq. 3.28 can be found in Appendix F.

According to the previous discussion, the absolute Jacobian matrix is a function of the artificial compressibility parameter and, consequently, part of the flux discretization scheme. Thus, its value affects the final flow solution. Contrary to the compressible equations, the influence of the pseudo-temporal term added to the governing equations does not vanish after convergence is achieved. Therefore, the simulation's accuracy is based on the β value, which may be problematic in practical applications. Research in this area has not yet concluded with the appropriate β computation. Its value choice is a fact of experience and trial and error process.

Moreover, Jacobian matrix' eigenvalues also depend on parameter β. Considering that stability criteria are based on their values, one concludes that the artificial compressibility parameter plays a significant role in the numerical instabilities reduction and the convergence rate acceleration. Based on the theory presented in subsection 3.2.2, the optimal choice of β in terms of stability is the one that minimizes the largest possible ratio of wave speeds generated between two neighboring finite volumes [313]. After some algebra, shown in Appendix C, it is proven that the optimal artificial compressibility parameter is given by

$$\beta^2 = 3v_i^2 \tag{3.32}$$

This formula suggests that optimal β is locally adjusted depending on each cell's velocity magnitude rather than remaining constant over the flow domain. However, this tactic leads to instabilities when very low-velocity magnitude regions exist and is avoided throughout this thesis. Nevertheless, the above formula can still indicate an appropriate global β value based on a characteristic flow field velocity.

A second-order discretization method is possible by applying the MUSCL method of subsection 3.2.3. The extrapolation scheme of eq. 3.9 is also valid for incompressible flows, where limiter expressions are given by eq. 3.10 or 3.11 and the required gradient of $\vec{V}$ is computed by eq. 3.13.

In case of viscous incompressible flows, the viscous fluxes, eq. 3.26, should be multiplied by the preconditioning matrix Γ. However, the viscous flux remains intact because $\Gamma \vec{f}_k^{vis} = \vec{f}_k^{vis}$, and thus, the exact discretization method presented in subsection 3.2.7 can be applied.

Finally, the pseudo-time step $\Delta\tau$ is computed by eq 3.18. The required sound speed is considered infinite for incompressible flows due to the constant density assumption. Artificial sound speed is used instead [87], which is defined as

$$c = \sqrt{v_i^2 + \beta^2} \tag{3.33}$$

3.6 Temporal Term Discretization of the Incompressible Equations

Discretization of the unsteady incompressible equations around moving boundaries presents similarities with the method developed for compressible flows in section 3.3. The ALE integral form of the governing equations, eq. 3.19, is also applied in this case, where the vector $\vec{U}$ is defined as

$$\vec{U} = \begin{bmatrix} 0 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

The equations' discretization is exemplified by using the simple case depicted in fig. 3.4, which leads to eq. 3.20. Temporal term is discretized by using eq. 3.21. The corresponding flux on a moving wall displays some differences with respect to the compressible case. It is expressed as

$$\vec{f}_k^w n_k^w = \begin{bmatrix} v_i^g n_i^w \\ v_1^w v_i^g n_i^w + p^w n_1^w - \tau_{1j}^w n_j^w \\ v_2^w v_i^g n_i^w + p^w n_2^w - \tau_{2j}^w n_j^w \\ v_3^w v_i^g n_i^w + p^w n_3^w - \tau_{3j}^w n_j^w \end{bmatrix} = \vec{U}^w v_i^g n_i^w + \begin{bmatrix} v_i^g n_i^w \\ 0 \\ 0 \\ 0 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ p^w n_1^w \\ p^w n_2^w \\ p^w n_3^w \end{bmatrix}}_{\vec{f}_k^{inv,w} n_k^w} - \underbrace{\begin{bmatrix} 0 \\ \tau_{1j}^w n_j^w \\ \tau_{2j}^w n_j^w \\ \tau_{3j}^w n_j^w \end{bmatrix}}_{\vec{f}_k^{vis,w} n_k^w}$$

Thus, the ALE formulation discrete form is

$$\begin{bmatrix} 0 \\ \frac{\Omega^{n+1} v_1^{n+1} - \Omega^{n+\frac{1}{2}} v_1^{n+\frac{1}{2}}}{\Delta t} \\ \frac{\Omega^{n+1} v_2^{n+1} - \Omega^{n+\frac{1}{2}} v_2^{n+\frac{1}{2}}}{\Delta t} \\ \frac{\Omega^{n+1} v_3^{n+1} - \Omega^{n+\frac{1}{2}} v_3^{n+\frac{1}{2}}}{\Delta t} \end{bmatrix} + \begin{bmatrix} v_i^g n_i^w \\ 0 \\ 0 \\ 0 \end{bmatrix} + \sum_{m=1}^{3} f_{ik}^m n_k^m \Delta S^m + f_{ik}^{inv,w} n_k^w \Delta S^w - f_{ik}^{vis,w} n_k^w \Delta S^w = 0$$

Normal velocity $v_i^g n_i^w$ can be substituted by using the discrete GCL

$$\frac{\Omega^{n+1} - \Omega^{n+\frac{1}{2}}}{\Delta t} - v_i^g n_i^w = 0$$

Then, the governing equations become

$$\frac{\Omega^{n+1} \hat{U}_i^{n+1} - \Omega^{n+\frac{1}{2}} \hat{U}_i^{n+\frac{1}{2}}}{\Delta t} + \sum_{m=1}^{3} f_{ik}^m n_k^m \Delta S^m + f_{ik}^{inv,w} n_k^w \Delta S^w - f_{ik}^{vis,w} n_k^w \Delta S^w = 0$$

with

$$\vec{\hat{U}} = \begin{bmatrix} 1 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Fluxes $\vec{f}_k^{inv,w} n_k^w$ and $\vec{f}_k^{vis,w} n_k^w$ are the same as those of a steady flow case. Moreover, $\vec{v}^g$ has been vanished from the equations' discrete expression, and there is no need for its computation.

Dual time-stepping, discussed in subsection 3.3.2, is used for the equations' propagation in time. The pseudo-time derivative multiplied with the preconditioner matrix is added to the unsteady incompressible equations,

$$\begin{gathered} \frac{\partial U_i}{\partial t} + \Gamma_{ij}^{-1} \frac{\partial V_j}{\partial \tau} + \frac{\partial f_{ik}}{\partial x_k} = 0 \Leftrightarrow \\ \Gamma_{ij} \frac{\partial U_j}{\partial t} + \frac{\partial V_i}{\partial \tau} + \Gamma_{ij} \frac{\partial f_{jk}}{\partial x_k} = 0 \end{gathered}$$

The preconditioned equations' discrete form is

$$\frac{\Omega^{n+1} \hat{U}_i^{\Gamma,n+1,q+1} - \Omega^{n+\frac{1}{2}} \hat{U}_i^{\Gamma,n+\frac{1}{2}}}{\Delta t} + \frac{\Omega^{n+1} V_i^{n+1,q+1} - \Omega^{n+1} V_i^{n+1,q}}{\Delta \tau} + \sum_{m=1}^{M} \left(f_{ik}^{\Gamma,m} n_k^m \Delta S^m \right)^{n+1,q+1} = 0 \tag{3.34}$$

where

$$\vec{\hat{U}}^{\Gamma} = \begin{bmatrix} \beta^2 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Flux $f_{ik}^{\Gamma,m} n_k^m$ computation is presented in section 3.5. The algorithm implemented for simulating unsteady flows is described in subsection 3.3.2. Geometrical and flow variables computation at the intermediate time step $(n + 1/2)$ is explained in subsection 3.3.3.

3.7 Numerical Solution of the Discretized Flow Equations

In sections 3.2, 3.3, 3.5 and 3.6, the numerical discretization of the compressible and incompressible flow equations was studied, which leads to a non-linear algebraic system of $K \times L$ equations, where K is the number of mesh cells, and L the number of PDEs modeling the flow phenomenon. This section presents a point-implicit method to solve this numerical system. Vector $\vec{R}_i \in \mathbb{R}^L$ represents the sub-system corresponding to residuals of the i^{th} finite volume and $\vec{W}_i$ is the vector of its unknown variables. It is equal to $\vec{U}$ for compressible flows (eq. 3.1) and $\vec{V}$ for incompressible flows (eq. 3.26). At each time step n, the Newton-Raphson algorithm [248] performs the system linearization. For an arbitrary cell P, it is expressed as

$$\frac{\widehat{\partial R_i^{P,q+1}}}{\partial W_j^{k,q+1}} \Delta W_j^{k,q+1} = -R_i^{P,q+1}, \quad i = 1, \cdots, L, \quad k, P = 1, \cdots, K$$

where $\Delta W_j^{k,q+1} = W_j^{k,q+1} - W_j^{k,q}$, with q denoting the current pseudo-time step. The residual at each cell is not only a function of its flow variables, but also of those corresponding to other cells around its region. Index k stands for the summation of these dependencies. For simplicity reasons and for decreasing the computational memory requirements, it is assumed that $\vec{R}^P$ depends only on $\vec{W}^P$ and $\vec{W}^{Q_m}$, where Q_m is the m^{th} first neighbor of cell P. Therefore, the system expressed in matrix form is

$$D^P \overrightarrow{\Delta W^P} + \sum_{m=1}^{M} OD^{m,P} \overrightarrow{\Delta W}^{Q_m} = -\vec{R}^P \tag{3.35}$$

with

$$\begin{aligned} D_{ij}^P &= \frac{\widehat{\partial R_i^P}}{\partial W_j^P} \\ OD_{ij}^{m,P} &= \frac{\widehat{\partial R_i^P}}{\partial W_j^{Q_m}} \quad i, j = 1, \cdots, L \end{aligned}$$

standing for the diagonal and off-diagonal matrix elements. The $q+1$ superscript has been neglected for the sake of brevity. The hat symbol indicates that the residual derivative is approximately estimated. Its exact computation is avoided reducing numerical operations, which may lead to complicated and costly algorithms. However, the computed derivative should be accurate enough to drive the system into convergence. It is essential to clarify that this approximation does not sacrifice the flow simulation's accuracy. The computation of D and OD matrices is presented below.

By differentiating eq. 3.23 or 3.34 one gets

$$
\begin{aligned}
D_{ij} =& \frac{\Omega^{n+1}}{\Delta t} + \frac{\Omega^{n+1}}{\Delta \tau} + \sum_{m=1}^{M} \left(\frac{\partial(\widehat{f_{ik}^{inv,m} n_k^m})}{\partial W_j^P} - \frac{\partial(\widehat{f_{ik}^{vis,m} n_k^m})}{\partial W_j^P} \right) \\
OD_{ij}^m =& \frac{\partial(\widehat{f_{ik}^{inv,m} n_k^m})}{\partial W_j^{Q_m}} - \frac{\partial(\widehat{f_{ik}^{vis,m} n_k^m})}{\partial W_j^{Q_m}}
\end{aligned}
$$

Repeated index m does not imply summation. Superscript P is omitted in terms in which neither P nor Q index is mentioned. The simplified inviscid flux derivatives for internal mesh faces are computed by differentiating eq. 3.4/3.5 or 3.28,

$$
\begin{aligned}
\frac{\partial(\widehat{f_{ik}^{inv,m} n_k^m})}{\partial W_j^P} =& \frac{1}{2} A_{ijk}^P n_k^{PQ} - \frac{1}{2} |\tilde{A}_{ijk} n_k^{PQ}| \\
\frac{\partial(\widehat{f_{ik}^{inv,m} n_k^m})}{\partial W_j^{Q_m}} =& \frac{1}{2} A_{ijk}^Q n_k^{PQ} - \frac{1}{2} |\tilde{A}_{ijk} n_k^{PQ}|
\end{aligned}
$$

The absolute Jacobian matrix derivative multiplied by the flow variables vector is assumed to be zero. In the case of a boundary face, flux contribution to the off-diagonal matrix is considered zero. The compressible inviscid flux derivative on the wall is

$$
\frac{\partial(\widehat{\vec{f}_k^{inv,w} n_k^w})}{\partial W_j^P} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
\frac{\partial p}{\partial U_1} n_1^w & \frac{\partial p}{\partial U_2} n_1^w & \frac{\partial p}{\partial U_3} n_1^w & \frac{\partial p}{\partial U_4} n_1^w & \frac{\partial p}{\partial U_5} n_1^w \\
\frac{\partial p}{\partial U_1} n_2^w & \frac{\partial p}{\partial U_2} n_2^w & \frac{\partial p}{\partial U_3} n_2^w & \frac{\partial p}{\partial U_4} n_2^w & \frac{\partial p}{\partial U_5} n_2^w \\
\frac{\partial p}{\partial U_1} n_3^w & \frac{\partial p}{\partial U_2} n_3^w & \frac{\partial p}{\partial U_3} n_3^w & \frac{\partial p}{\partial U_4} n_3^w & \frac{\partial p}{\partial U_5} n_3^w \\
\frac{\partial p}{\partial U_1} v_n & \frac{\partial p}{\partial U_2} v_n & \frac{\partial p}{\partial U_3} v_n & \frac{\partial p}{\partial U_4} v_n & \frac{\partial p}{\partial U_5} v_n
\end{bmatrix}
$$

where $v_n = v_k^g n_k^w$ and

$$
\frac{\partial p}{\partial \vec{U}} = (\gamma - 1) \begin{bmatrix} \frac{1}{2} v_i^2 \\ -v_1 \\ -v_2 \\ -v_3 \\ 1 \end{bmatrix}
$$

All the above flow quantities are computed by extrapolating their values from the cell's centroid to each face. The incompressible inviscid flux derivative on the wall is

$$\frac{\partial(\widehat{\vec{f}_k^{inv,w} n_k^w})}{\partial W_j^P} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ n_1^w & 0 & 0 & 0 \\ n_2^w & 0 & 0 & 0 \\ n_3^w & 0 & 0 & 0 \end{bmatrix}$$

The viscous part of the equations is similarly treated. Differentiation of the compressible viscous flux is based on the following assumptions. Firstly, a simplified orthogonal correction formula is used, instead of eq. 3.17,

$$\left.\frac{\partial \Phi}{\partial x_k}\right|_m \simeq \left.\frac{\partial \Phi}{\partial \alpha}\right|_m \alpha_k$$

where Φ is a velocity component or temperature. Another simplification is made by linearizing the energy equation's viscous terms. Therefore,

$$\frac{\partial(\widehat{\vec{f}_k^{vis,m} n_k^m})}{\partial W_j^P} = \begin{bmatrix} 0 & 0 & 0 & 0 & \\ \widehat{\frac{\partial \tau_{n_1}}{\partial U_1}} & \widehat{\frac{\partial \tau_{n_1}}{\partial U_2}} & \widehat{\frac{\partial \tau_{n_1}}{\partial U_3}} & \widehat{\frac{\partial \tau_{n_1}}{\partial U_4}} & \widehat{\frac{\partial \tau_{n_1}}{\partial U_5}} \\ \widehat{\frac{\partial \tau_{n_2}}{\partial U_1}} & \widehat{\frac{\partial \tau_{n_2}}{\partial U_2}} & \widehat{\frac{\partial \tau_{n_2}}{\partial U_3}} & \widehat{\frac{\partial \tau_{n_2}}{\partial U_4}} & \widehat{\frac{\partial \tau_{n_2}}{\partial U_5}} \\ \widehat{\frac{\partial \tau_{n_3}}{\partial U_1}} & \widehat{\frac{\partial \tau_{n_3}}{\partial U_2}} & \widehat{\frac{\partial \tau_{n_3}}{\partial U_3}} & \widehat{\frac{\partial \tau_{n_3}}{\partial U_4}} & \widehat{\frac{\partial \tau_{n_3}}{\partial U_5}} \\ v_i \widehat{\frac{\partial \tau_{n_i}}{\partial U_1}} + \widehat{\frac{\partial q_n}{\partial U_1}} & v_i \widehat{\frac{\partial \tau_{n_i}}{\partial U_2}} + \widehat{\frac{\partial q_n}{\partial U_2}} & v_i \widehat{\frac{\partial \tau_{n_i}}{\partial U_3}} + \widehat{\frac{\partial q_n}{\partial U_3}} & v_i \widehat{\frac{\partial \tau_{n_i}}{\partial U_4}} + \widehat{\frac{\partial q_n}{\partial U_4}} & v_i \widehat{\frac{\partial \tau_{n_i}}{\partial U_5}} + \widehat{\frac{\partial q_n}{\partial U_5}} \end{bmatrix}$$

with

$$\begin{aligned} \widehat{\frac{\partial \tau_{n_i}}{\partial U_j}} &= -\mu \left(\frac{\partial v_i}{\partial U_j} \alpha_k n_k + \frac{\partial v_k}{\partial U_j} \alpha_i n_k - \frac{2}{3} \frac{\partial v_k}{\partial U_j} \alpha_k n_i \right) \\ \widehat{\frac{\partial q_n}{\partial U_j}} &= -k \frac{\partial T}{\partial U_j} a_k n_k \end{aligned}$$

and

$$\frac{\partial \vec{v}}{\partial \vec{U}} = \frac{1}{\rho} \begin{bmatrix} -v_1 & 1 & 0 & 0 & 0 \\ -v_2 & 0 & 1 & 0 & 0 \\ -v_3 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad \frac{\partial T}{\partial \vec{U}} = \frac{1}{c_v \rho} \begin{bmatrix} -E \\ -v_1 \\ -v_2 \\ -v_3 \\ 1 \end{bmatrix}$$

where $c_v = \frac{R}{\gamma - 1}$ is the heat capacity at constant volume. All flow quantities needed in the above expressions are computed on the face as explained in subsection 3.2.7. Regarding the incompressible case, a similar strategy is followed. The derivatives of the stress tensor projected in the normal direction are

$$\widehat{\frac{\partial \tau_{n_i}}{\partial U_j}} = -\mu \left(\alpha_i n_j + \alpha_k n_k \delta_{i+1,j}\right)$$

The same formulas are valid also for boundary fluxes for both compressible and incompressible flows. Finally, the viscous off-diagonal term is

$$OD_{ij}^m(\vec{W}^{Q_m}) = -D_{ij}(\vec{W}^{Q_m})$$

The system of eqs. 3.35 for every cell P is iteratively solved by using the Symmetric Gauss-Seidel (SGS) method [266]. The process terminates after a user-defined number of iterations is reached. Finally, the steps followed for simulating an unsteady flow are

1. Initialize flow field.
2. Loop over mesh faces to compute contributions to $\vec{R}$, D, and OD from both neighboring cells (scatter-add technique [186]).
3. Solve system, eq. 3.35, by applying the SGS iterative method, until the predefined iterations' number is reached.
4. Use system's solution $(\Delta\vec{W})$ to recompute flow solution.
5. If convergence criterion is met, go to step 7.
6. Start a new pseudo-time step by continuing with step 2.
7. Start a new time step by initializing the past and current flow field with the solution achieved in step 3.
8. If number of time steps is reached, terminate. Otherwise go to step 2.

3.8 The Ghost-Cell Method

The cut-cell method for solving compressible and incompressible flows has been discussed in detail in sections 3.1 to 3.6. Its Cartesian mesh treatment distinguishes the method from the rest of IBMs. The cut-cells' arbitrary shape results in a complicated data structure and flow solver. On the other hand, the introduction of the ghost-cell method, as a subclass of the IBMs, eases these difficulties keeping the benefits of the automated mesh generation a Cartesian mesh offers. The intersection between the immersed geometry and the mesh is not detected, and wall boundary conditions are indirectly implemented, allowing to treat complicated moving geometries in a straightforward manner. Its conceptual simplicity has attracted the scientists' interest resulting in a high number of publications, which leads to the development of numerous variations of the method and its application to a considerable amount of industrial cases.

In this thesis, a combination of the ghost-cell method developed in [74] and the ghost-fluid method introduced in [92] is programmed to simulate compressible inviscid flows [215]. According to this, both the fluid and solid parts of the mesh are used. The governing equations are discretized according to section 3.2 and solved throughout the entire Cartesian mesh. The corresponding finite volumes coincide with the cubic Cartesian cells, even for cells intersected by the solid wall, without performing any further geometrical modification. Therefore, the only mesh boundary is its outer box surface. Consequently, special treatment is needed in the vicinity of the solid wall to implement the no-penetration condition, which is described in detail in subsection 3.8.1.

3.8.1 Wall Boundary Conditions Implementation

The present approach firstly defines a ghost layer of predefined length equal to $3\sqrt[3]{\Delta x \Delta y \Delta z}$ inside the solid body, where Δx, Δy and Δz are the local cells' dimensions. If a cell's centroid is placed inside this layer, it is called a "ghost-cell". The method's main purpose is to mirror the flow field image through the boundary on the ghost layer, defining each ghost-cell's flow variables. This process is exemplified in fig. 3.7 for a 2D case. It shows a mesh detail intersected by the solid wall depicted by a straight red line. The ghost layer is colored in gray. Two nodes G and G' are also shown. Point G' is the image of the ghost cell's centroid G through the boundary. Let $\vec{v}^{G'}$ be the velocity vector corresponding to G'. Then, the velocity at the ghost-cell's centroid is defined as

$$v_i^G = v_i^{G'} - 2v_j^{G'} n_j n_i \tag{3.36}$$

where $\vec{n}$ is the unit vector normal to the boundary. Their mean velocity computed at the GG' segment's midpoint satisfies the no-penetration condition $\vec{v}^w \cdot \vec{n} = 0$. Many interpolation methods have been developed for the $\vec{v}^{G'}$ computation. Most of them express the velocity as a function of the neighboring cells' velocity and their distance from point G'.

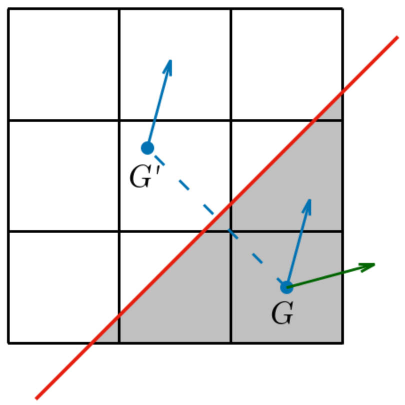


Figure 3.7: Ghost-cell G is placed inside the ghost layer, colored in gray. It's image through the red wall is noted as G'. Velocity vector $\vec{v}^{G'}$, presented in blue, is copied to cell G. Its normal to the wall component is reversed forming the green vector.

In order to avoid such complicated geometrical structures, the following method is preferred. Firstly, the Signed Distance Function (SDF) Φ is computed at each cell's centroid. It is defined as the shortest distance between the centroid and the wall's surface. Its sign is positive for centroids placed inside the solid region and negative for the rest. The $\Phi = 0$ iso-surface represents the sold wall. A direct outcome of the SDF definition is that the Φ gradient on the wall is equal to the unit normal vector pointing from the fluid to the solid region. This property is used to define the normal vector for each cell P as

$$n_i^P = \left. \frac{\partial \Phi}{\partial x_i} \right|_P$$

Another notable property is that Φ satisfies the eikonal equation

$$\left(\frac{\partial \Phi}{\partial x_i}\right)^2 = 1$$

Various methods have been proposed for the SDF computation. Some of them are based on the eikonal PDE numerical solution [280], [281], [345] and others on geometrical computations [75]. The developed method belongs to the second category. Initially, cells intersected by the geometry are detected, forming a front. Their distance from the surface is accurately computed. The closest surface point coordinates are also stored for each cell. Subsequently, a second front is defined, which consists of cells neighboring the first front. Each newly examined cell computes its distance from the surface points stored in its first front neighbors and stores their minimum as well as the corresponding surface point. The front propagation continues until all cells are processed. This procedure may introduce inaccuracies, which are canceled by repeating the algorithm several times until no further corrections are possible.

The second step of the wall boundary condition implementation is to transfer the primitive variables from the flow domain ($\Phi < 0$) to the ghost layer ($\Phi > 0$) along the normal direction. This process is made by solving the

$$\frac{\partial V_i}{\partial n} = 0$$

PDE in the ghost layer for each primitive variable. A more useful formulation is

$$\frac{\partial V_i}{\partial n} = 0 \Leftrightarrow \frac{\partial V_i}{\partial x_j} n_j = 0 \Leftrightarrow \frac{\partial V_i}{\partial x_j}\frac{\partial \Phi}{\partial x_j} = 0 \Leftrightarrow \frac{\partial V_i}{\partial \tau} + \frac{\partial \Phi}{\partial x_j}\frac{\partial V_i}{\partial x_j} = 0$$

where a pseudo-time derivative has been added. This equation is hyperbolic, and an upwind scheme is applied for its discretization. It is solved by repetitively accessing each ghost-cell and correcting its primitive variables by

$$V_i^{n+1} = V_i^n - \Delta\tau \left.\frac{\partial \Phi}{\partial x_j}\frac{\partial V_i}{\partial x_j}\right|_n \tag{3.37}$$

where n is the iterations' counter. Φ gradient is computed by the Least Square Method explained in subsection 3.2.5. V_i gradient is discretized by using a backward, in case $\frac{\partial \Phi}{\partial x_i} > 0$, or forward, in case $\frac{\partial \Phi}{\partial x_i} < 0$, first-order finite differences scheme. For example, the V_i derivative w.r.t. the x_1 Cartesian coordinate is

$$\frac{\partial V_i^P}{\partial x_1} = \begin{cases} \frac{V_i^P - \bar{V}_i^{Q_L}}{\Delta x_1}, & \left.\frac{\partial \Phi}{\partial x_1}\right|_P > 0 \\ \frac{\bar{V}_i^{Q_R} - V_i^P}{\Delta x_1}, & \left.\frac{\partial \Phi}{\partial x_1}\right|_P < 0 \end{cases}$$

where $\bar{V}_i^{Q_L}$ and $\bar{V}_i^{Q_R}$ are the left and right numerical averages between cell's P neighboring values. Fig. 3.8 presents such a case, in which $\bar{V}_i^{Q_R} = (V_i^{Q_1} + V_i^{Q_1})/2$.

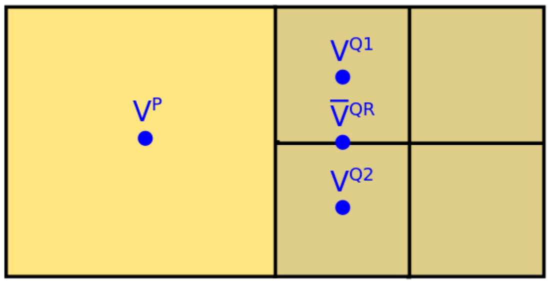


Figure 3.8: Cell P borders on two neighboring cells Q_1 and Q_2 on its r.h.s. The mean value of their primitive variables equals to $\bar{V}_i^{Q_R}$, which is used for the eq. 3.37 discretization.

The iterative process's stability is controlled by the pseudo-time step, which is

$$\Delta\tau = min(\frac{\Delta x_1}{n_1}, \frac{\Delta x_2}{n_2}, \frac{\Delta x_3}{n_3})$$

The last step is the reversal of the ghost-cells velocity's normal component. Eq. 3.36 is transformed to

$$v_i^{G,new} = v_i^G - 2v_j^G n_j n_i \tag{3.38}$$

Therefore, the velocity at each ghost-cell is paired with the mirrored flow velocity across the immersed boundary canceling its normal component at the wall.

The described method does not take the flow conservation laws into consideration, and thus, mass, momentum, and energy leakage from the flow field to the solid's interior is unavoidable. Fig. 3.9 demonstrates this argument by comparing the way the ghost-cell and the cut-cell methods treat the solid boundary. It's evident that the use of mesh squares as finite volumes allows for the flow to enter the solid region violating the flow conservation laws. Mesh adaptation close to the fluid-solid intersection increases the extrapolation's accuracy sharpening the solid-fluid interface and alleviating the flow penetration into the solid bodies. Therefore, a denser mesh is usually used compared to the cut-cell method.

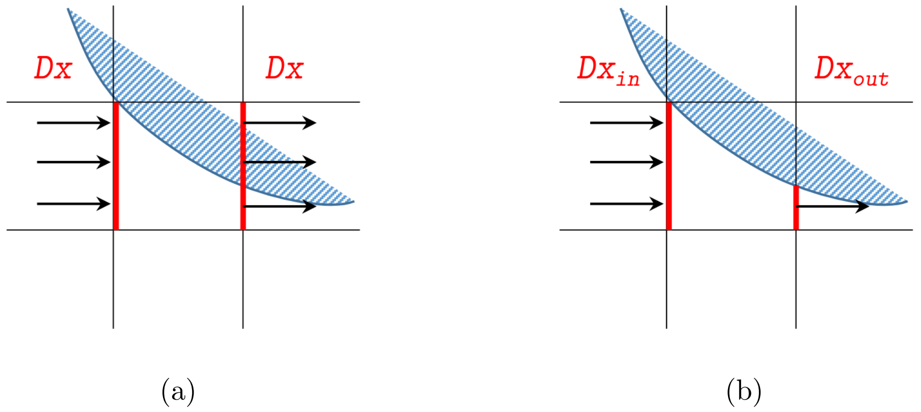
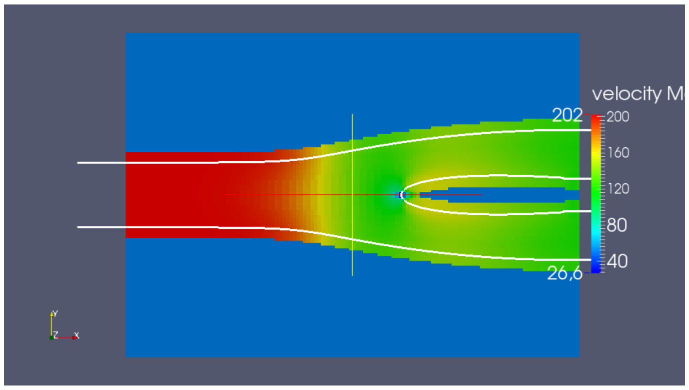


Figure 3.9: The ghost-cell method (a) does not detect the mesh-geometry intersection allowing small flow portions to escape from the fluid domain, violating the conservation laws. On the contrary, the cut-cell method (b) prevents the flow from entering into the solid region (blue) by discarding the cut-cells' solid part.

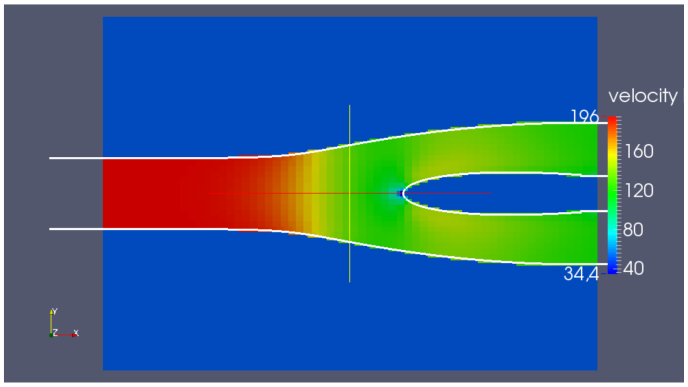
A comparison in terms of accuracy is made between the ghost-cell and the cut-cell method in the following application. A duct with one inlet and two outlets is used. Total pressure (1 *bar*) and total temperature (293 K) are imposed at the inlet and static pressure (0.88 *bar*) at both outlets. Velocity magnitude contours are shown in fig. 3.10. The white line stands for the duct's wall and the blue region indicates the mesh part in which the governing equations were not solved. When the ghost-cell method is used, fig. 3.10a, the velocity field is successfully mirrored into the ghost layer, indirectly imposing the no-penetration boundary condition. On the other hand, when the cut-cell method is implemented, fig. 3.10b, the fluid region's and duct's boundaries coincide. Mass loss percentage deviation is measured for three different mesh sizes for both methods. Results are summarized in table 3.1. As expected, the cut-cell method successfully satisfies the mass conservation. Contrarily, the error is high enough in the ghost-cell method and decreases by the mesh size increase. Moreover, mesh independence is achieved in much larger meshes resulting in a higher computational cost.

Mesh size (K)	Ghost-Cell (%)	Cut-Cell (%)
5	-	0.091
10	2.34	0.018
20	2.23	0.018

Table 3.1: The one-inlet-two-outlets duct: Mass conservation percentage deviation between inlet and outlets. Comparison between the ghost-cell and the cut-cell method for three different mesh sizes.



(a)



(b)

Figure 3.10: The one-inlet-two-outlets duct: Iso-velocity magnitude contours. The white line indicates the duct's wall. Flow simulation is implemented by (a) the ghost-cell and (b) the cut-cell method. Flow field presence in the solid region in (a) corresponds to the ghost layer.

3.8.2 The Unsteady Ghost-Cell Method Implemented in Moving Walls

In unsteady cases, the geometry motion causes changes in the SDF. At each time step, Φ should be modified, increasing the simulation's computational cost. However, its recomputation through the algorithm, presented in subsection 3.8.1, is usually avoided and other methods are used instead. For example, in turbomachinery applications, Φ field is rotated following the rotor blades' motion surpassing the algorithm's repetitive implementation. Since mesh boundaries remain intact, grid velocity is zero, and the corresponding surface integral of ALE formulation, eq. 3.19, vanishes. Wall velocity ($\vec{v}^{w}$) is induced to the discretized system through the ghost-cells' flow variables. Mirroring implemented in steady cases by eq. 3.38 is modified becoming

$$v_i^{G,new} = v_i^G + 2(v_j^w - v_j^G)n_j n_i \tag{3.39}$$

The temporal term is discretized by backward first-order finite differences. Contrary to the cut-cell method, cells' shape remains intact during the flow simulation. Thus, geometries motion does not cause cells appearance or disappearance. Flow variables stored at each finite volume continuously change throughout the unsteady phenomenon, allowing for a straightforward temporal term discretization. Its time step should be small enough to guarantee that only ghost-cells appear in the fluid region, preventing the entrance of other solid cells. Thus, a large number of time iterations is unavoidable, and a fully explicit discretization method is preferable due to its low computational cost at each time step. In such a case, no pseudo-time iterations are performed, and the flow variables are computed as

$$U_i^{n+1} = U_i^n - \Delta t \sum_{m=1}^{M} (f_{ik}^m n_k^m \Delta S^m)^n \tag{3.40}$$

avoiding the formation and solution of a linear system. Section 9.5 implements the described method to predict the flow around a compressor rotor.

Consequently, the ghost-cell method is advantageous compared to other CFD methods in cases concerning flows around complex geometries and perplexed geometry motions. Moreover, it can easily handle cases in which the initial topology changes. Furthermore, it is an easily implemented method. However, its accuracy is lower than the cut-cell method, but can be increased to some extent by using a much more dense mesh close to the fluid-solid interface.

Chapter 4

Flow Solver Assessment

This chapter presents a detailed validation/verification of the developed cut-cell flow solver, in numerous compressible and incompressible cases selected from the literature. External and internal aerodynamics, with inviscid and viscous (laminar) flows, as well as flows around stationary or moving bodies are considered. The applications were chosen based on the presence of analytical flow solutions or experimental or other reliable CFD software results. Firstly, applications for inviscid flows are considered, where the accuracy of the proposed discretization and the imposition of accurate boundary conditions along the solid wall will be demonstrated. Then, laminar flow cases are developed, and the way the Cartesian mesh irregularities on the boundary affect the layer's development. Finally, the method's ability to correctly solve the flow equations around moving solid bodies while satisfying the conservation laws, is investigated. The purpose of this chapter is to demonstrate the cut-cell software ability to produce highly accurate results, equivalent to that obtained by using body-fitted meshes, maintaining all Cartesian mesh advantages.

4.1 Compressible Flow Solver Assessment

In this section, the validation/verification of the compressible flow solver is presented. Regarding inviscid flow simulations, the external aerodynamics around a NACA0012 airfoil, a wedge, the ONERA M6 wing, and the flow within a converging-diverging duct in transonic flow conditions are presented. Concerning laminar flow simulations, velocity and temperature boundary layer profiles are reproduced in a flat plate case and the NACA0012 airfoil. The cut-cell software results are compared with analytical solutions and/or the outcome of other CFD codes should those be available.

4.1.1 Inviscid Flow Over the NACA0012 Isolated Airfoil

This case concerns the 2D inviscid transonic flow over an isolated airfoil. The accuracy of the cut-cell code is assessed through comparisons with computational results published in the AGARD No 211 report [97], where the Euler equations were solved by using a body-fitted O-type structured mesh of 20K nodes. The studied geometry is the symmetric NACA0012 airfoil with modified/closed trailing edge. Its upper surface is defined by

$$\bar{y}(x) = 5t(0.2969\sqrt{\bar{x}} - 0.126\bar{x} - 0.3516\bar{x}^2 + 0.2843\bar{x}^3 - 0.1015\bar{x}^4)$$

where $t = 0.12$ is the thickness parameter, $x = \bar{x}/x_0$, $y = \bar{y}/x_0$ and $x_0 = 1.008930411365$, $x \in [0, 1]$. The far-field flow conditions are $M_\infty = 0.85$ and $\alpha_\infty = 1°$. Mesh adaptation was used to increase the simulation's accuracy by refining the mesh six times, every 600 pseudo-time iteration steps. The final mesh of 80K cells and a close-up of the pressure side's shock region are shown in figs. 4.1a and 4.1b, respectively. The wall-clock time of this computation is 35 min. using 48 processors. The computed iso-Mach lines and contours are shown in fig. 4.2, where the two shock waves can be seen in both airfoil sides. Fig. 4.4 shows the convergence results, where the residual overshootings indicate the iteration, at which the mesh adaptation occurs. Comparison of the Mach number distribution over the airfoil's surface and the pressure coefficient with results given by [97] shows good agreement, fig. 4.3. The shock waves have been detected in the same position, although no adaptation has been used by [97]. Consequently, the corresponding pressure jumps computed by the cut-cell software are steeper leading to a small disagreement in the lift and drag coefficients, as shown in table 4.1.

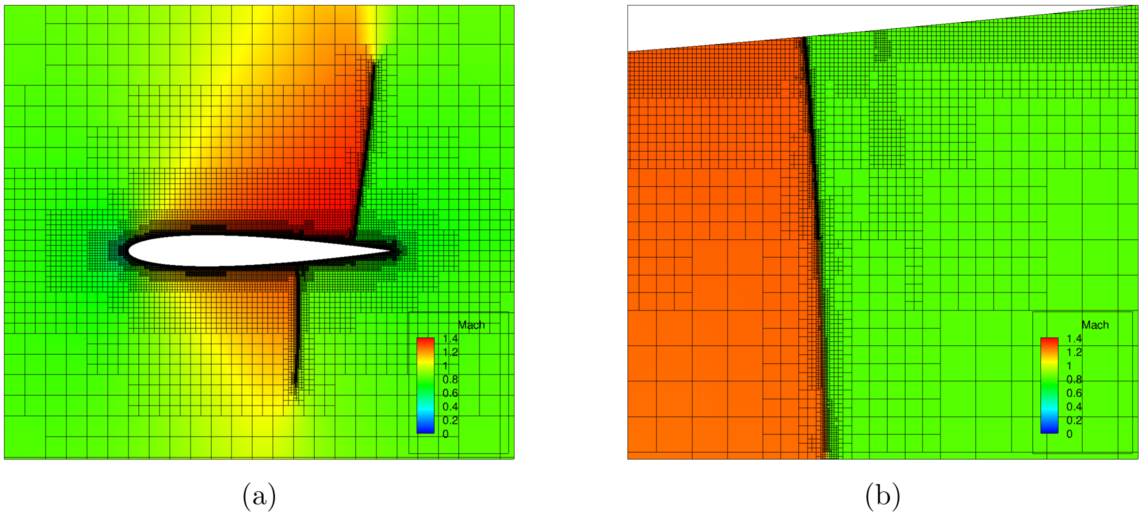


Figure 4.1: Inviscid flow of a compressible fluid over the NACA0012 isolated airfoil: (a) Mesh adaptation in the vicinity of the pressure and suction sides' shock waves. (b) Mesh adaptation detail around the suction side's shock wave. Thanks to mesh adaptation, shock waves of infinitesimally low thickness have been computed.

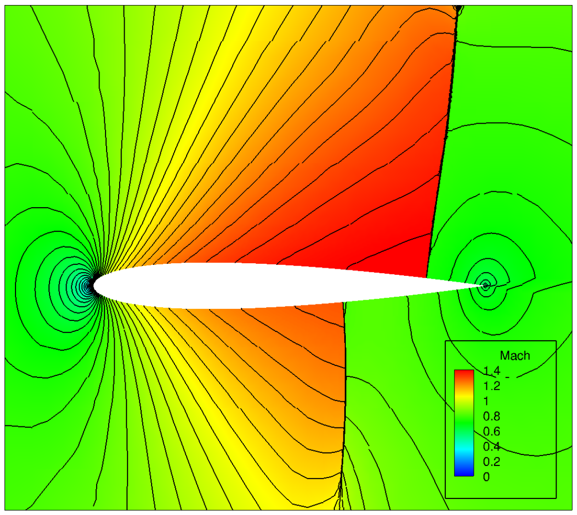


Figure 4.2: Inviscid flow of a compressible fluid over the NACA0012 isolated airfoil: Computed Mach number iso-lines.

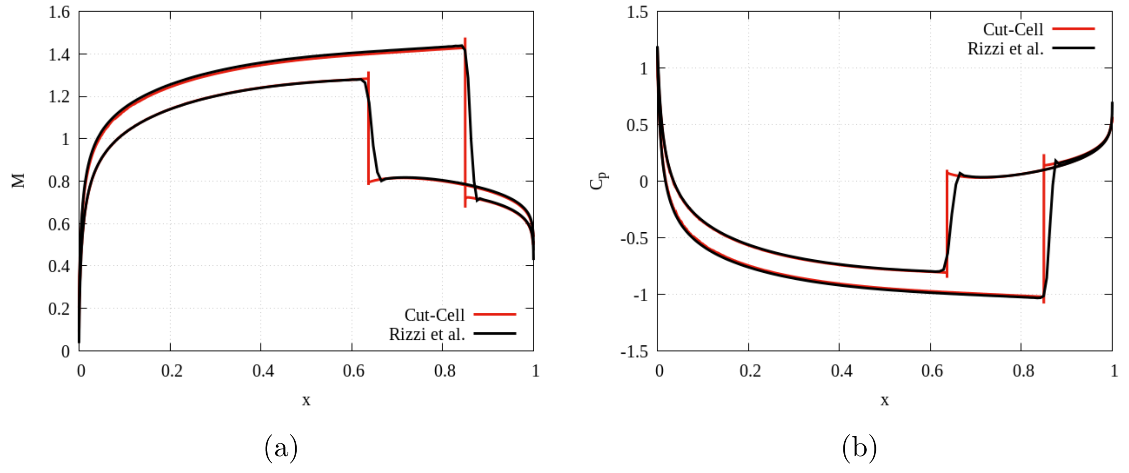


Figure 4.3: Inviscid flow of a compressible fluid over the NACA0012 isolated airfoil: (a) Mach number and (b) pressure coefficient distributions over the airfoil surface. Comparison between the cut-cell results and the CFD results of [97] computed using a body-fitted mesh.

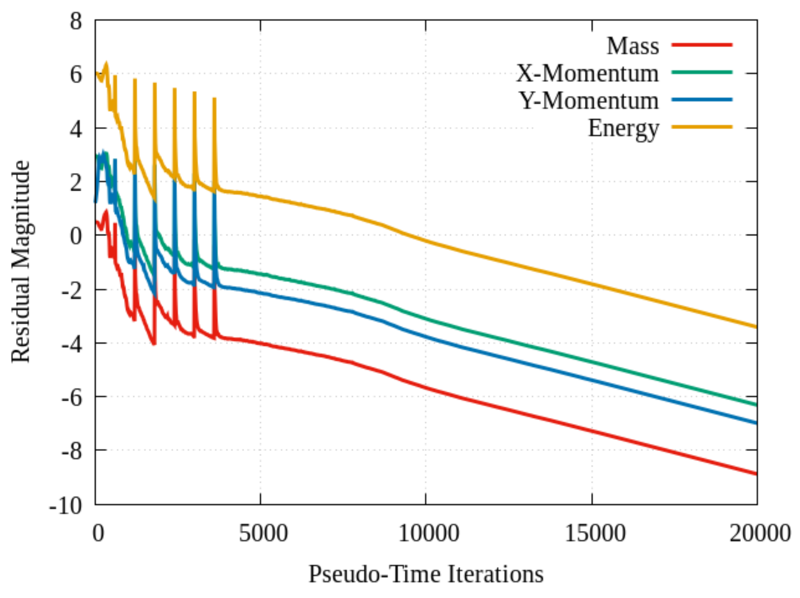


Figure 4.4: Inviscid flow of a compressible fluid over the NACA0012 isolated airfoil: Convergence history. Residual overshootings indicate iterations at which mesh adaptation occurs.

	C_l	C_d
Cut-Cell	0.335	0.0554
Reference	0.330	0.0528
Deviation (%)	1.52	4.92

Table 4.1: Inviscid flow of a compressible fluid over the NACA0012 isolated airfoil: Lift and drag coefficients computed by the cut-cell software and results given by [97].

4.1.2 Inviscid Flow Over a Wedge

The purpose of this application is to test the cut-cell method's ability to accurately simulate a discontinuity in the worst-case scenario, which happens when a 45° oblique shock wave occurs in a square flow domain, meaning that the wave's direction is parallel to all mesh' cells diagonals. Therefore, a wedge is placed into a supersonic compressible flow. For a given corner angle θ (wedge' half angle), an oblique shock wave of angle β is created, as shown in fig. 4.5. The upstream horizontal streamlines are uniformly deflected after the shock wave, changing the flow direction and parallelizing it to the wedge surface. The flow remains uniform before and after the shock wave, while a discontinuous change occurs along the shock surface. According to the inviscid gas dynamics theory, the $\theta-\beta-M$ equation [181]

$$tan\theta = 2cot\beta \frac{M_1^2 sin^2\beta - 1}{M_1^2[\gamma + cos(2\beta)] + 2}$$

expresses the relation between the wedge and shock angles and the upstream Mach number. Mach numbers before and after the shock are related using the equation

$$M_2^2 sin^2(\beta - \theta) = \frac{1 + \frac{\gamma-1}{2} M_1^2 sin^2\beta}{\gamma M_1^2 sin^2\beta - \frac{\gamma-1}{2}}$$

where $\gamma = 1.4$ is the heat capacity ratio [181]. The Mach number upstream the wedge is set equal to 2. According to the $\theta-\beta-M$ equation, a 45° oblique shock wave is created by setting $\theta \simeq 14.73°$.

The symmetric Mach number field and the adapted mesh are shown in figs. 4.6a and 4.6b respectively. The shock capturing algorithm successfully detected the discontinuity by refining the mesh in a small region around the shock. Despite the difficulty of capturing a 45° shock wave by using a Cartesian mesh, the cut-cell software accurately solved the flow equations producing two regions of constant flow variables and a sharp discontinuity between them. The Mach number distribution along the horizontal line $y = 0.4m$ and the corresponding analytical solution are plotted in fig. 4.7. The comparison between them shows the high accuracy of the proposed method.

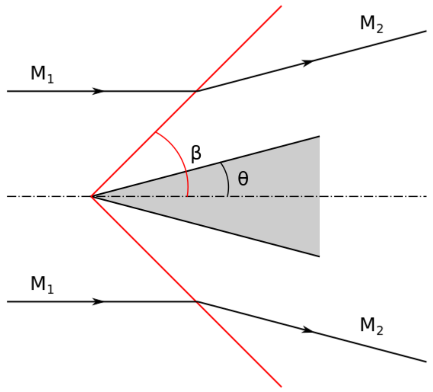
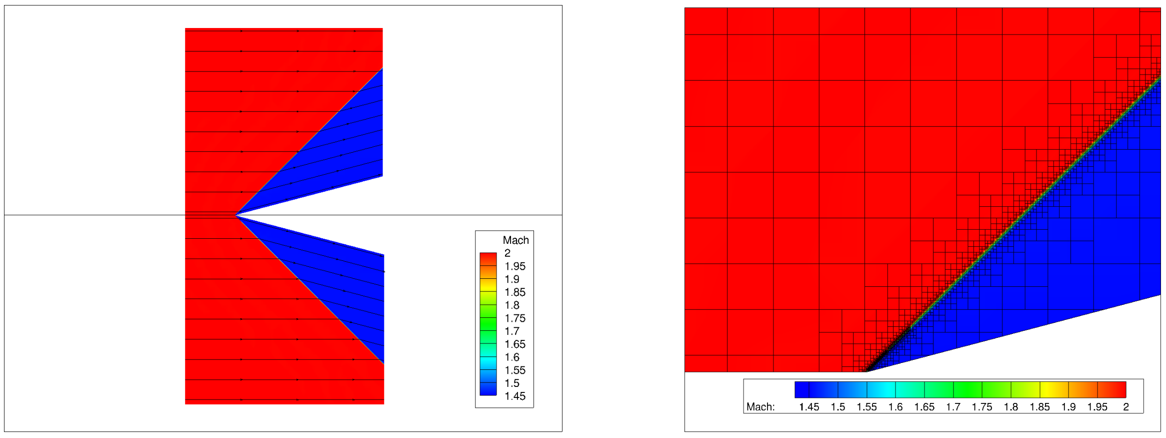


Figure 4.5: Inviscid flow of a compressible fluid over a wedge: Theoretical solution of the oblique shock (red line) created by the horizontal supersonic flow over the wedge. Two streamlines (in black) denote the velocity vector change before and after the shock wave.



(a) (b)

Figure 4.6: Inviscid flow of a compressible flow over a wedge: (a) Mach number field and velocity streamlines computed by the cut-cell software and (b) adaptive mesh refinement over the oblique shock wave.

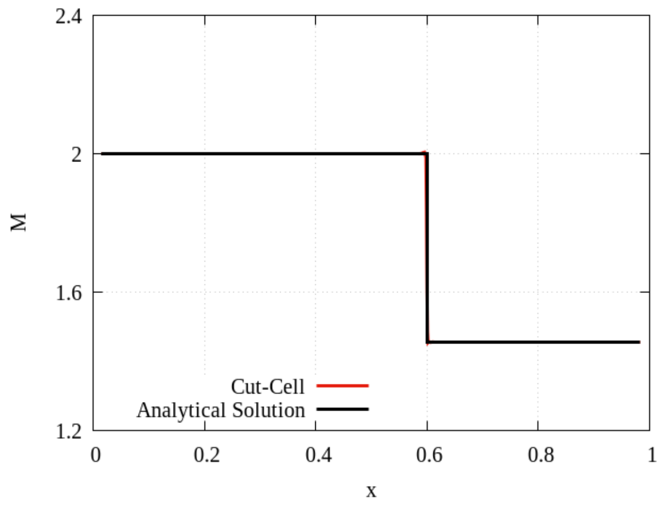


Figure 4.7: Inviscid flow of a compressible fluid over a wedge: Mach number distribution along line $y = 0.4\ m$. Comparison between the analytical (black) and cut-cell (red) solution.

4.1.3 Convergent-Divergent Duct Flow

The flow inside a convergent-divergent duct with adiabatic walls is the next verification case. It is appropriate to study mass, momentum, and energy conversation throughout the duct, being a crucial issue in most IBMs. Assuming that the duct area variation is moderate, the perpendicular to the duct axis velocity components are small compared to the parallel ones. According to that, the flow is considered constant across any cross-section, and an analytical quasi-1D flow solution can be obtained. More details about the analytical solution can be found in [14]. A comparison between the analytical solution and the numerical results along the duct's axis of symmetry gives essential evidence about the cut-cell software accuracy. The cross-section distribution in the axial direction is

$$A(x) = \begin{cases} 1.75 - 0.75cos[(0.2x - 1)\pi], & 0 \leq x < 5 \\ 1.25 - 0.25cos[(0.2x - 1)\pi], & 5 \leq x \leq 10 \end{cases}$$

Total pressure and temperature are the imposed boundary conditions at the inlet ($p_t = 1\ bar$ and $T_t = 290\ K$). The static outlet pressure is set equal to 0.86 bar, causing the appearance of a shock wave in the duct's divergent part. The computed by the cut-cell method Mach number flow field is shown in fig. 4.8. The computation accuracy is increased by using mesh adaptation over the normal shock, as depicted in fig. 4.9. It is computationally verified that the flow and mesh adaptation are fully axisymmetric. Mach number and static pressure distributions along the duct's centerline are compared with the quasi-1D analytical solution. The comparison presented in fig. 4.10 shows the cut-cell software's ability to predict the exact shock wave position and compute a highly accurate flow field. The mass and energy flux difference between the duct inlet and outlet is significantly small, as shown in table 4.2, indicating the cut-cell method's ability to satisfy flow conservation. Moreover, table 4.3 compares the inlet mass and energy flux and the exerted axial force on the duct between numerical and analytical solutions, confirming the high accuracy of the programmed cut-cell software.

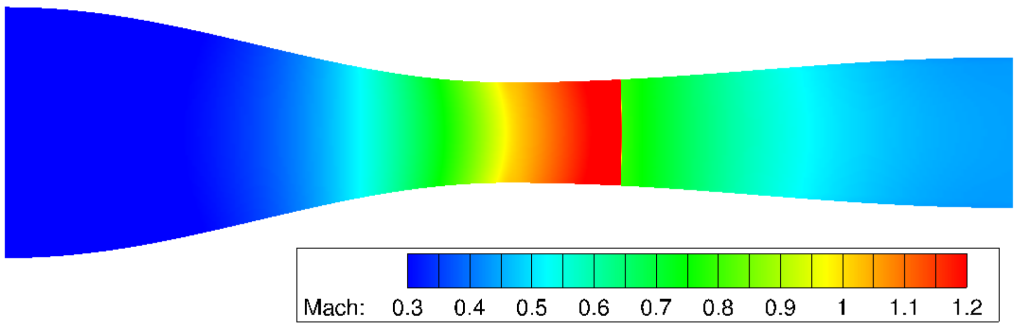


Figure 4.8: Inviscid flow of a compressible fluid in a duct: Mach number field and velocity streamlines. A normal shock wave is formed in its divergent part.

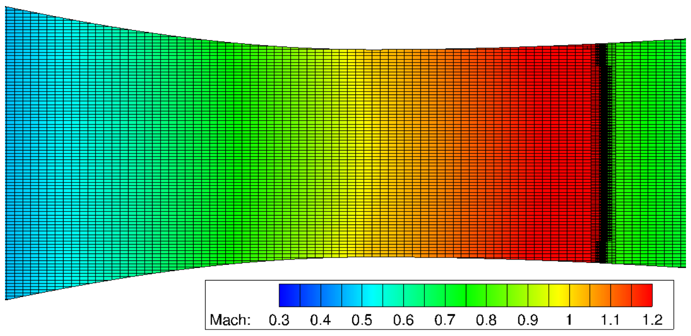
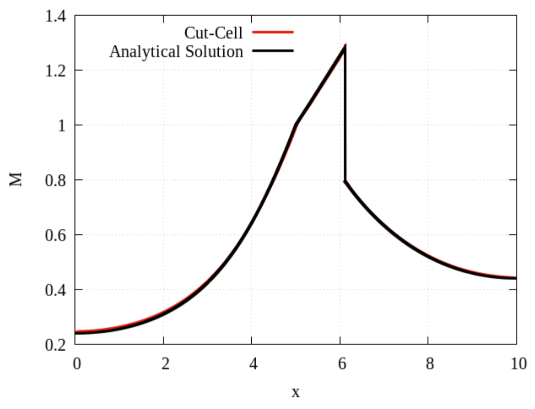
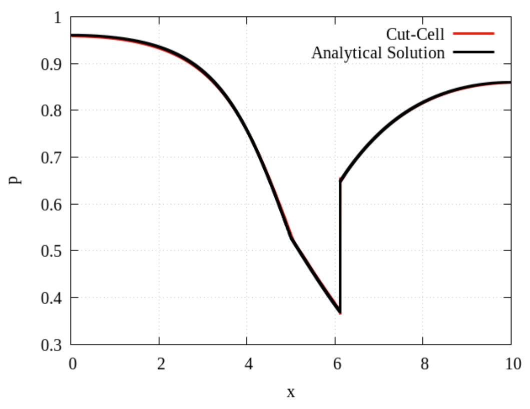


Figure 4.9: Inviscid flow of a compressible fluid in a duct: Mach number field iso-areas. Mesh adaptation close to the normal shock.



(a)



(b)

Figure 4.10: Inviscid flow of a compressible fluid in a duct: (a) Mach number and (b) pressure distributions. Comparison between the cut-cell solution along the centerline (red) and pseudo-1D analytical solution (black).

	Mass (Kg/s)	Energy (MJ/s)
Inlet	237.27	69.133
Outlet	237.29	69.138
Deviation (%)	0.0067	0.0072

Table 4.2: Inviscid flow of a compressible fluid in a duct: Duct's inlet and outlet mass and energy flux.

	Mass (Kg/s)	Force (kN)	Energy (MJ/s)
Cut-Cell	237.27	95.527	69.133
Reference	237.32	95.52	69.147
Deviation (%)	0.0013	0.0082	0.013

Table 4.3: Inviscid flow of a compressible fluid in a duct: Cut-cell and analytical results regarding the inlet mass and energy flux and the exerted force on the duct. Deviation between the aforementioned values is also shown.

4.1.4 Parallel Flow Over a Flat Plate

The following subsections aim the validation/verification of the software for the simulation of viscous (laminar) flows. The next application focuses on the mesh' Cartesian structure effect on the boundary layer development. Most of the time, the Cartesian mesh lines are far from orthogonal to the geometry surface. This irregularity affects the velocity spatial derivative accuracy close to the solid wall and harms the accurate computation of viscous fluxes. In particular, the skin friction appears quite noisy in most IBMs [36]. However, the proposed cut-cell method overcomes these difficulties, confirmed by the following flat plate boundary layer study placing the plate parallel and inclined by 15° to the mesh lines. These cases correspond practically to the same phenomenon, and their study investigates how the mesh orientation at the plate's surface affects the simulation accuracy. In both cases, the flow domain is 1.25 m long and 1.0 m high, enforcing no-slip wall boundary conditions only after 20% of its length. Total pressure and temperature are imposed at the inlet and static pressure at the outlet. The characteristic dimensionless numbers at the far-field are $M_\infty = 0.5$, $Re_\infty = 850$, $Pr_\infty = 0.72$ and the far-field angle-of-attack is always parallel to the plate.

The cut-cell software results are compared with the Blasius theory solution extended for compressible flows [61]. According to that, the velocity and total enthalpy fields inside the boundary layer are

$$\begin{aligned} u(x,y) &= u_\infty f'(\eta) \\ H(x,y) &= H_\infty g(\eta) \end{aligned}$$

where x, y are the parallel and vertical directions to the plate and η is defined as

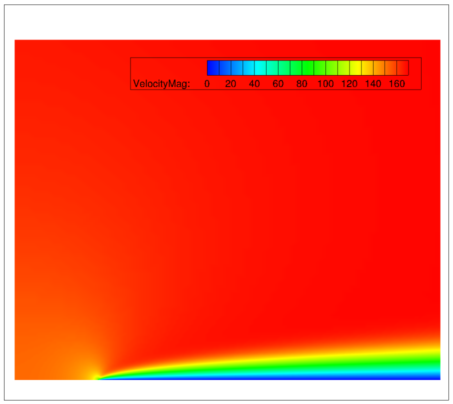
$$\eta = \sqrt{\frac{u_\infty}{2\rho_\infty \mu_\infty x}} \int_0^y \rho dy$$

The f and g functions are the solutions of the following o.d.e.'s.

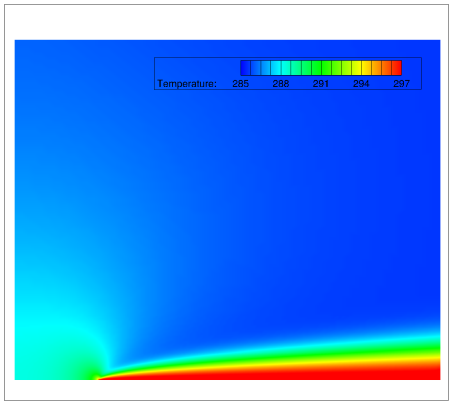
$$\begin{aligned} & f''' + ff'' = 0 \\ & g'' + Pr_\infty f g' = \bar{\sigma}(1 - Pr_\infty)\left(f'f''\right)', \ \bar{\sigma} = \frac{(\gamma-1)M_\infty^2}{1 + \frac{\gamma-1}{2} M_\infty^2} \end{aligned}$$

The boundary conditions are $f(0)=0$, $f'(0)=0$, $f'(\infty)=1$, $g'(0)=0$, $g(\infty)=1$. The aforementioned analytical solution requires adiabatic wall boundary conditions and the use of a linear relation between dynamic viscosity and temperature, expressed as $\mu/\mu_\infty = T/T_\infty$. Thermal conductivity should follow the same rule ($k/k_\infty = T/T_\infty$).

Figs. 4.11 and 4.12 show the velocity and temperature boundary layers for the horizontal and the inclined flat plate respectively. A mesh detail close to the beginning fo the boundary layer is shown in fig. 4.13. The mesh irregularity does not prevent the formulation of a smooth velocity field, especially in the inclined plate where non-uniform fully unstructured cut-cells are generated. This argument is further established by comparing the velocity and temperature profiles of numerical and analytical solutions taken at 90% of the flow domain length (figs. 4.14, 4.15). Even in the inclined plate case, the computed profiles remain smooth and very close to the analytical ones, meaning that the non-uniform cut-cells do not sacrifice the flow simulation's accuracy. Furthermore, skin friction coefficient is well predicted along the plate's surface and compares nicely with the analytical solution, fig. 4.16. A mesh sensitivity analysis is presented in the same plot, where the mesh refinement leads to more accurate results as the red line becomes the blue one, being much closer to the straight analytical line. Moreover, in the inclined case, the results are compared with an other cut-cell software [36], in which a quadratic reconstruction in the wall-normal direction is used near the walls to mitigate mesh irregularity. The governing equations were discretized using the finite volume approach and the HLLC Riemann solver. The comparison shows good agreement confirming the proposed method's high accuracy.

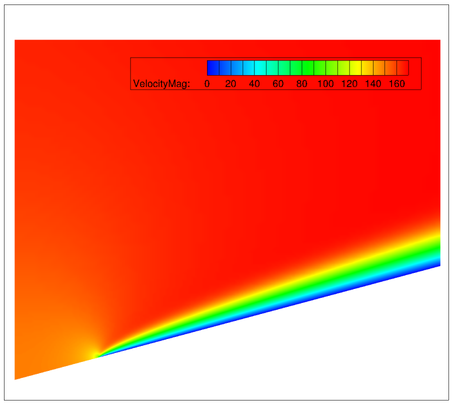


(a)

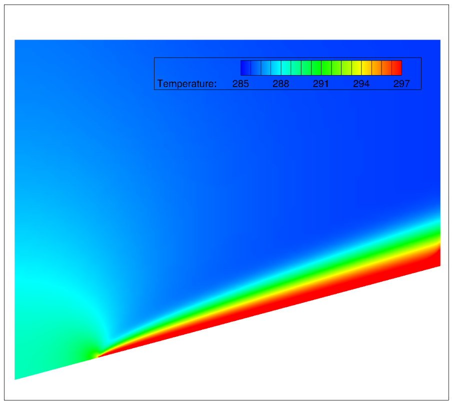


(b)

Figure 4.11: Laminar flow of a compressible fluid over flat plate: (a) Velocity and (b) temperature boundary layers over a horizontal plate.

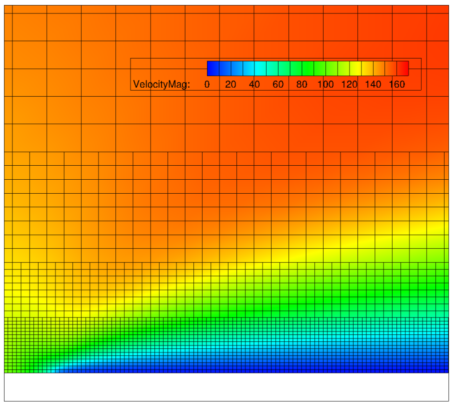


(a)

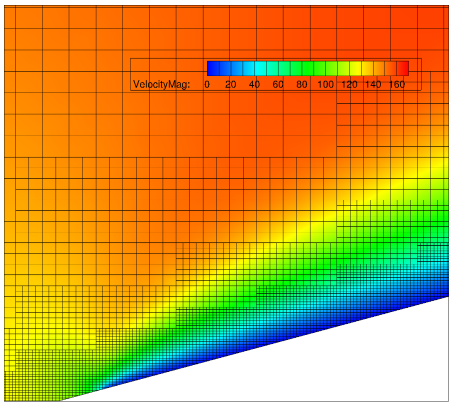


(b)

Figure 4.12: Laminar flow of a compressible fluid over flat plate: (a) Velocity and (b) temperature boundary layers over an inclined plate.

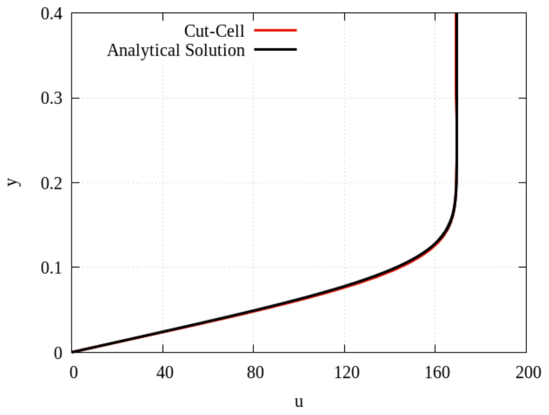


(a)

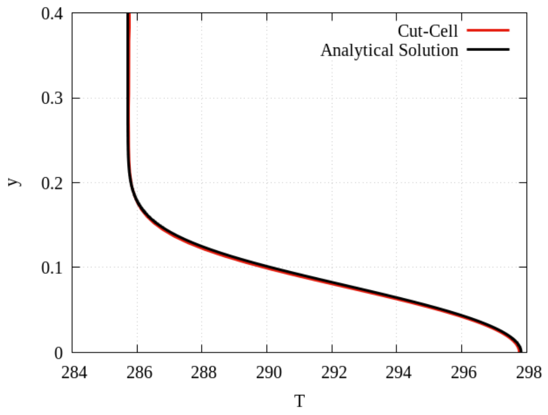


(b)

Figure 4.13: Laminar flow of a compressible fluid over flat plate: Velocity magnitude contours over (a) a horizontal and (b) an inclined plate. The non-uniform cut-cells do not affect the computed flow field smoothness.

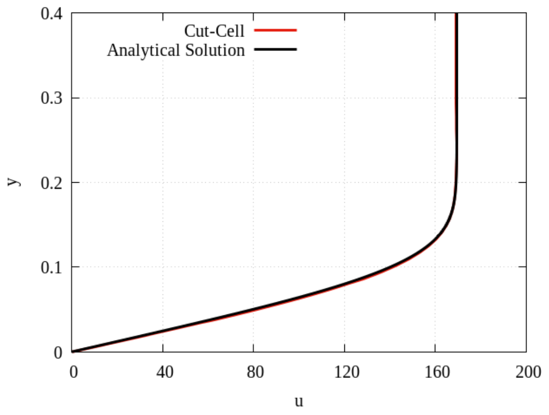


(a)

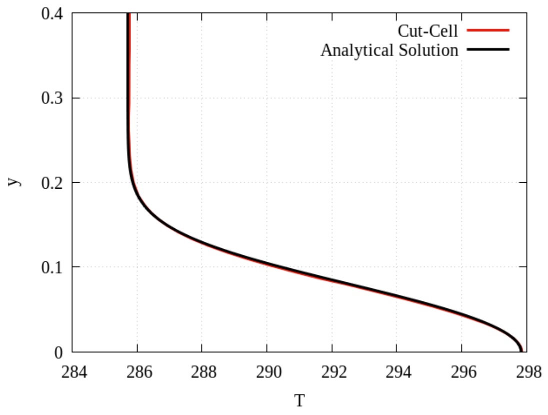


(b)

Figure 4.14: Laminar flow of a compressible fluid over flat plate: (a) Velocity and (b) temperature profiles over a horizontal plate at 90% of its length. Comparison with analytical solutions.



(a)



(b)

Figure 4.15: Laminar flow over a compressible flat plate: (a) Velocity and (b) temperature profiles over an inclined plate computed along a vertical line positioned at 90% of the flow domain length. Comparison with analytical solutions.

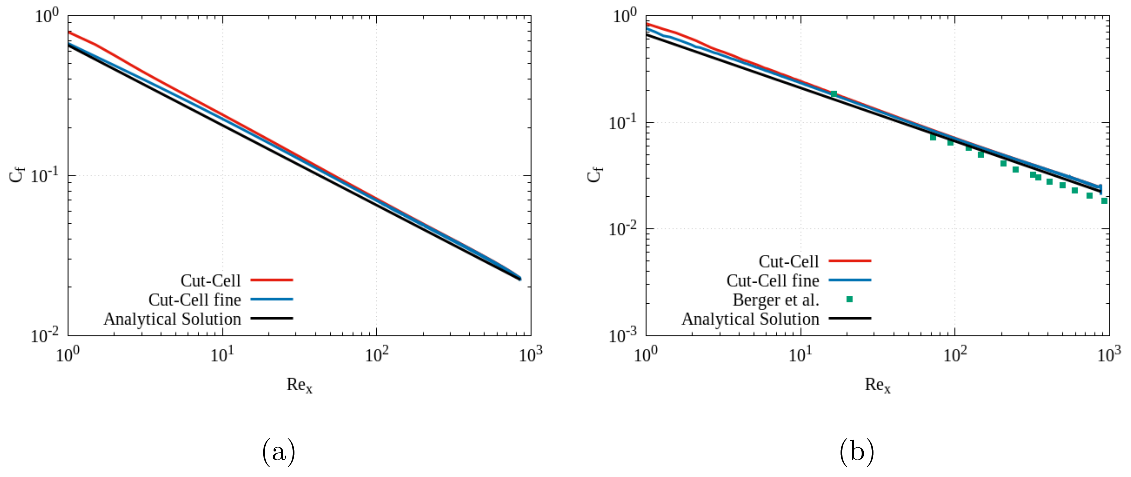
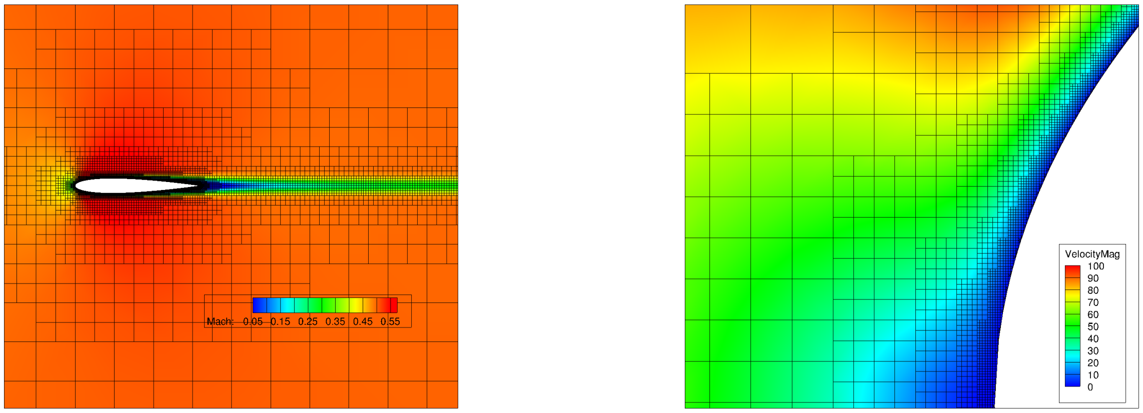


Figure 4.16: Laminar flow of a compressible fluid over flat plate: Skin friction coefficient for (a) a horizontal and (b) an inclined plate. Results from an other cut-cell software [36] are shown with green dots.

4.1.5 Laminar Flow Over the NACA0012 Isolated Airfoil

Another test case is a NACA0012 isolated airfoil exposed to a laminar flow at $M_\infty = 0.5$, $Re_\infty = 5000$, $Pr_\infty = 0.72$, and $\alpha_\infty = 0°$. In contrast to the flat plate case, the development of the boundary layer on curved walls is studied. Fig. 4.17a shows the mesh refinement close to the airfoil's surface to ensure the accurate boundary layer computation. Moreover, cells have been further subdivided into smaller parts in the downstream direction by defining, in the mesh generation software input file, the region where more refinement is needed. Fig. 4.17b shows a close-up view of the mesh around the airfoil's leading-edge, indicating the challenging process of computing a smooth boundary layer due to the problematic surface orientation concerning the Cartesian mesh lines direction. However, mesh non-orthogonality at the boundary does not affect the flow field smoothness. The Mach number field is presented in fig. 4.18. Finally, results are compared with corresponding data from [297], where a body-fitted structured C-type mesh of 8M nodes was used. In that study, the convective terms were discretized by applying a finite-volume approach and a three-point second-order scheme with a Roe-type numerical dissipation. The viscous terms were discretized with a second-order central difference approximation. Fig. 4.19 compares the pressure and skin friction coefficient distributions around the body with the aforementioned data from the literature. Both distributions are smooth along the airfoil and agree well with the cited results.



(a) (b)

Figure 4.17: Laminar flow of a compressible fluid over the NACA0012 isolated airfoil: (a) Mesh is refined close to the airfoil and downstream. (b) Mesh close-up view around the leading-edge.

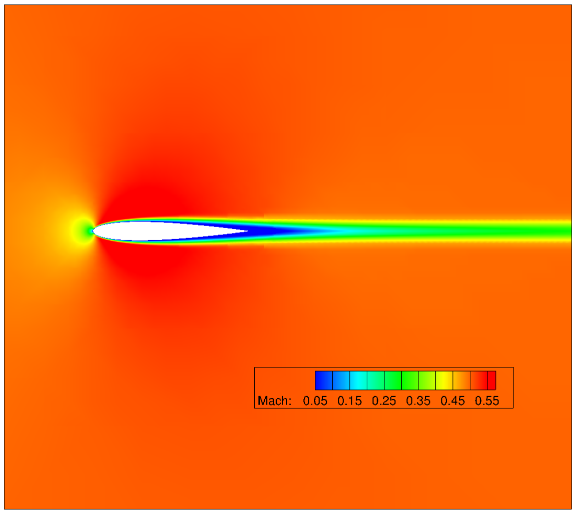
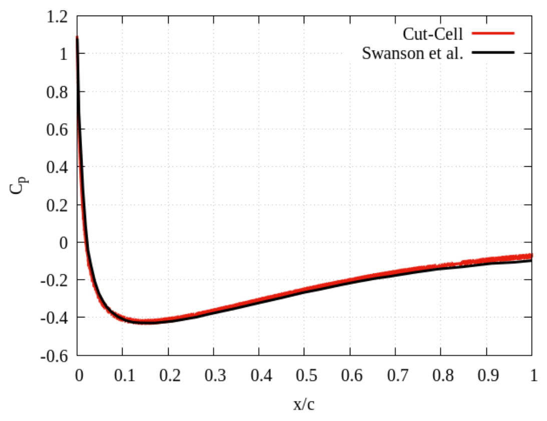
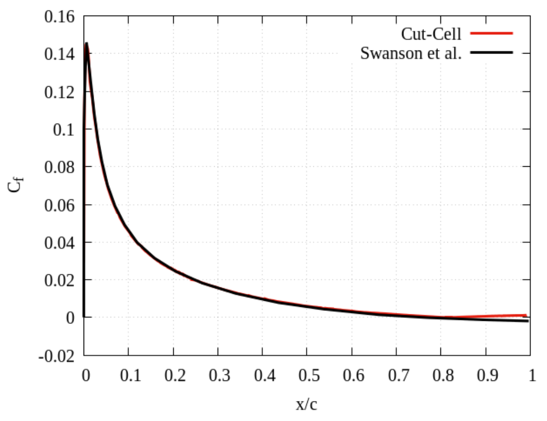


Figure 4.18: Laminar flow of a compressible fluid over the NACA0012 isolated airfoil: Mach number iso-areas computed by the cut-cell method.



(a)



(b)

Figure 4.19: Laminar flow of a compressible flow over the NACA0012 isolated airfoil: (a) Pressure and (b) skin friction coefficient distributions around the airfoil. Comparison between the cut-cell software results (red) and CFD results provided by [297] (black).

4.1.6 Inviscid Flow over ONERA M6 wing

This case is concerned with the flow over the ONERA M6 wing, which is a typical CFD validation case for external flows leading to a great number of CFD and experimental data available in the literature. The wing's analytical geometrical description can be found in [97]. Far-field flow parallel to the XZ plane (fig. 4.20) of $M_\infty = 0.84$ and $\alpha_\infty = 3.06°$ forms a transonic and turbulent phenomenon. However, the presented results of the cut-cell software, assume that the flow is inviscid. The following comparative study is based on results produced by CFL3D, a turbulent flow solver created by NASA which makes use of body-fitted meshes [286].

A mesh of 1.4M cells which adapts along the two shock waves formed in the wing's suction side was used. A slice vertical to the wing spanwise direction is displayed in fig. 4.20, where the Mach number field and mesh adaptation are shown. Assuming that the pressure coefficient at wing's sections along the span is accurately computed despite the inviscid flow simplification, a comparison is performed consisting of cut-cell, CFL3D results and experimental data provided by [275]. Fig. 4.21 shows the sections defined at 20%, 44%, 65%, 80%, and 90% of span length, where the pressure coefficient has been measured. Results plotted in fig. 4.22 show good agreement among the cut-cell software, CFL3D, and experimental data. Shock position has been accurately predicted causing an abrupt static pressure rise. The difference between experimental data and cut-cell results is significantly small and comparable with the corresponding difference from CFL3D results, confirming the cut-cell method's ability to successfully handle 3D applications.

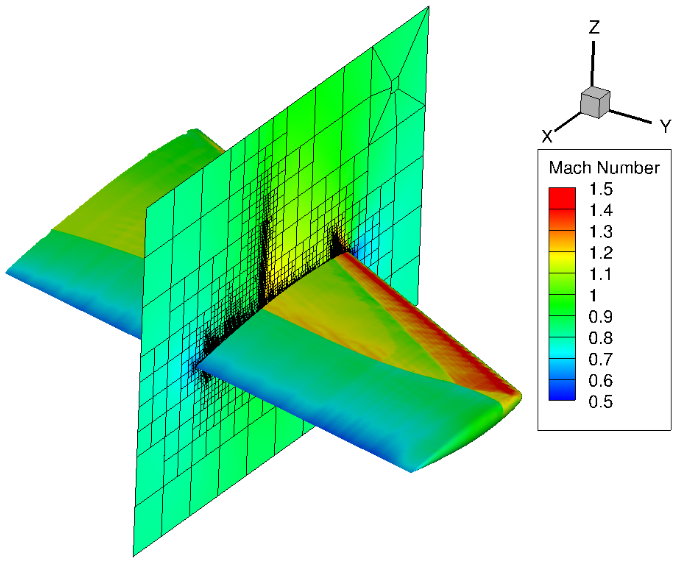


Figure 4.20: Inviscid flow of a compressible fluid over ONERA M6 wing: Mach number field on a slice perpendicular to the spanwise direction. Mesh is adapted close to the two normal shocks formed on the wing suction side.

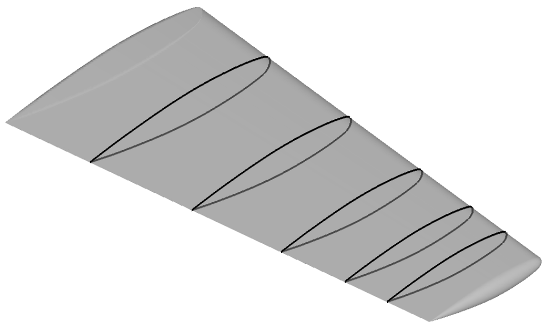
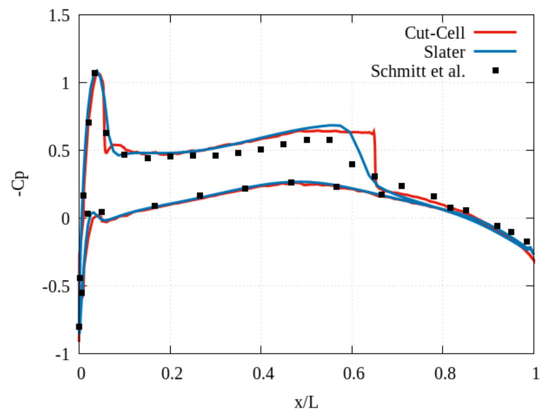
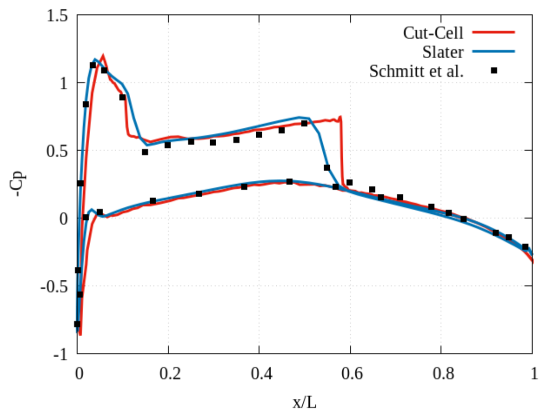


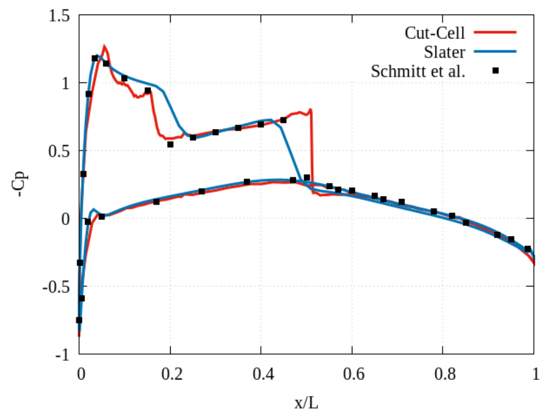
Figure 4.21: Inviscid flow of a compressible fluid over ONERA M6 wing: Slices defined at 20%, 44%, 65%, 80%, and 90% of span length, where the pressure coefficient is measured (see fig. 4.22).



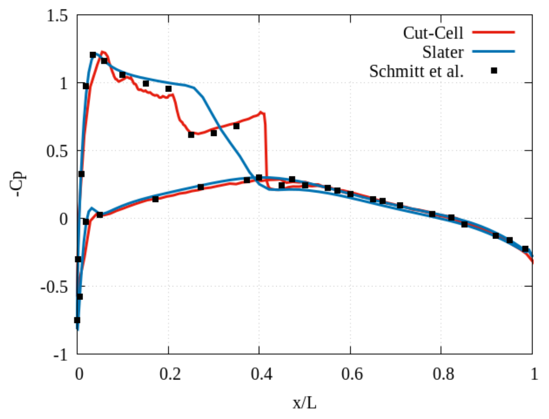
(a)



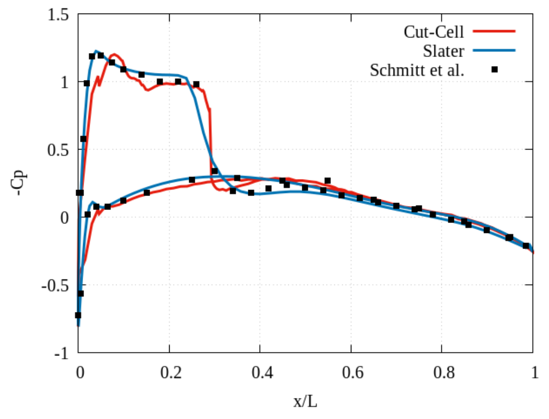
(b)



(c)



(d)



(e)

Figure 4.22: Inviscid flow of a compressible fluid over ONERA M6 wing: Pressure coefficient distribution on slices shown in fig. 4.21 starting from the wing's root and ending close to the tip. Results are produced by the cut-cell software (red), CFL3D (blue) and experimental measurements (black).

4.2 Incompressible Flow Solver Assessment

Herein, the programmed cut-cell flow solver for incompressible fluids is assessed in a number of internal and external aerodynamics test cases. The inviscid flow around the Joukowski airfoil, a cylinder, and inside a convergent-divergent duct is studied. The software's ability to predict viscous effects is analyzed through the laminar flow over a cylinder and inside a driven cavity and a 3D S-bend duct. The cut-cell software results are compared with experimental data or analytical solutions wherever available.

4.2.1 Inviscid Flow over the Joukowski airfoil

The Joukowski foil is a common case study in fluid dynamics, although it does not find any practical aeronautical application. Its value is derived from the existence of an analytical solution obtained by a conformal mapping introduced by Nicolai Zhukovsky, which transforms the well known potential flow over a cylinder to the flow past a family of airfoil shapes.

The airfoil's surface is defined as $x = Re(\zeta)$, $y = Im(\zeta)$, where ζ is given parametrically in the complex plane by

$$\zeta = z + \frac{a^2}{z},$$
$$\frac{z}{\alpha} = 1 + \frac{R}{\alpha}(e^{i\theta} - e^{-i\beta})$$

where $\theta \in [-\beta, 2\pi - \beta]$. Max. and min. θ values correspond to the trailing edge. The airfoil's baseline curvature is determined by the parameter β and its thickness by R/α, where α is the angle of attack (in rad). Velocity magnitude on its surface is given by

$$\frac{v}{v_\infty} = [2sin(\theta - \alpha) + sin(\alpha + \beta)] \left| \frac{z}{z - \frac{\alpha^2}{z}} \right|$$

Pressure and lift coefficients are equal to $C_p = 1 - \left(\frac{v}{v_\infty}\right)^2$ and $C_l = 8\pi \frac{R}{c} sin(\alpha + \beta)$ respectively, where c is the airfoil's chord. More details about the Joukowski conformal mapping can be found in [267]. The parameters mentioned above are: $\alpha = \beta = 5°$ and $R/\alpha = 1.1$.

The airfoil is exposed to flow of $v_\infty = 10\ m/s$ and $p_\infty = 1\ bar$. A mesh of 90K cells was used, and the simulation wall-clock time was 16 min. on 48 processors. The governing equations residual convergence is plotted in fig. 4.24. The velocity contours are shown in fig. 4.23. The pressure coefficient computed by the cut-cell method agrees well with the analytical solution except from a small region close to the leading edge, as shown in fig. 4.25. Its smoothness and accuracy should be mentioned. Moreover, a mesh sensitivity analysis is made, where the lift coefficient error is computed for every gradually refined mesh, fig. 4.26. The last two presented meshes have the same number of cells, and their only difference is detected in the number of points used for the airfoil's representation. 2K points were used in all but the last case, in which their number increased by a factor of 10. Increasing the number of points in the geometry significantly increases the lift coefficient accuracy. The cut-cell method allows for different geometry and mesh resolutions, and their incompatibility may cause inaccuracies. A general rule can be formulated, indicating that the geometry resolution should always be higher than the mesh resolution so as at least two points of the solid body's contour belong to each mesh cell. The lift coefficient error computed in the most refined mesh is significantly small, as demonstrated in table 4.4.

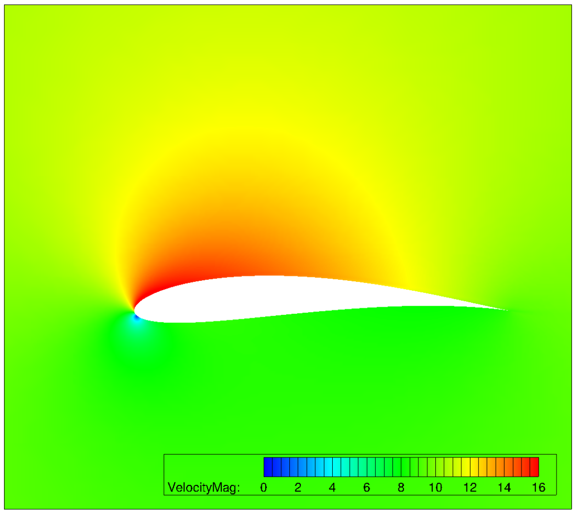


Figure 4.23: Inviscid flow of an incompressible fluid over Joukowski airfoil: Velocity magnitude iso-areas.

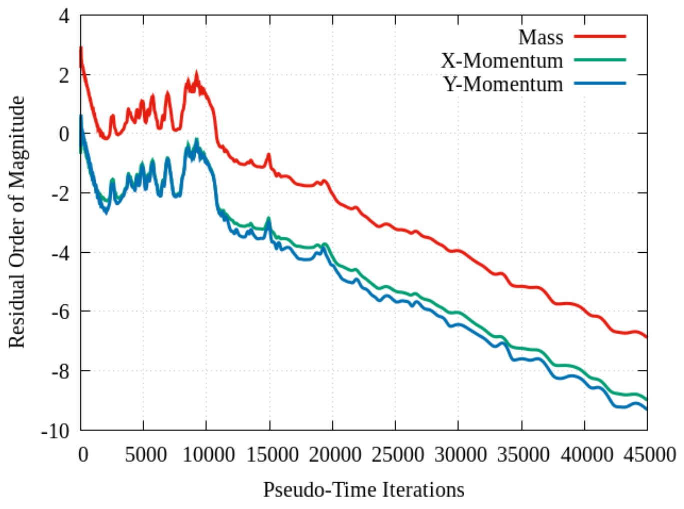


Figure 4.24: Inviscid flow of an incompressible fluid over Joukowski airfoil: Convergence of the residuals of mass and momentum equations in the cut-cell method.

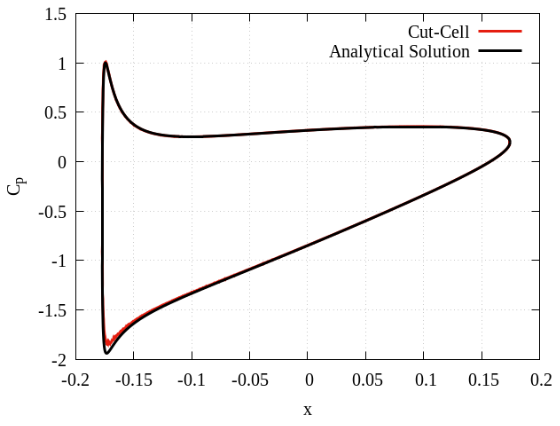


Figure 4.25: Inviscid flow of an incompressible fluid over Joukowski airfoil: Pressure coefficient distribution computed by the cut-cell software (red) are compared with the analytical solution (black).

	C_l
Cut-Cell	1.19092
Reference	1.19093
Deviation (%)	0.00084

Table 4.4: Inviscid flow of an incompressible fluid over Joukowski airfoil: Lift coefficient values given by the cut-cell software and the analytical solution. Their percentage deviation is shown as well.

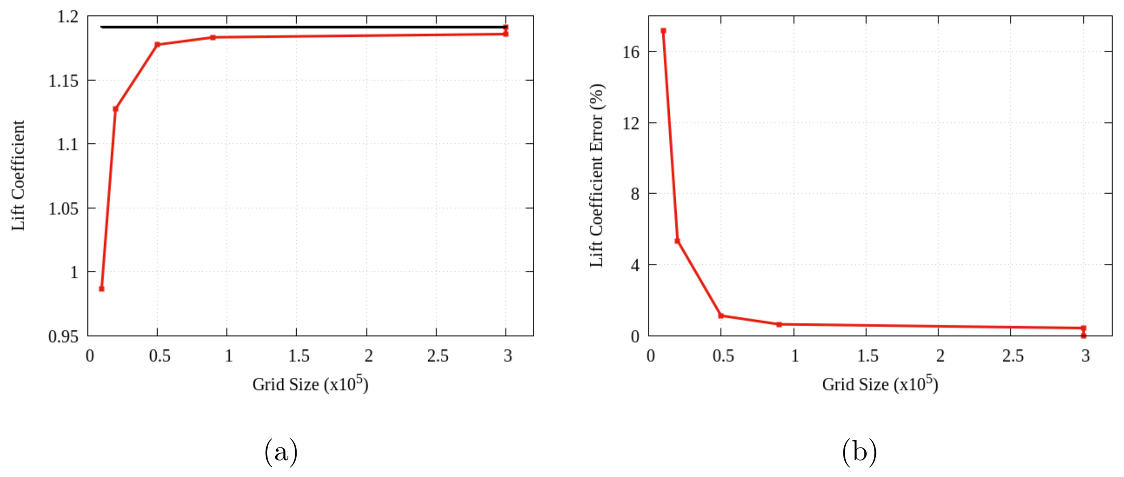


Figure 4.26: Inviscid flow of an incompressible fluid over Joukowski airfoil: (a) Lift coefficient and (b) the corresponding percentage error variation due to the gradual mesh refinement. The last two points represent two meshes with the same number of cells but a different number of points on the airfoil's surface. The black line in the left figure represents the value given by the analytical solution.

4.2.2 Inviscid Flow over cylinder

Flow simulation over a cylinder is pretty challenging for an IBM and a proper validation case for the cut-cell software. Due to its shape, the flow on its surface is far from parallel to the Cartesian mesh lines producing high artificial dissipation. Fig. 4.27 shows a mesh detail to illustrate the non-orthogonality of mesh lines on the solid boundary. The computational results are compared with the analytical solution available for potential flow [267]. According to this theory, the pressure coefficient and tangential velocity on the cylinder's surface parameterized as $\vec{x}=(cos\theta, sin\theta)$ is given by

$$v_t = -2v_\infty|sin\theta|$$
$$C_p = 1 - 4sin^2\theta$$

where v_∞ is the freestream velocity. Fig. 4.28 presents the velocity and pressure contours around the cylinder corresponding to the far-field conditions $v_\infty=20\ m/s$ and $p_\infty = 1\ bar$. Tangential velocity along the wall and pressure coefficient distributions are displayed in fig. 4.29. The plotted curves are smooth despite mesh irregularities close to its boundary. Comparison with the analytical solution certifies the software's accuracy.

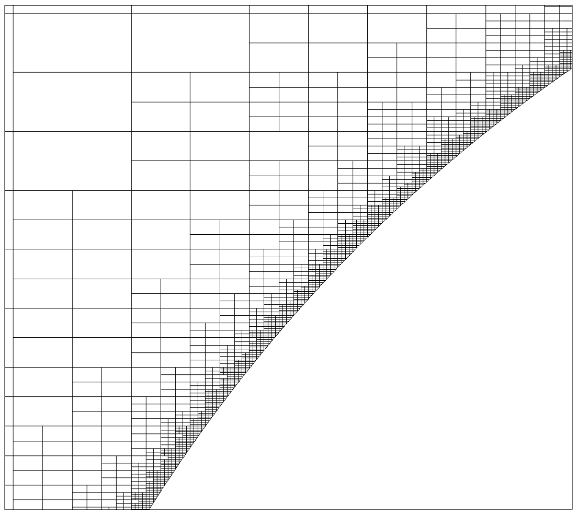


Figure 4.27: Inviscid flow of an incompressible fluid over cylinder: Mesh detail shows its irregularity close to the cylinder's boundary.

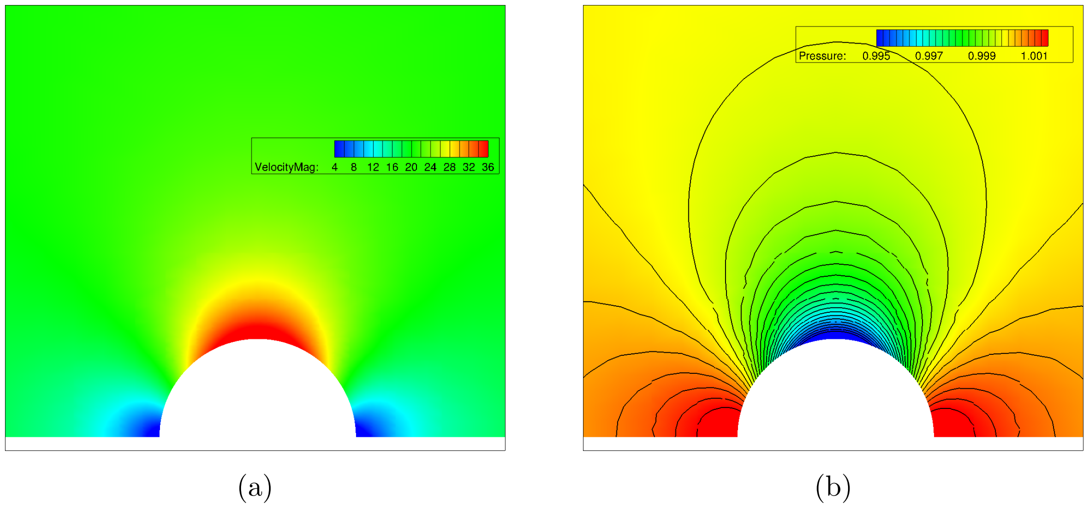
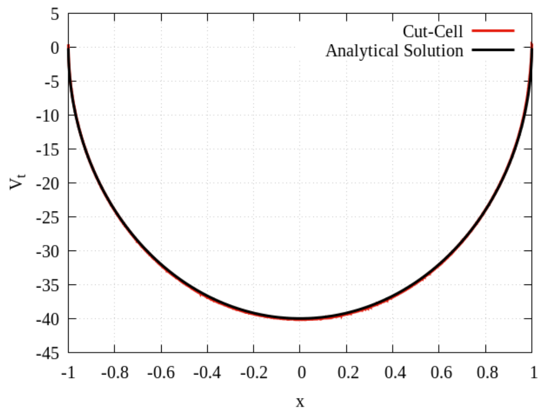
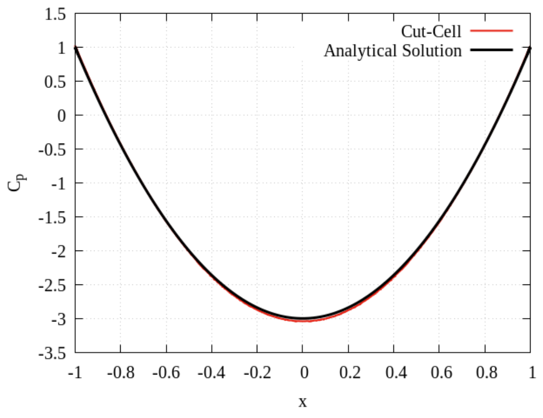


Figure 4.28: Inviscid flow of an incompressible fluid over cylinder: (a) Velocity magnitude contours and (b) iso-bar lines.



(a)



(b)

Figure 4.29: Inviscid flow of an incompressible fluid over cylinder: (a) Tangential velocity over the cylinder's surface. (b) Pressure coefficient distribution. Curves are given by the cut-cell software (red) the analytical solution (black).

4.2.3 Convergent-Divergent Duct Flow

The case of a flow inside a convergent-divergent duct allows for the study of the incompressible flow solver ability to satisfy mass and momentum conversation throughout the duct. The duct shape has already been defined in subsection 4.1.3. Total pressure (1 *bar*) is imposed at the inlet and static pressure (0.995 *bar*) at the outlet. Flow results accuracy is tested as follows. Firstly, fig. 4.30 shows the computed velocity iso-areas, where its symmetry is computationally verified. Secondly, mean velocity in every section is computed analytically and compared successfully with the numerical one fig. 4.31a. Finally, the software's ability to keep the total pressure constant along the duct is investigated, fig. 4.31b. Mass flow difference between the duct inlet and outlet, shown in table 4.5, is essentially small verifying the cut-cell method's high accuracy. Table 4.6 compares the inlet mass flow and the exerted axial force on the duct between the cut-cell and analytical solutions, certifying the softwares's high accuracy. The analytical value of the exerted force is computed based on uniform velocity profile assumption at each cross-section. Consequently, the proposed cut-cell method preserves the flow equations conservation property and prevents flow loss through the solid walls by successfully imposing no-penetration over the walls.

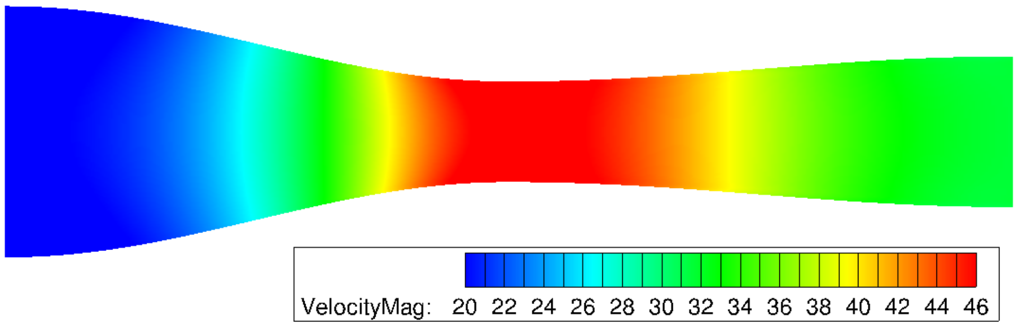


Figure 4.30: Inviscid flow of an incompressible fluid in duct: Velocity magnitude iso-areas.

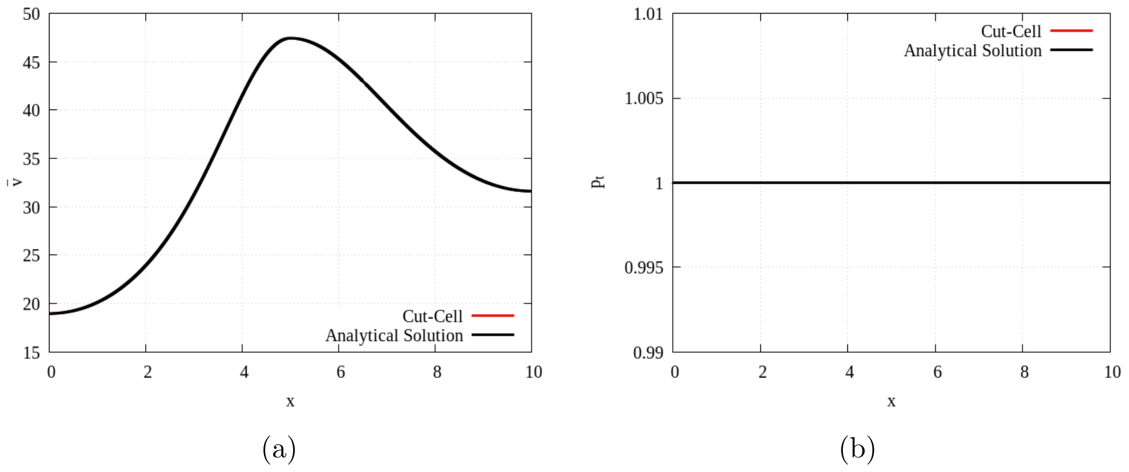


Figure 4.31: Inviscid flow of an incompressible fluid in duct: (a) Mean velocity and (b) total pressure at each duct cross-section. Comparison between cut-cell (red) and analytical solution (black).

	Mass (Kg/s)
Inlet	47.428
Outlet	47.431
Deviation (%)	0.0057

Table 4.5: Inviscid flow of an incompressible fluid in duct: Mass flow at duct's inlet and outlet. Their deviation is shown as well.

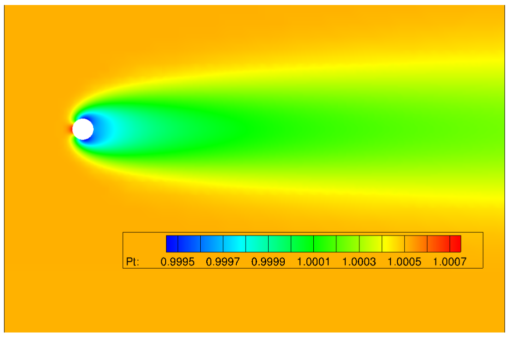
	Mass (Kg/s)	Force (kN)
Cut-Cell	47.428	99.7009
Reference	47.434	99.7
Deviation (%)	0.0063	0.0009

Table 4.6: Inviscid flow of an incompressible fluid in duct: Mass flow and axial force comparison between numerical and analytical results.

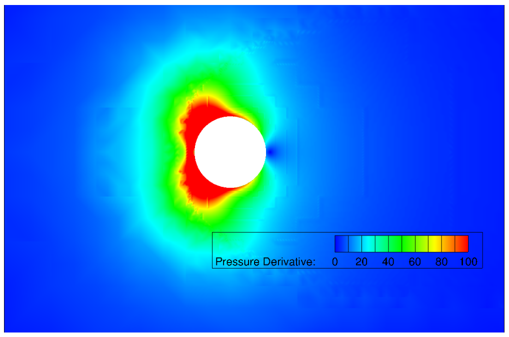
4.2.4 Laminar flow over a Cylinder

The analysis presented in the previous subsections completed the inviscid incompressible flow solver assessment. The next cases focus on the validation of the softwares's viscous part. The first case is concerned with the laminar flow over a circular cylinder for which extensive experimental and numerical data are available in the literature. Its experimental and numerical study lasts almost over a century and continues even today to analyze the complex cylinder wake flow phenomena. In the presented case, the resulting Reynolds number based on the cylinder's diameter is 10, which corresponds to a steady flow without periodic vortex shedding [94]. A dynamic mesh adaptation technique to the cylinder's wake is used, which allows the increase in flow simulation accuracy. The wake's region is identified by measuring the total pressure losses at each finite volume over the flow field, fig. 4.32a. Cells detected with a high amount of losses are subdivided into four smaller parts. This criterion also marks cells close to the cylinder's boundary before the flow separation occurs, and therefore not being part of the wake. For this reason, cells belonging to areas with high pressure gradient values are excluded from the refining process, fig. 4.32b. After four successive adaptations, the final mesh consists of 190K cells. Fig. 4.33 shows the final mesh along with the total pressure field, indicating the coincidence between the mesh refinement and the high total pressure loss areas. The flow equations convergence took 75 min. on 48 processors and is plotted in fig. 4.34. The velocity magnitude field over the cylinder is displayed in fig. 4.35.

Velocity measurements were made in the cylinder's wake [228], and results were compared with the aforementioned numerical solution. The origin of the Cartesian mesh is defined at the cylinder's center. The velocity distributions plotted at various x/d positions, d being the cylinder's diameter, agree well with the experimental results, fig. 4.36. However, there are some discrepancies, especially at large y/d, which are examined as follows. Fig. 4.37 shows the symmetrical velocity profile computed by the cut-cell method along $x/d=4$. The corresponding experimental data are not entirely symmetric due to small measurement errors which partially explains their difference from numerical results. Moreover, the velocity profile along $x/d=1$ differs notably from the measurements close to $y/d=0$. Fig. 4.38 compares the cut-cell results with data provided by a CFD software that uses body-fitted meshes [299]. The two software results come to a close agreement ensuring the argument that a small measurement error occurs in this specific area. Finally, table 4.7 shows the difference in the cylinder's drag coefficient between the numerical and experimental data. Their slightly high deviation is expected due to the numerical and experimental differences mentioned above.



(a)



(b)

Figure 4.32: Laminar flow of an incompressible fluid over a cylinder: (a) Total pressure and (b) static pressure gradient magnitude over the cylinder. Cells with high total pressure losses (blue-green area of the left figure) and small pressure gradient (blue area of the right figure) are subdivided into smaller cells increasing the flow simulation's accuracy.

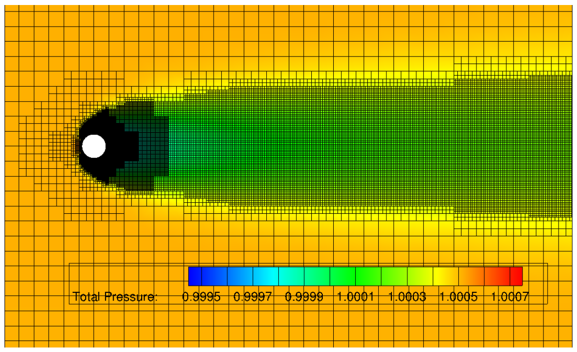


Figure 4.33: Laminar flow of an incompressible fluid over a cylinder: Final mesh adapted over the area of high total pressure loss.

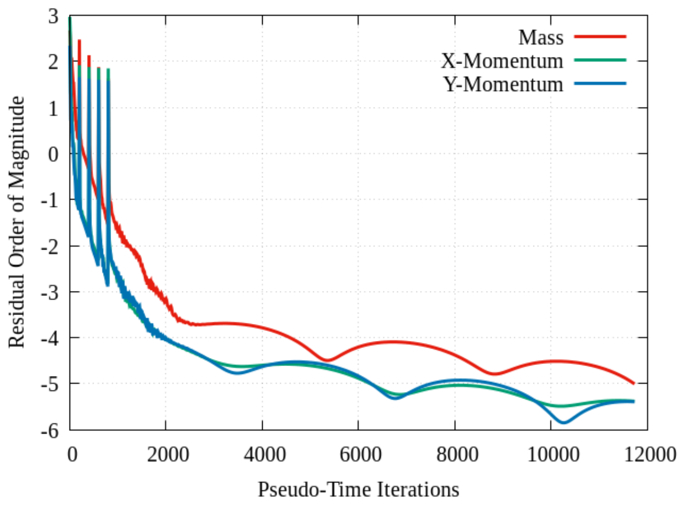


Figure 4.34: Laminar flow of an incompressible fluid over a cylinder: Convergence of the residuals of mass and momentum equations. Residual overshootings indicate the iteration at which mesh adaptation occurs.

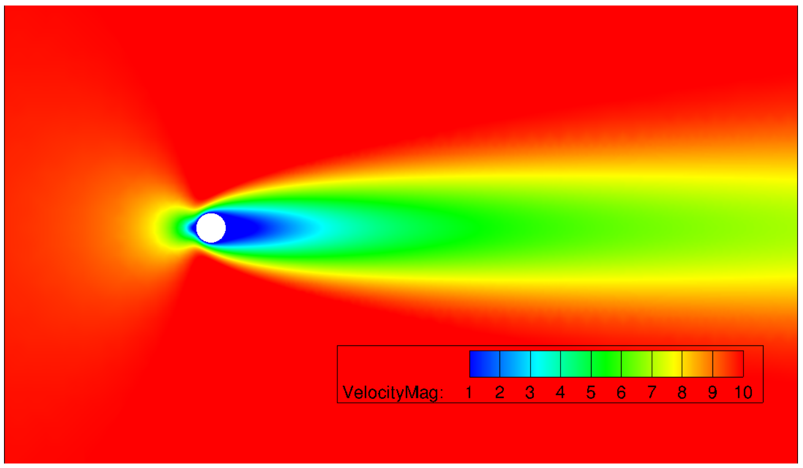
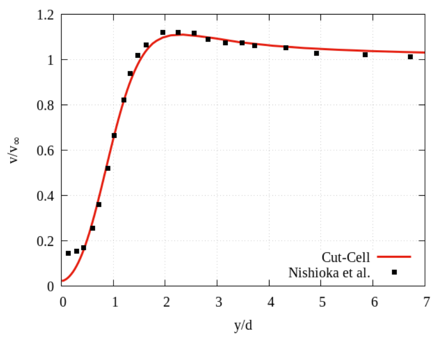
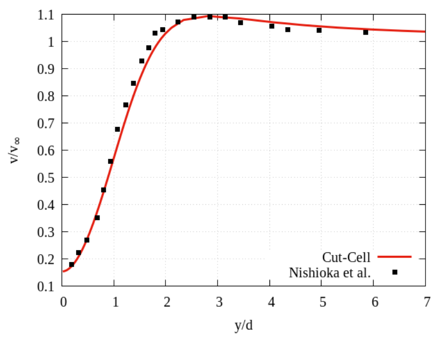


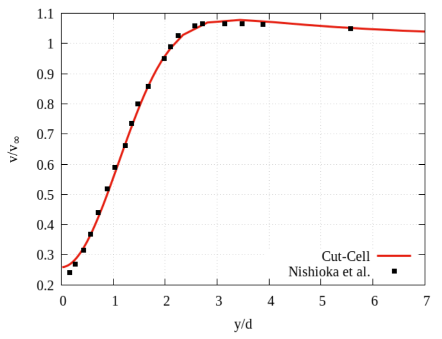
Figure 4.35: Laminar flow of an incompressible fluid over a cylinder: Velocity magnitude iso-areas.



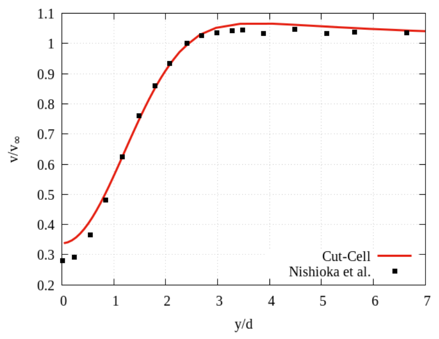
(a) $x/d\!=\!1$



(b) $x/d\!=\!2$



(c) $x/d\!=\!4$



(d) $x/d\!=\!7$

Figure 4.36: Laminar flow of an incompressible fluid over a cylinder: Wake velocity distributions measured at various distances from the cylinder. Comparison between cut-cell results (red) and experimental data [228] (black).

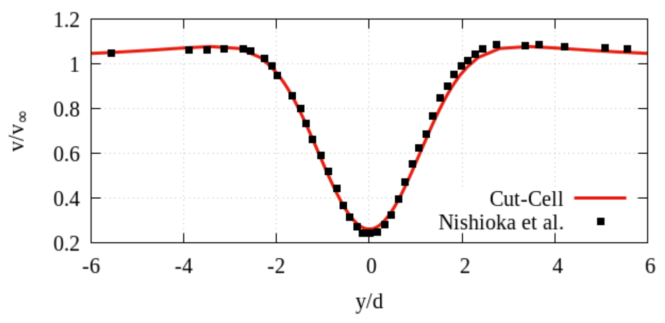


Figure 4.37: Laminar flow of an incompressible fluid over a cylinder: Symmetrical velocity profile at $x/d=4$ computed by the cut-cell software (red). Experimental data (black) are given by [228].

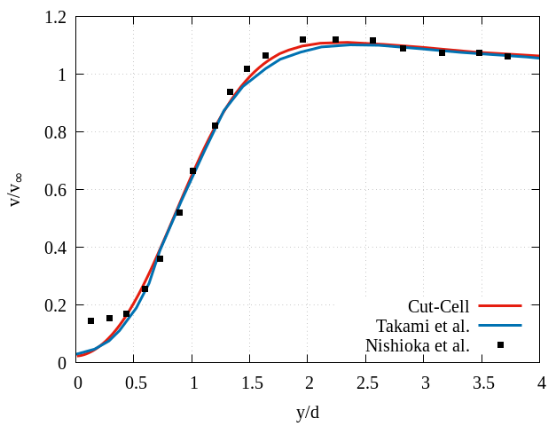


Figure 4.38: Laminar flow of an incompressible fluid over a cylinder: Velocity profile at $x/d = 4$ given by the cut-cell method (red), numerical results [299] (blue) and experimental data [228] (black).

	C_d
Cut-Cell	2.798
Reference	2.746
Deviation (%)	1.894

Table 4.7: Laminar flow of an incompressible fluid over a cylinder: Drag coefficient computed by the cut-cell software. Its deviation from the corresponding experimental data [228] is also shown.

4.2.5 Driven Cavity Flow

Despite the singularities at its corners, the laminar incompressible flow in a square-shaped cavity with its top wall sliding uniformly has been used very often as a problem for testing and assessing numerical techniques [2]. Published results are available for a wide range of Reynolds numbers. In this study, the Reynolds number is set equal to 1000, and results are compared with data given by [101]. These data are taken from a CFD software using a body-fitted mesh and have been cross-checked numerous times by many independent researchers during the last decades. A mesh of 140K cells is used. Its edges are purposely not coincident with the cavity's geometric boundaries, giving rise to cut-cells generation, which justifies the choice of this case as a verification case. Velocity magnitude iso-areas are shown in fig. 4.39. Streamlines show the development of a central, nearly circular vortex and a secondary vortex in each of the bottom corners. Fig. 4.40 shows velocity profiles along the horizontal and vertical cube axes of symmetry, on each of them the vertical and horizontal velocity component is plotted respectively. Moreover, vorticity distribution is displayed in a slice along the horizontal axis in fig. 4.41. Results are consistent with the analysis of [101], verifying the reliability and accuracy of the cut-cell method.

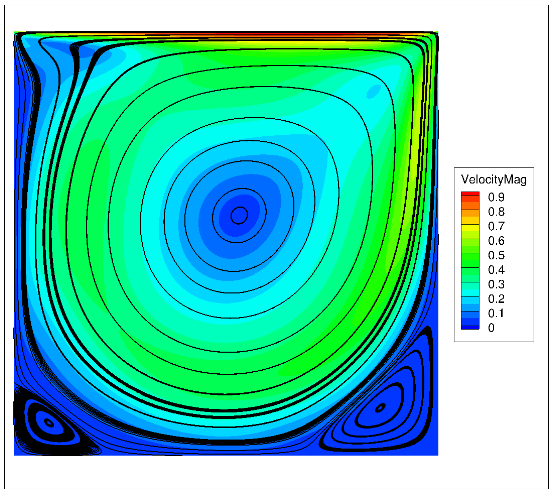


Figure 4.39: Laminar flow of an incompressible fluid inside the driven cavity: Velocity magnitude iso-areas and streamlines. Streamline pattern depicts the primary and the two secondary vortices.

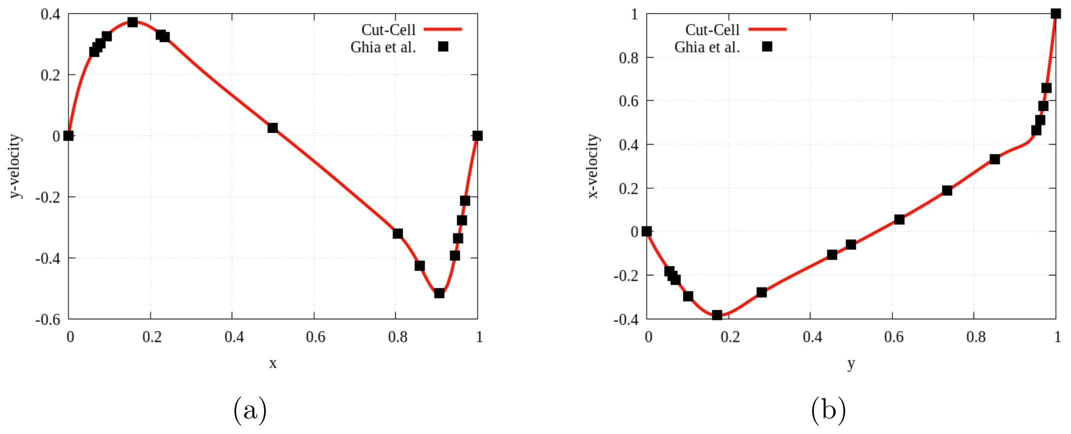


Figure 4.40: Laminar flow of an incompressible fluid inside the driven cavity: (a) Vertical velocity component along the horizontal axis of symmetry and (b) horizontal velocity component along the vertical axis of symmetry. Results are computed by the cut-cell software (red) and numerical data provided by [101] (black).

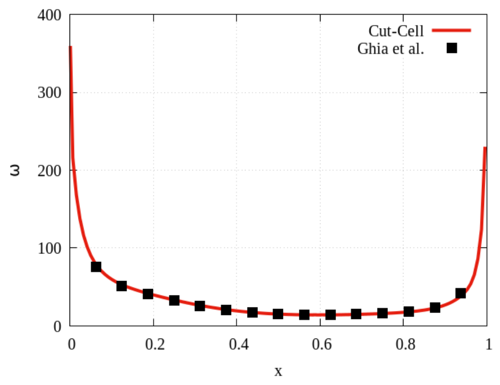


Figure 4.41: Laminar flow of an incompressible fluid inside the driven cavity: Vorticity magnitude distribution along the horizontal axis of symmetry. Comparison between results given by the cut-cell software (red) and numerical data from [101] (black).

4.2.6 Laminar flow in a 3D S-Shaped Duct

This study aims to validate the 3D cut-cell software by comparing its results with benchmark experimental data. A duct with a square cross-section creates a 3D laminar flow with mild curvature, small center-line displacement and, therefore, reduced flow separation at the duct walls, fig. 4.42. The mean line of the duct consists of two straight parts and two circular segments of opposite curvature. The exact geometry representation is given by analytical expressions described in [301]. The Reynolds number based on cross-section width is 790. A Mesh of 240K cells is used, details of which are shown in fig. 4.43, for the inlet cross-section and along the streamwise direction. Mesh is refined close to the walls to ensure correct prediction of the developed boundary layer. Fig. 4.44 presents the computed velocity contours in three cross-sections along the duct, the position of which is clear in fig. 4.42. In cross-sections 4.44a and 4.44b, just before the duct's second turn, secondary flows drive the boundary layer to thicken on the top wall. In cross-section 4.44c, the accumulation of low-speed fluid near the inner wall develops into vortices within the bend, causing a severely distorted flow field. The computed and experimental streamwise velocity profiles in the symmetry plane [301] are compared at five stations along the duct, shown in fig. 4.45, and agree well with one another. CFD results of a body-fitted structured mesh of 180K nodes are provided by [308] and are also plotted with a blue line. The two CFD results differ equally from the experimental data, proving that the cut-cell method's accuracy is equivalent to that of conventional

CFD methods using body-fitted meshes.

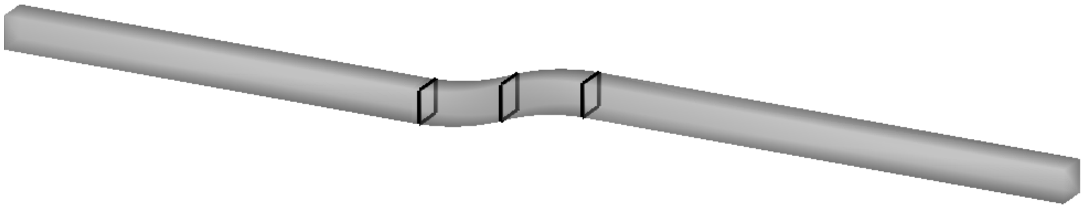


Figure 4.42: Laminar flow of an incompressible fluid in a duct: The S-shaped duct geometry. The three marked cross-sections depict the positions where the velocity fields are shown in fig. 4.44.

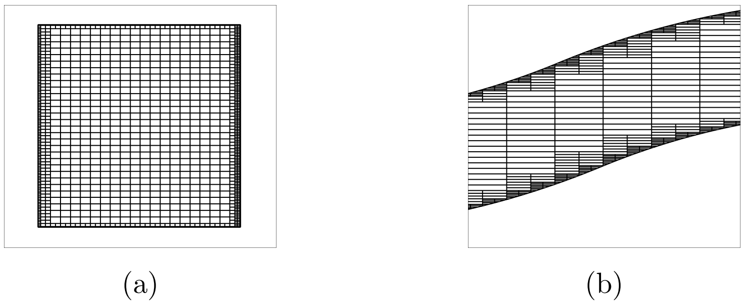


Figure 4.43: Laminar flow of an incompressible fluid in a duct: Mesh details (a) on the inlet cross-section and (b) on the duct's symmetry plane.

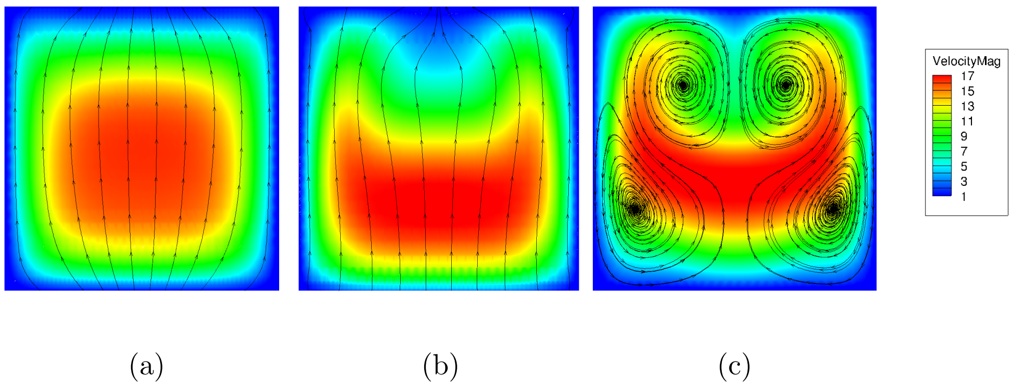


Figure 4.44: Laminar flow of an incompressible fluid in a duct: Velocity magnitude contours and streamlines at the three cross-sections along the duct depicted in fig. 4.42. The low-speed flow on the top wall in (a) and (b) creates a double vortex within the duct, shown in (c).

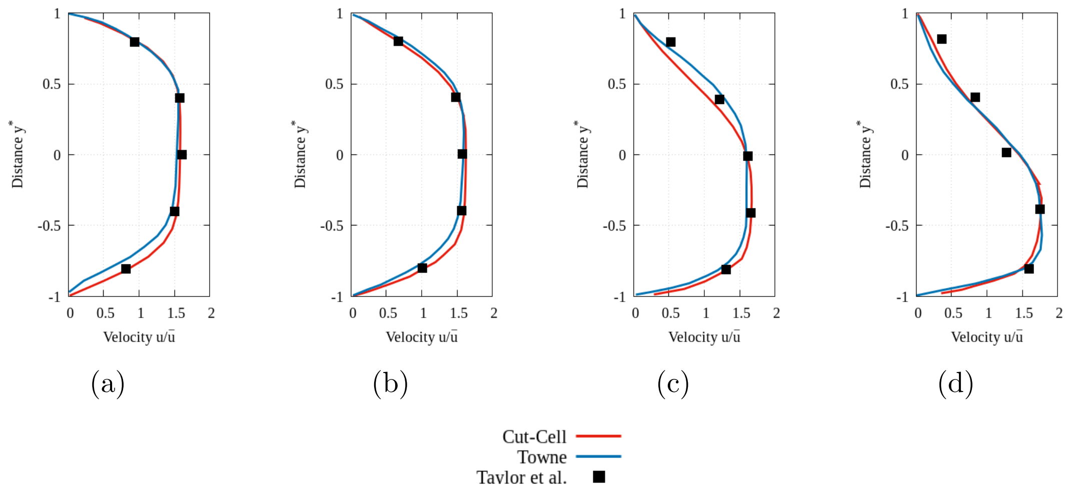


Figure 4.45: Laminar flow of an incompressible fluid in a duct: cut-cell (red) and experimental (black) streamwise velocity profiles over the symmetry plane. Computational results from [308] are also plotted (blue). Velocity is normalized by its mean value on each cross-section. $y^* = 2(y - y_{min})/(y_{max} - y_{min}) - 1$ is a proper non-dimensional parameter of the duct's height.

4.3 Unsteady Flow Solver Assessment

The cut-cell method for predicting flows around moving boundaries within a fixed Cartesian mesh is validated. The moving boundary slides on a fixed Eulerian mesh at each time step, avoiding repetitive re-meshing, which may harm the flow solver's speed and robustness. Cells close to moving boundaries change shape after geometry's motion and even appear or disappear from the fluid domain. The developed method's ability to maintain conservation for large displacements of moving boundaries while still retaining the solver's accuracy is the target of this section. The primary analysis concerns examining a pseudo-2D application of a moving piston and a more complex 2D case of an oscillating airfoil. The study assumes compressible inviscid flows and presents numerical results, including comparisons with analytical solutions or other CFD results and experimental data.

4.3.1 Piston Motion

The method's ability to satisfy the flow conservation laws is demonstrated by simulating a piston propagating through an initially quiescent fluid inside a tube. Two different cases are studied. In the first case, the piston moves into the fluid creating a shock wave traveling along the tube. In the second case, the piston is pulled back causing an expansion wave. Both phenomena are 1D and are described by analytical expressions [181]. Here, a 2D uniform mesh and solver are used to verify the method's ability to prevent flow leakage through the piston walls. In both cases, the piston's velocity corresponds to $M = 2$ and its displacement is equal to the width of around three cells, meaning that at least three cells appear or disappear from every mesh row at each time step. Despite the piston's considerable displacement, mesh quality remains excellent and the use of complicated deformation tools is avoided, fig. 4.46. Pressure contours are plotted at three different time steps for both cases in figs. 4.47 and 4.48. Numerical results are compared with the exact analytical solution. Pressure and density distributions along the tube are plotted in figs. 4.49 and 4.50. In the first case, the shock is predicted at the correct location ahead of the traveling piston. If conservation were not satisfied, the shock would be formed in the wrong position [219]. In the second case, the agreement between numerical and analytical solutions in the expansion region is very good.

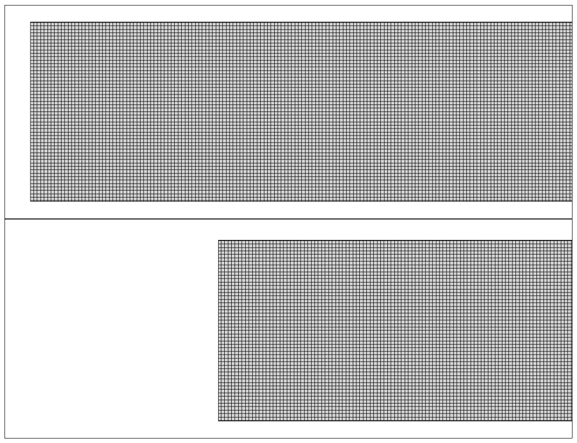


Figure 4.46: Inviscid flow of a compressible fluid in a piston tube: Mesh corresponding to two piston's positions.

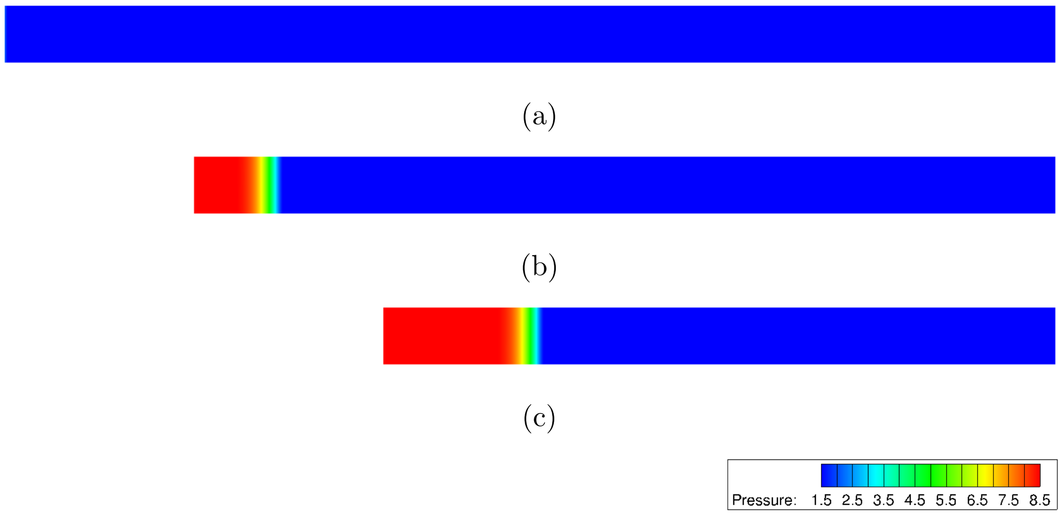


Figure 4.47: Inviscid flow f a compressible fluid in a piston tube: Piston moves to the right, forming a shock wave propagating through the fluid.

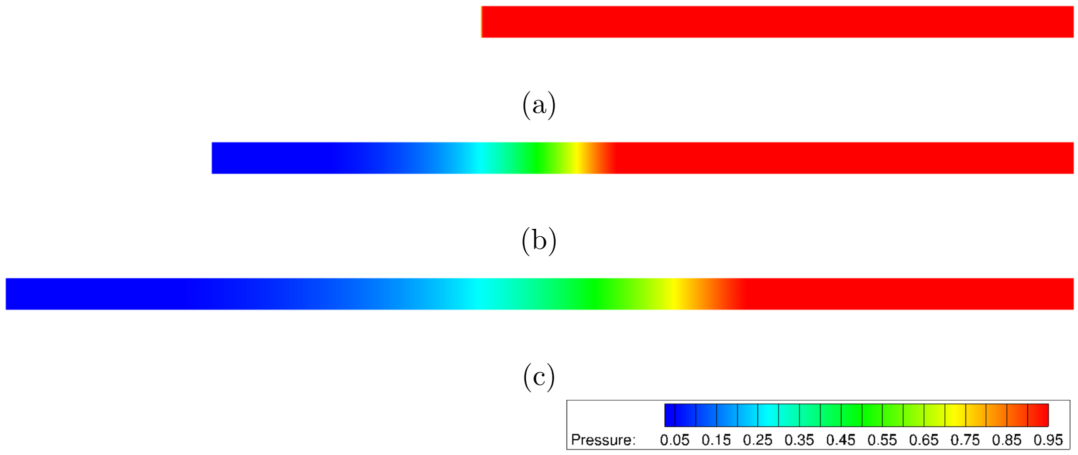
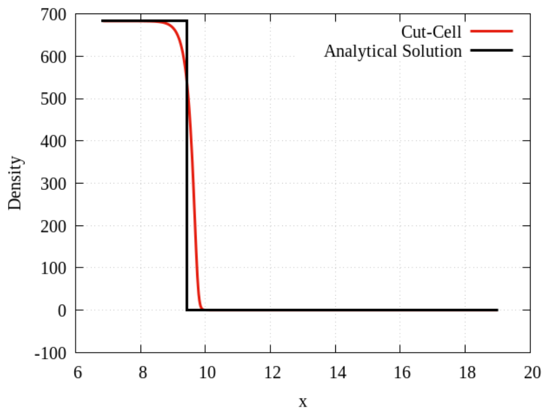
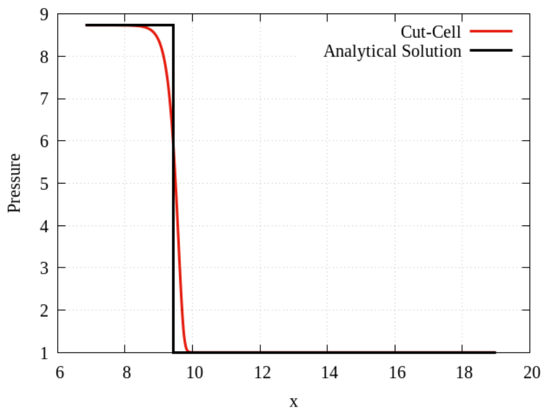


Figure 4.48: Inviscid flow of a compressible fluid in a piston tube: Piston moves to the left, producing an expansion wave along the tube.

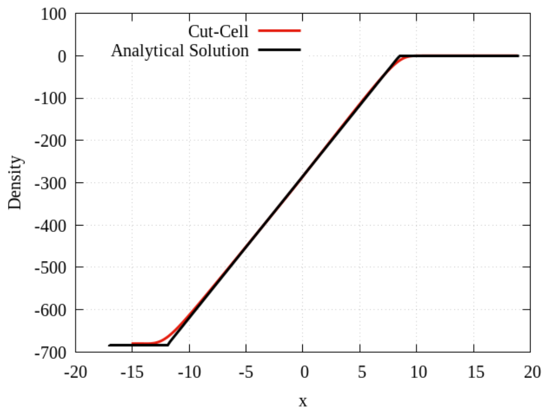


(a)

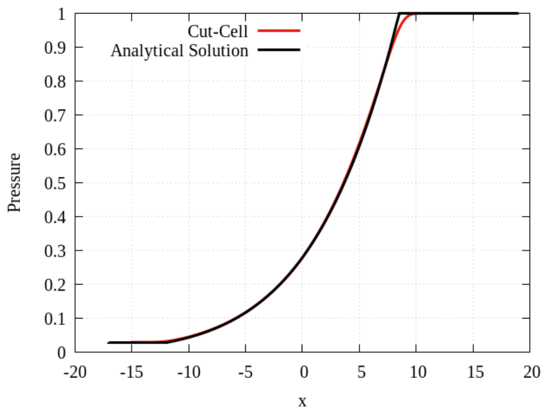


(b)

Figure 4.49: Inviscid flow of a compressible fluid in a piston tube: Comparison between numerical and analytical solutions, when piston moves into the fluid.



(a)



(b)

Figure 4.50: Inviscid flow of a compressible fluid in a piston tube: Comparison between numerical and analytical solution, when piston is pulled back.

4.3.2 Flow around Oscillating NACA0012

Results of compressible inviscid flow over an oscillating NACA0012 airfoil are presented. This case has been studied extensively by many researchers using body-fitted [255] or Cartesian meshes [219]. The pitching motion around the quarter-chord is prescribed by the sinusoidal function $\alpha(t) = \alpha_\infty + \alpha_0 sin(\omega t)$, where $\alpha(t)$ is the angle between the airfoil's chord and the x-axis. The far-field angle-of-attack and amplitude are $\alpha_\infty = 0.16°$ and $\alpha_0 = 2.51°$, respectively. The reduced frequency is $k = \omega c/u_\infty = 0.1628$ and the free-stream Mach number is 0.755. A steady transonic field initializes the unsteady phenomenon, in which five periods are computed. The simulation uses 150 time steps per complete cycle of the airfoil motion.

As the airfoil oscillates, the shock shifts between the airfoil's upper and lower surfaces. Shock is well captured due to mesh adaptation, which follows its motion. Moreover, mesh refinement close to the airfoil's boundary is adjusted at each time step. These two phenomena and the cells' appearance or disappearance from the flow domain can be seen in fig. 4.51, where the airfoil is shown at two snapshots of its motion. Fig. 4.52 shows the C_L–α curve, where the initial transition and periodicity establishment is evident. Phase lag between the variations in angle-of-attack and lift causes the characteristic hysteresis. The curve is compared with experimental [174] and numerical data from two different sources [15], [309]. It closely matches the obtained CFD results and differs from the experimental points, leading to the conclusion that error may affect measurements' accuracy. Four time-equidistant Mach number contour snapshots are shown in fig. 4.53. The relative motion scheme affects neither the contours' smoothness nor the shock's sharpness. The hysteresis is also evident in fig. 4.53b and fig. 4.53c, where the airfoil passes through its equilibrium position on the upstroke and downstroke, respectively.

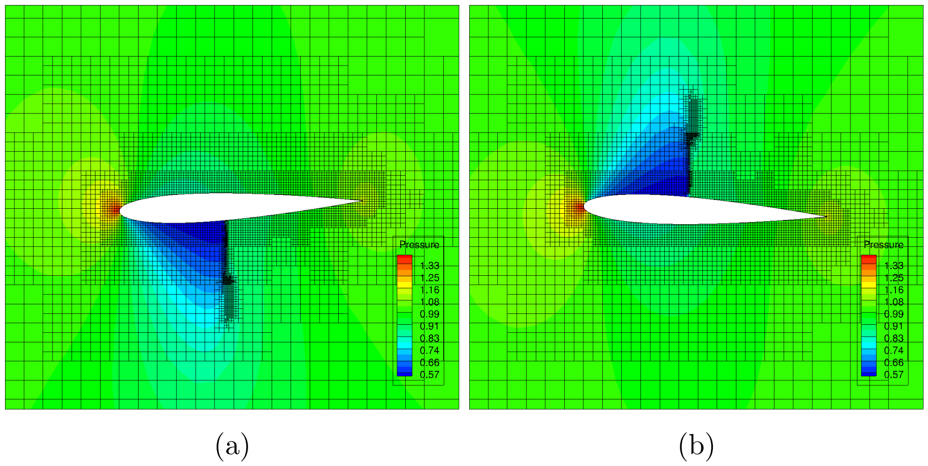


Figure 4.51: Inviscid flow of a compressible fluid around an oscillating NACA0012 airfoil: Mesh and iso-bar contours at two snapshots of the oscillating airfoil. Mesh adapts to the shock wave location and the airfoil's contour. Moreover, cells appear or disappear from the flow domain at each time step.

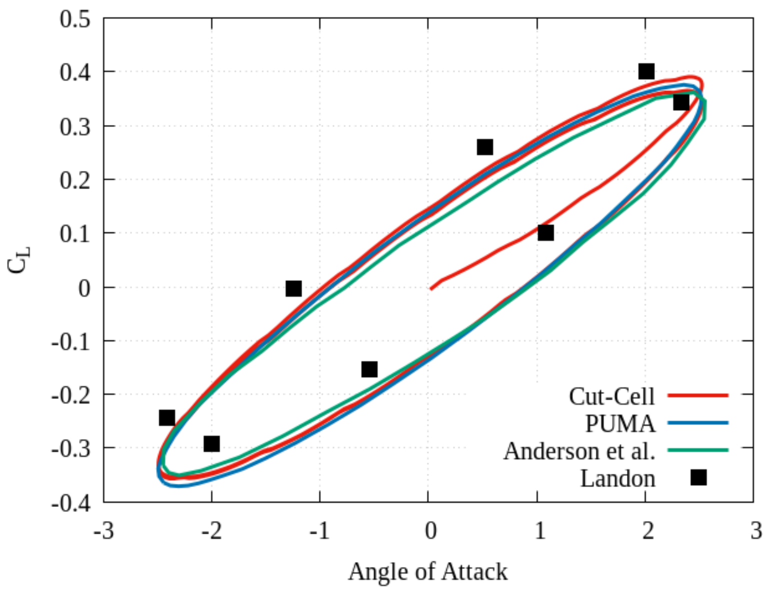
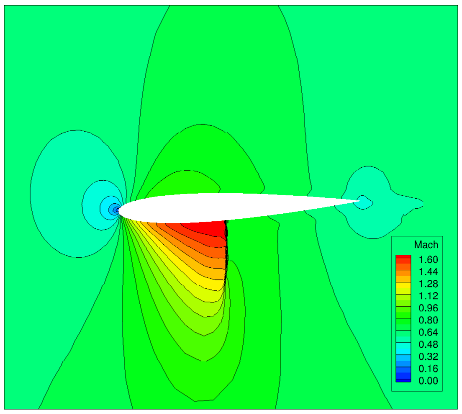
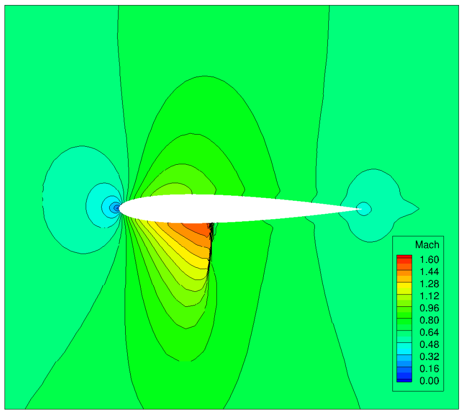


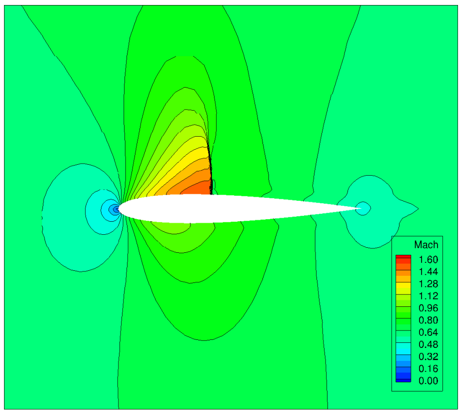
Figure 4.52: Inviscid flow of a compressible fluid around an oscillating NACA0012 airfoil: Time lag between airfoil's motion and the computed lift. Comparison among results of the proposed cut-cell method (red), experimental data (black) [174] and other CFD softwares [15] (green) and [309] (blue).



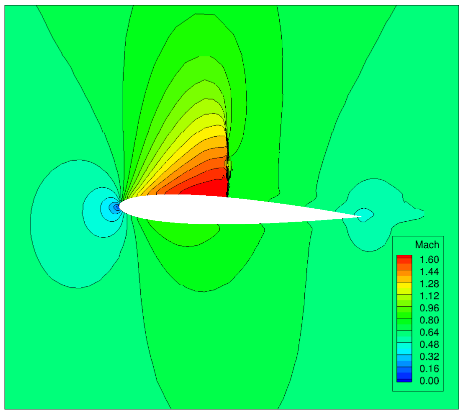
(a)



(b)



(c)



(d)

Figure 4.53: Inviscid flow of a compressible fluid around an oscillating NACA0012 airfoil: Mach contours at four equidistant period's time steps. (a) and (d) snapshots correspond to the airfoil's oscillation extreme positions, while (b) and (c) correspond to the equilibrium position.

Chapter 5

Flow Simulation in Industrial Applications

The IBMs were introduced to overcome difficulties related to the classical CFD methods' inability to handle flow simulations around complex geometries or solid bodies' complex motion. In some cases, these methods are the only possible alternative available due to body-fitted mesh generation or deformation unsuccessful attempts. Such applications can widely be found in the industry, where flow simulation accuracy is important since small computational errors could probably lead to considerable financial costs. Consequently, the cut-cell method remains a perfect choice due to its superiority regarding accuracy compared to other IBMs.

This chapter aims to show the developed method's ability to handle current industrial cases. An explanation is given in each application about the reasons why the use of the cut-cell method is essential and much more efficient than other CFD approaches. The presented applications deal with the unsteady compressible or incompressible internal flows in a moving valve, a scroll machine, a diaphragm pump, and a submersible pump. Their CFD analysis is part of projects investigating new designs of the machines and mechanism mentioned above. The produced results' accuracy and reliability are not under examination due to the absence of experimental or corresponding CFD results in the literature. However, the reader is referred to chapter 4 for a detailed study on the proposed method's validation/verification.

5.1 Incompressible Flow inside a Butterfly Valve

This case is concerned with the incompressible flow inside a valved duct. Valves are widely used in many applications to prevent undesirable backflow. The valve is modeled as a shutter rotating around an axis and moves from open to closed position in $0.5s$, fig. 5.1. In the presence of large boundary movements or when two bodies approach and finally touch each other, mesh deformation becomes costly, delicate, or even impossible. Therefore, the cut-cell software suits perfectly as it avoids morphing a body-fitted mesh. The coarse background mesh remains stationary while the immersed valve is allowed to move, covering and uncovering grid cells. Mesh is continuously refined at each time step close to the moving geometry to increase the flow simulation accuracy. Fig. 5.2 shows three time-instants during the valve's motion. Mesh generation and partitioning at 26 time steps in total and computation of all information needed to transfer the flow solution from one mesh to the next takes 36 min in one processor. According to fig. 5.3, mesh size ranges from 830K to 900K cells, and around 8K cells appear or disappear at each time step. At the last time step, the valve is positioned in parallel with mesh lines, and mesh is refined only along the streamwise direction explaining the abrupt decrease in cells' number. Moreover, at the same time step, the valve covers a notable amount of cell rows, increasing the number of disappeared cells, fig. 5.2b. The duct's length and diameter are set equal to 2 m and 10 cm, respectively. Total and static pressure are imposed at its inlet and outlet so as the isentropic velocity defined as $v_{inv}^2 = 2(p_t^{in} - p^{out})/\rho$ is equal to 0.045 m/s. The cut-cell software ran for 9 hours in 24 processors to complete the 26 time steps of the unsteady phenomenon. Velocity magnitude and pressure fields for different valve positions are shown in fig. 5.5, respectively. As the valve rotates and gradually blocks the flow motion, the velocity field reduces in magnitude until it becomes zero and the pressure field becomes uniform by taking on two distinct values before and after the valve. Flow trajectories around the rotating disc are presented in fig. 5.4.

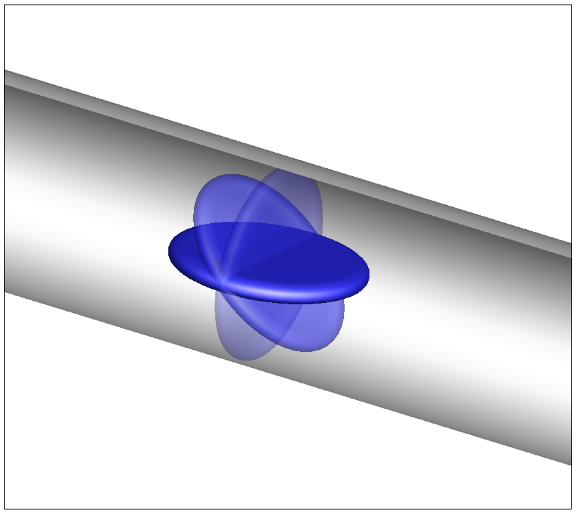
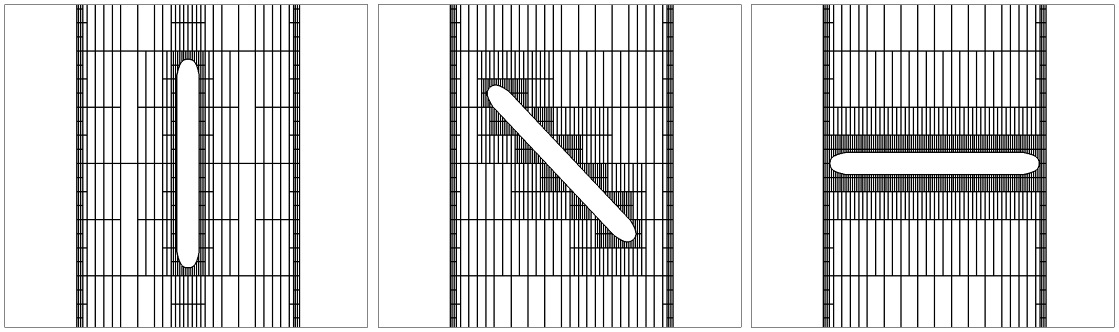
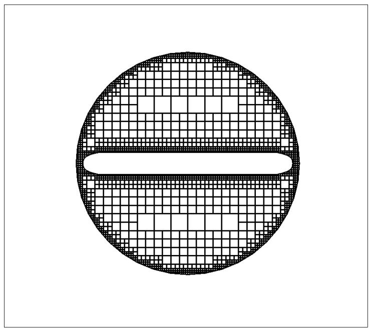


Figure 5.1: Laminar flow of an incompressible fluid inside a valved duct: Butterfly valve inside duct, rotating around an axis.



(a)



(b)

Figure 5.2: Laminar flow of an incompressible fluid inside a valved duct: (a) Adapted Cartesian mesh in a slice along the streamwise direction for three positions of the butterfly valve. (b) View of the valve at a duct's cross-section in the valve's fully open position.

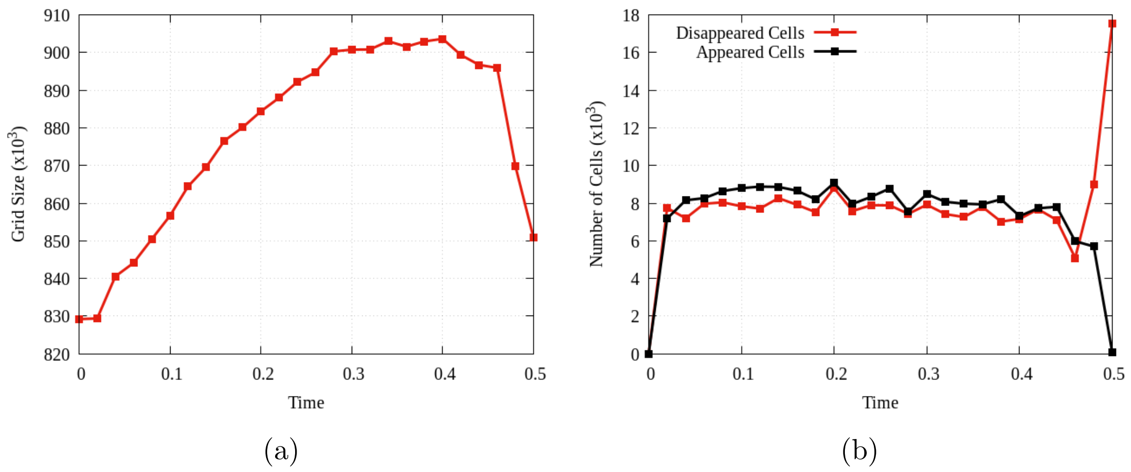


Figure 5.3: Laminar flow of an incompressible fluid inside a valved duct: (a) Mesh size evolution for each time step during the unsteady simulation (b) Number of disappeared and appeared cells at each time step caused by the valve's rotational motion.

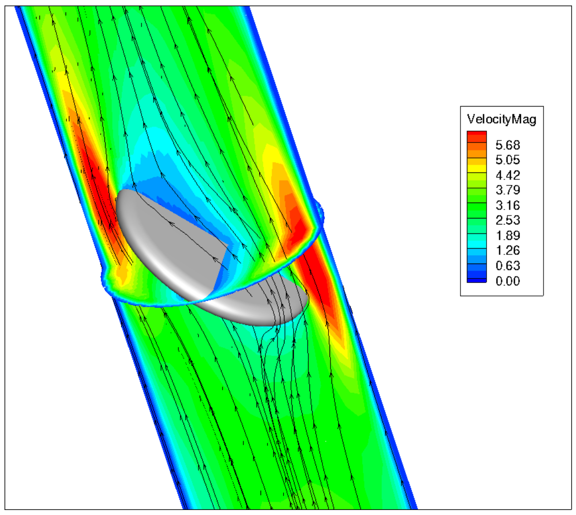
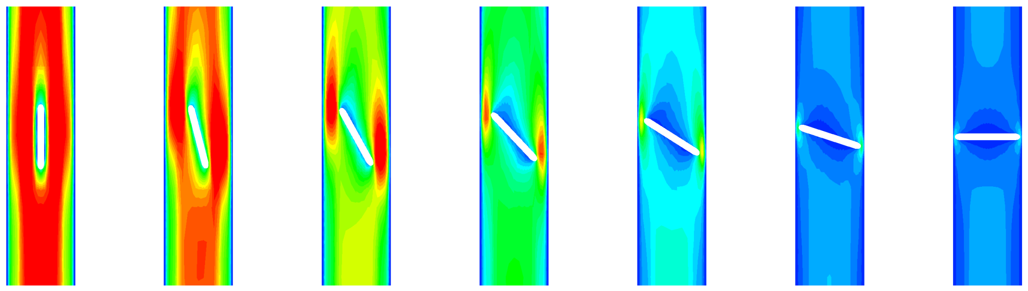
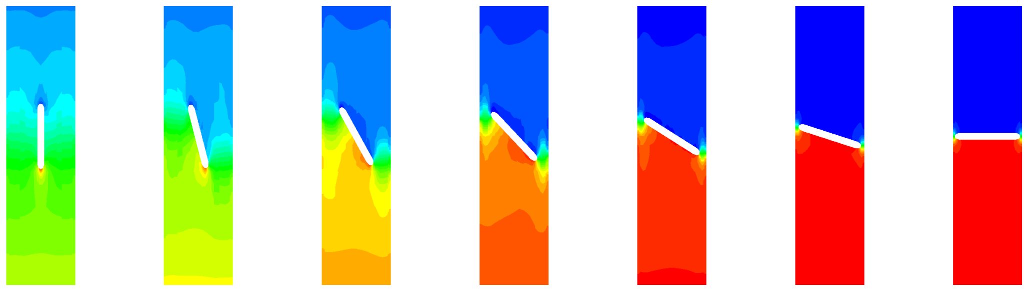


Figure 5.4: Laminar flow of an incompressible fluid in a valved duct: Flow trajectories within the valved duct with the disc half-open.



(a)



(b)

Figure 5.5: Laminar flow of an incompressible fluid inside a valved duct: Instantaneous (a) velocity magnitude and (b) pressure iso-areas within the valved duct, at equally distributed time instants.

5.2 Compressible Flow in a Scroll Expander

The interest in the theoretical and experimental Organic Rankine Cycle's (ORC) study has grown dramatically in the past decades [253]. The ORC produces electric energy from low temperature, non-costly heat sources, such as solar thermal power, geothermal heat sources and engine exhaust gases, making it a promising environmentally friendly technology [303]. Its name refers to the organic working fluid, which can evaporate at a lower temperature than water. One of the essential parts of an ORC is its expander. Displacement-type machines are more beneficial than turbo-machines due to their lower flow rates, higher pressure ratios, and much lower rotational speeds [243]. Scroll expander is an advantageous displacement machine due to its reduced number of moving parts, wide output power range, and low manufacturing cost [342]. However, experimental and CFD work is limited regarding scroll machines in expander mode [175]. The present work attempts to fill a part of this literature gap by proposing the cut-cell method as an alternative tool for the flow simulation on the scroll machine's complex geometry, with a lot of advantages. Research presented in this subsection is part of the results of a project titled "*Development of a small-scale low-temperature supercritical ORC with optimized scroll expander and heat exchanger*" funded by the Business Plan "Cooperation 2011 - Partnership of Manufacturing and Research Parties Specialized in Research & Technology Sectors." with Greece and the European Union's co-financing.

Creux first proposed the scroll machine in 1905 as a new compressor design [72]. Only after 75 years and the appearance of accurate and reliable tools was its manufacturing possible. The interest in its geometric modeling and dynamical analysis led to a significant amount of publications, one of the first complete studies being [216]. The following analysis is based on [31], where the details and proofs of the following mathematical analysis can be found. The scroll machine consists of two symmetric spirals, fig. 5.6c. The one is stationary, fig. 5.6a, while the other is allowed to orbit around the first one, fig. 5.6b. Orbiting is defined here as the motion where the Cartesian coordinate axes of the two spirals remain aligned while the moving origin orbits around the other. Flow is entered at the scroll's center and pushes the orbiting spiral along its way to the outlet. Fig. 5.8 presents the scroll's operation during one period of its motion. Blue parts correspond to areas occupied by the fluid. As the red spiral moves, the blue area increases, and fluid's pressure decreases. Finally, a generator is responsible for converting the scroll's motion to electrical energy.

The stationary spiral geometry is based on two involutes, the inner and the outer, unwrapping from a circle of radius r_b, called the base circle. The machine's coordinate system origin is placed at the center of the stationary base circle. The involute's parametric equations are given by

$$x(\phi) = r_b[cos\phi + (\phi - \phi_0)sin\phi]$$
$$y(\phi) = r_b[sin\phi - (\phi - \phi_0)cos\phi]$$

where ϕ corresponds to the range from ϕ_0 to ϕ_e. The inner and outer involutes differ in the value of these parameters, which are defined as ϕ_{i0}, ϕ_{ie} and ϕ_{o0}, ϕ_{oe}, respectively. Only the part from ϕ_s to ϕ_e, correspondingly for both involutes, is part of the spiral geometry, fig. 5.7a. The rest part from ϕ_0 to ϕ_s, called the two-arc discharge region geometry in scroll compressors terminology, is shown in fig. 5.7b. It consists of two arcs (red and blue) continuing the inner and outer involutes respectively and a straight tangent to the arcs line (green) of length L. The exact value of both cycles center and radius arises from a geometrical analysis developed in [272] and is presented in the same figure. It can be proved that the spiral thickness is

$$t_w = r_b(\phi_{i0} - \phi_{o0})$$

The moving spiral is reflected through the stationary one's origin and shifted by r_0 computed as

$$r_0 = r_b\pi - t_w$$

The displacement vector is

$$\vec{r}_0 = r_0 \left(cos(\phi_{ie} - \frac{\pi}{2} - \theta), sin(\phi_{ie} - \frac{\pi}{2} - \theta)\right)$$

where θ is a function of time and corresponds to different positions of the orbiting spiral. More details can be found in [272]. In our model every 2D geometrical representation of a scroll machine is uniquely defined by the parameters $r_b, \phi_{is}, \phi_{ie}, t_w$ and L. Their values are given in table 5.1 generating the specific geometry studied in this section, where h is the scroll's height.

Leakage loss between the two spirals is responsible for a significant reduction in the scroll's efficiency and, therefore, is one of the most important and challenging phenomena to model in a scroll expander. Most CFD software have difficulties in

r_b	$0.004968\ m$
ϕ_{is}	$240.78°$
ϕ_{ie}	$800.20°$
t_w	$0.0042\ m$
L	$0.0094\ m$
h	$0.1467\ m$

Table 5.1: Laminar flow of a compressible fluid in scroll: Parametric values generating the scroll machine studied in this section.

simulating the leakage flow as they can hardly account for the mesh generation and deformation due to scroll geometry and motion complexity [333], [88]. There are several widely used models available. The most common treat the flow as an isentropic compressible flow through a nozzle [191], [337]. The proposed method avoids using these models, and it is capable of solving the flow equations through the leakage and, therefore, computing the machine's quantities of interest, such as the expansion ratio, with an acceptable accuracy.

Mesh generation in the scroll geometry is a very challenging task. As far as conventional methods are concerned, a body-fitted mesh should be generated in the blue area of fig. 5.8a and follow the domain's motion until its final shape, shown in fig. 5.8i. Its transformation is so extreme that any mesh deformation tool would probably fail. On the other hand, a Cartesian mesh remains stationary during the spiral's orbit, following its motion by applying refinement techniques at each time step in areas close to solid boundaries. Mesh generated at the starting point of the scroll's operation is shown in fig. 5.9. Fig. 5.10 focuses on two mesh details, showing the proposed mesh generator capabilities to successfully handle "abnormal" geometrical shapes. In fig. 5.10a, complex cut-cells have been formed close to the wall's corners, while in fig. 5.10b, cut-cells have filled the narrow gap between the stationary and moving spirals allowing for the proper leakage simulation. Extra refinement around that area significantly increases the mesh size and, therefore, is avoided due to computational resource restrictions.

The transient laminar flow of a compressible fluid inside the scroll machine is simulated during a single operating cycle. Speed of revolution is set equal to 2000 *rpm*. Flow is entered from a circular hole of a radius of 0.01 *m*, and its direction is perpendicular to the scroll's plane. Inlet boundary conditions are total pressure 40.37 *bar* and total temperature 90° *C*. Velocity magnitude iso-areas at 10 equidistant time instants are shown in fig. 5.11. The initially quiescent fluid is distorted by the spiral's motion, which pushes it through the exit reducing its pressure. Finally, the discarded fluid exits from the outer square boundaries. Fluid motion is more apparent by the streamlines' direction, shown in fig. 5.12. Scroll's pressure ratio and mass flow rate are 2.77 and 0.605 kg/s, respectively.

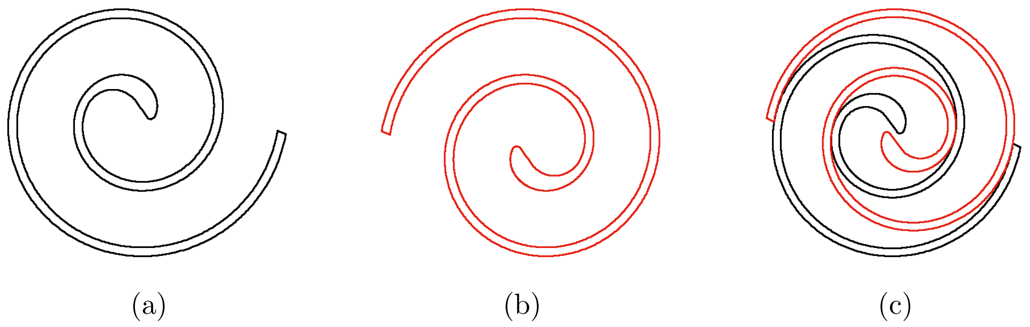


Figure 5.6: Laminar flow of a compressible fluid in a scroll: (a) Stationary and (b) orbiting spirals generated by using the parametric values from table 5.1. Their collaboration is shown in (c).

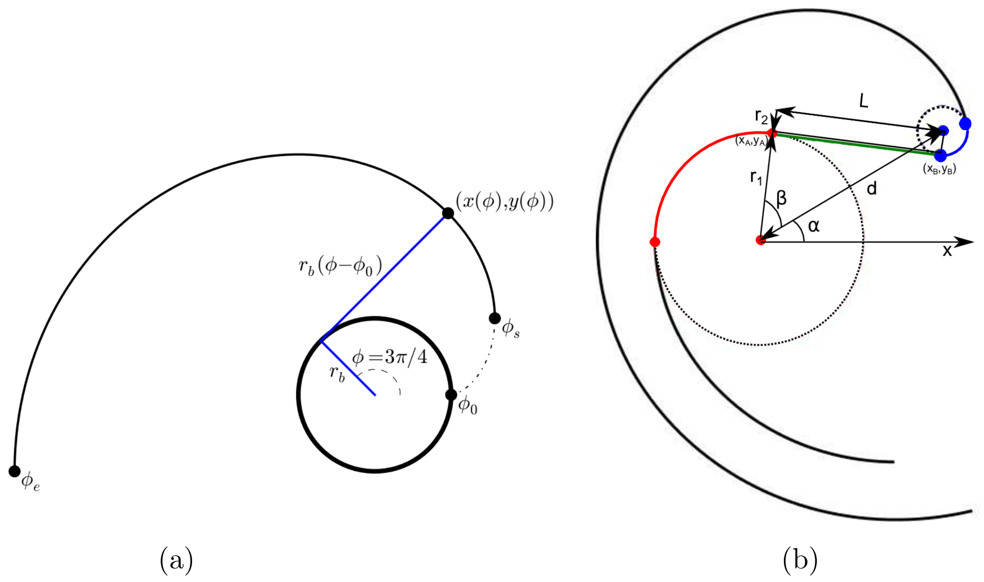


Figure 5.7: Laminar flow of a compressible fluid in a scroll: (a) Involute unwrapped from circle. Characteristic variables ϕ_0, ϕ_s, and ϕ_e are also shown. (b) The discharge region geometry consists of two arcs in red and blue and a straight line in green. Post-processed figures taken from [31].

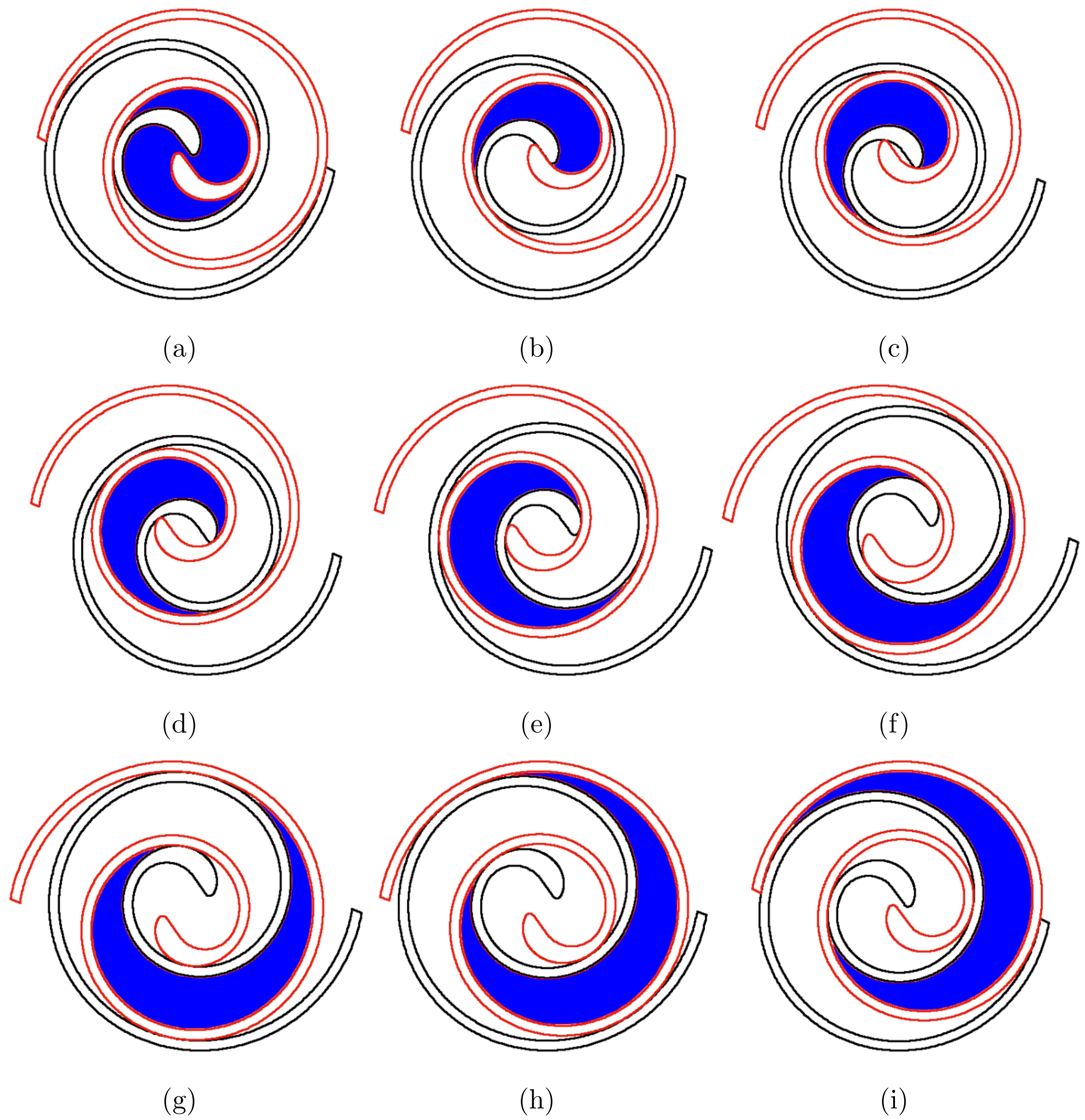


Figure 5.8: Laminar flow of a compressible fluid in a scroll: Equidistant snapshots in a period of scroll's motion. The blue area covered with fluid depicts the flow decompression.

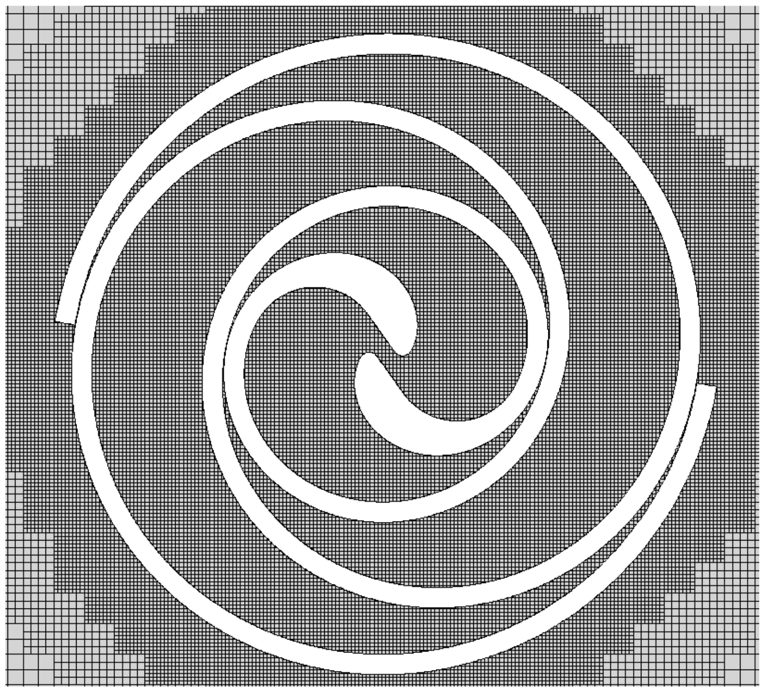


Figure 5.9: Laminar flow of a compressible fluid in a scroll: Mesh generated at the first time instant of scroll's motion.

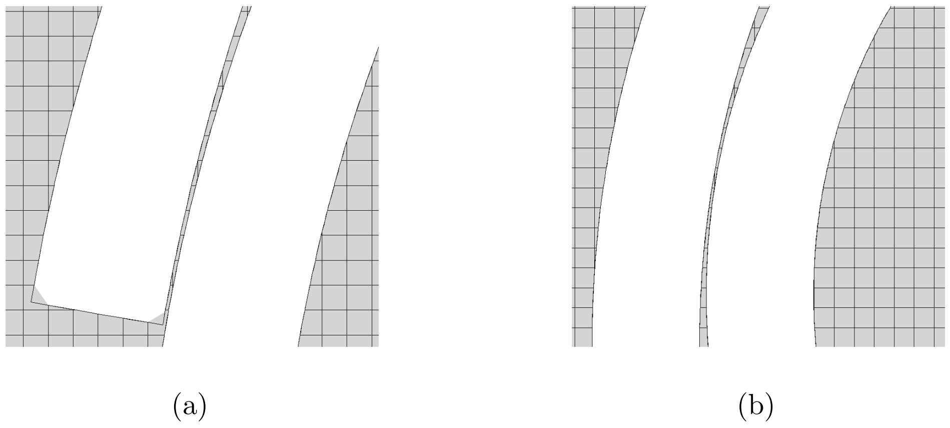
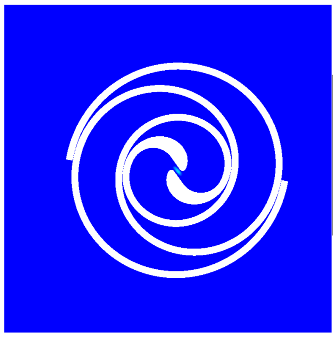
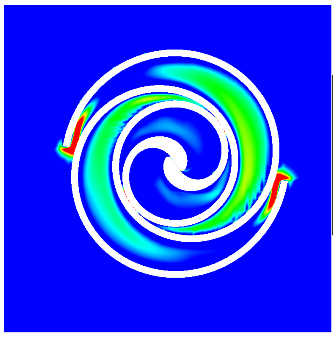
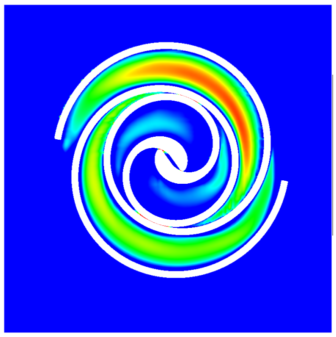
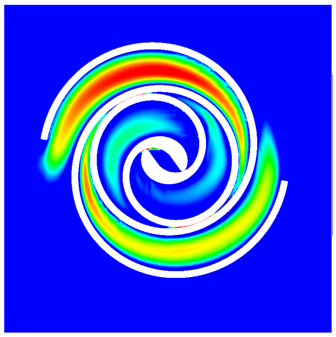


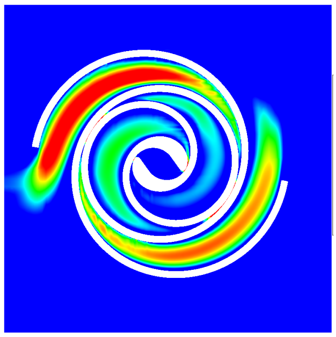
Figure 5.10: Laminar flow of a compressible fluid in a scroll: Mesh details in (a) the vicinity of spiral's edge and (b) the gap of a tiny small thickness between stationary and moving spirals.

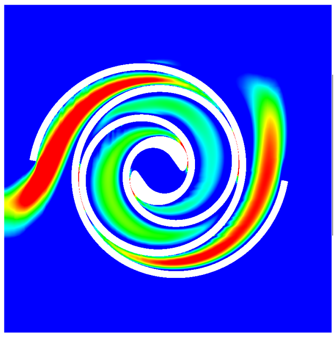

(a)

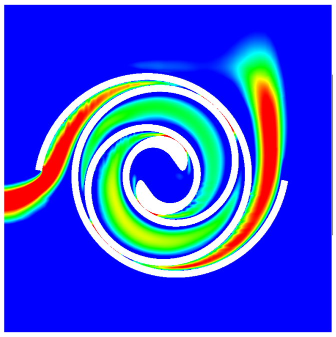

(b)

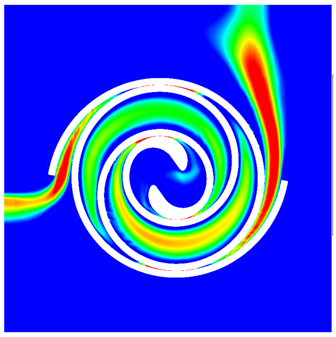

(c)

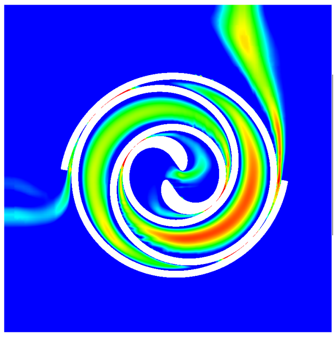

(d)

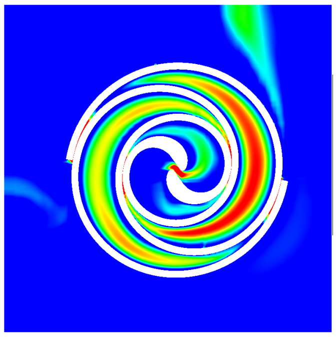

(e)


(f)


(g)


(h)


(i)


(j)

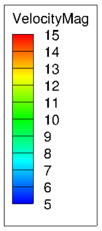


Figure 5.11: Laminar flow of a compressible fluid in a scroll: Velocity magnitude iso-area snapshots taken every 0.03 s during an operating cycle.

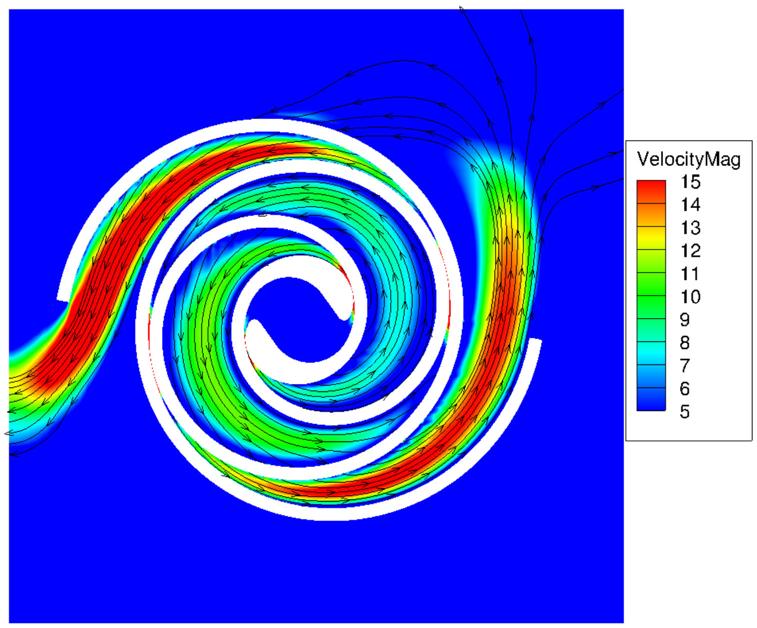


Figure 5.12: Laminar flow of a compressible fluid in a scroll: Velocity magnitude iso-areas and streamlines.

5.3 Incompressible Flow inside a Valveless Diaphragm Micropump

Diaphragm or membrane pumps are positive displacement pumps. They consist of the main chamber, an inlet and outlet duct, and a periodically moving diaphragm which is the passing flow's driving force. The inlet and outlet ducts might either be valveless diffusers or tubes of a constant cross-section with valves (valved pumps). Depending on the application, they are often preferred over bladed pumps since they are cheaper and can pump various fluids in a noiseless manner [49]. They are manufactured in large or small scales, with the large (usually valved) ones used for cleaning tank bottoms or pumping sewage, while the small (valved [320] or valveless [64]) ones (micropumps) mostly used as medical analysis devices [227], in biochemical-processing applications, or to deliver drugs to patients. In such cases, the valves are usually replaced by diffusers. Unfortunately, these pumps often suffer from undesirable back-flow at the exit during a percentage of their period, which can be reduced by adequately adjusting the diaphragm motion characteristics. This study is thoroughly analyzed in section 9.4.

The valveless micropump design studied in this section, fig. 5.13, firstly introduced by [292], is based on an existing micropump found in the literature [288]. Its length is 1 cm, and the chamber's length, height and volume are 8.862 mm, 0.5 mm and around 40 mm^3, respectively. The inlet and outlet diffusers are identical. The inlet cross-sectional area is 0.03 mm^2 and the outlet area is 0.2 mm^2. Its working principles are similar to the respiratory system of humans. The elastic diaphragm is deformed by a piezoelectric device causing its periodical motion, inducing fluid motion to the right. When the diaphragm moves upwards, the chamber volume increases, and higher mass flow enters the micropump from the inlet than from the outlet. Conversely, when the diaphragm moves down, the pressure is increased, driving the fluid to exit mainly from the outlet. The preferred flow direction is determined by the diffuser/nozzle elements design, which allows for a lower pressure loss in the diffuser than in the nozzle direction for the same flow velocity. Therefore, the net volume is pumped from the inlet to the outlet during a complete pump cycle, even though the diffuser/nozzle elements allow fluid motion in both directions [292], [254], [13].

In this study, the diaphragm does not follow the conventional motion found in most commercial micropumps. Its displacement is described by the mathematical model presented below, enabling the diaphragm to efficiently guide the incoming flow towards the outlet by suppressing or, at least, reducing the exit's undesirable back-flow phenomena at the exit. The diaphragm motion is parameterized using 8 design variables denoted as $b_i, i=0,7$. x and z axes correspond to the longitudinal and spanwise directions, respectively. The diaphragm motion takes place along the y-axis and the origin of the coordinate system is at the center of the rectangular diaphragm of size $L_x \times L_z$. $L_x^m = 0.9b_0L_x$ and $L_z^m = 0.9b_4L_z$ define the part of its area which is allowed to move. At each time step, the diaphragm is deformed in bell shape around $x_c = L_x^m(\frac{t}{T} - \frac{1}{2})$, $z_c = 0$ expressed as

$$y = -y_{max} f(\tau_x) f(\tau_z)$$

where

$$\begin{aligned} f(\tau) &= 6\tau^2 - 8\tau^3 + 3\tau^4 \\ y_{max} &= b_1 \left(1 - \left|1 - \frac{2t}{T}\right|\right) e^{-b_5\left(t-\frac{T}{2}\right)^2} \end{aligned}$$

where b_1 is the maximum displacement over all time steps achieved at the half period,

b_5 controls the function's abruptness and $T = 0.02s$ is the period. Every point located outside the neighborhood with center x_c and radius $Dx = b_3\left(\frac{L_x^m}{2} - |x_c|\right)$ remains stationary, while every point belonging to the neighborhood with radius $dx = b_2 Dx$ is displaced at y_{max}. For the rest points, transition is performed smoothly by using the polynomial mentioned above. This behavior is summarized in the τ_x definition which is given as

$$\tau_x = \begin{cases} 0 & , x \in [-\frac{L_x^m}{2}, x_s^1] \cup [x_e^2, \frac{L_x^m}{2}] \\ 1 & , x \in [x_s^2, x_e^1] \\ \frac{x - x_s^1}{x_s^2 - x_s^1} & , x \in [x_s^1, x_s^2] \\ \frac{x_e^2 - x}{x_e^2 - x_e^1} & , x \in [x_s^2, x_s^1] \end{cases}$$

where $x_s^1 = x_c - Dx$, $x_s^2 = x_c - dx$, $x_e^1 = x_c + dx$, $x_e^2 = x_c + Dx$. τ_z is defined similarly by setting $Dz = b_6\left(\frac{L_z^m}{2} - |z_c|\right)$ and $dz = b_7 D_z$. The exact parametric values are given in table 5.2.

b_0	0.79294
b_1	0.00045
b_2	0.066482
b_3	0.85741
b_4	0.64411
b_5	0.036511
b_6	0.67559
b_7	0.073943

Table 5.2: Laminar flow of an incompressible fluid inside a diaphragm pump: Values given to parameters controlling the diaphragm's motion studied in this section.

Having defined the micropump's shape and the diaphragm's motion, the unsteady CFD flow simulation follows. Fig. 5.14 shows the Cartesian mesh generated at the first time instant and at the diaphragm's maximum displacement. Mesh quality is maintained despite the extreme boundary displacement, illustrating the cut-cell method's advantage to handle such cases successfully. Only the half pump is simulated due to its symmetry along the $z = 0$ plane. Total pressure and axial velocity direction are imposed at the inlet and static pressure at the outlet, so as $v_{inv} = \sqrt{2(p_t^{in} - p^{out})/\rho} = 0.875\ m/s$, inducing a low mass flow rate through the pump. In case the flow instantaneously exits from the inlet, only total pressure is used as a boundary condition. After solving the flow equations for four successive periods consisting of 20 time steps each, the flow has become periodic. Fig. 5.15 shows velocity magnitude iso-areas and streamlines on the symmetry plane at 8 time instants over a single period. Back-flow at the last three time instants is evident. At some time steps, the diaphragm's displacement is large enough to cause the appearance or disappearance of around 2700 cells. Despite this significant change in mesh size, the mass flow deviation is less than 0.1%, verifying the software's ability to ensure mass conservation even when high boundary deformations occur.

Research presented in this section has been funded by the Business Plan "Development of Human Resources, Education and Lifelong Learning" entitled "Support Researchers with Emphasis on Young Researchers" with the co-financing of Greece and the European Union. The project title is "*Design-Optimization of Diaphragm Pumps under Operational/Manufacturing Uncertainties using the cut-cell Method and Polynomial Chaos Expansion*".

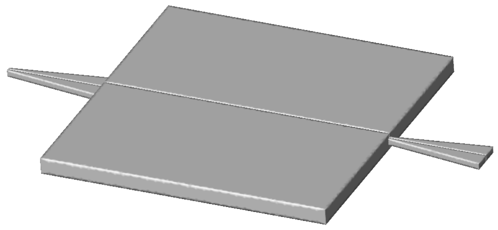


Figure 5.13: Laminar flow of an incompressible fluid inside a diaphragm pump: Micropump's geometry. Flow enters from the left and exits from the right diffuser. The diaphragm is placed on the chamber's upper surface.

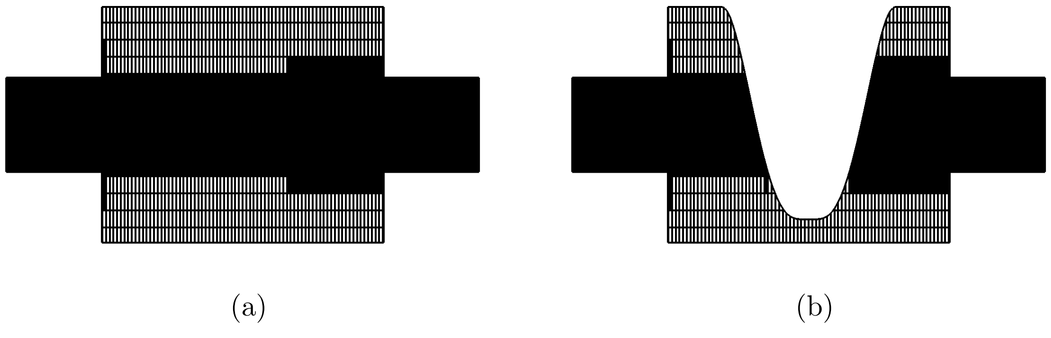


Figure 5.14: Laminar flow of an incompressible fluid inside a diaphragm pump: Cartesian mesh inside the pump (a) at the initial time instant and (b) at the half of the motion's period in which the displacement is maximum. Axes not in scale.

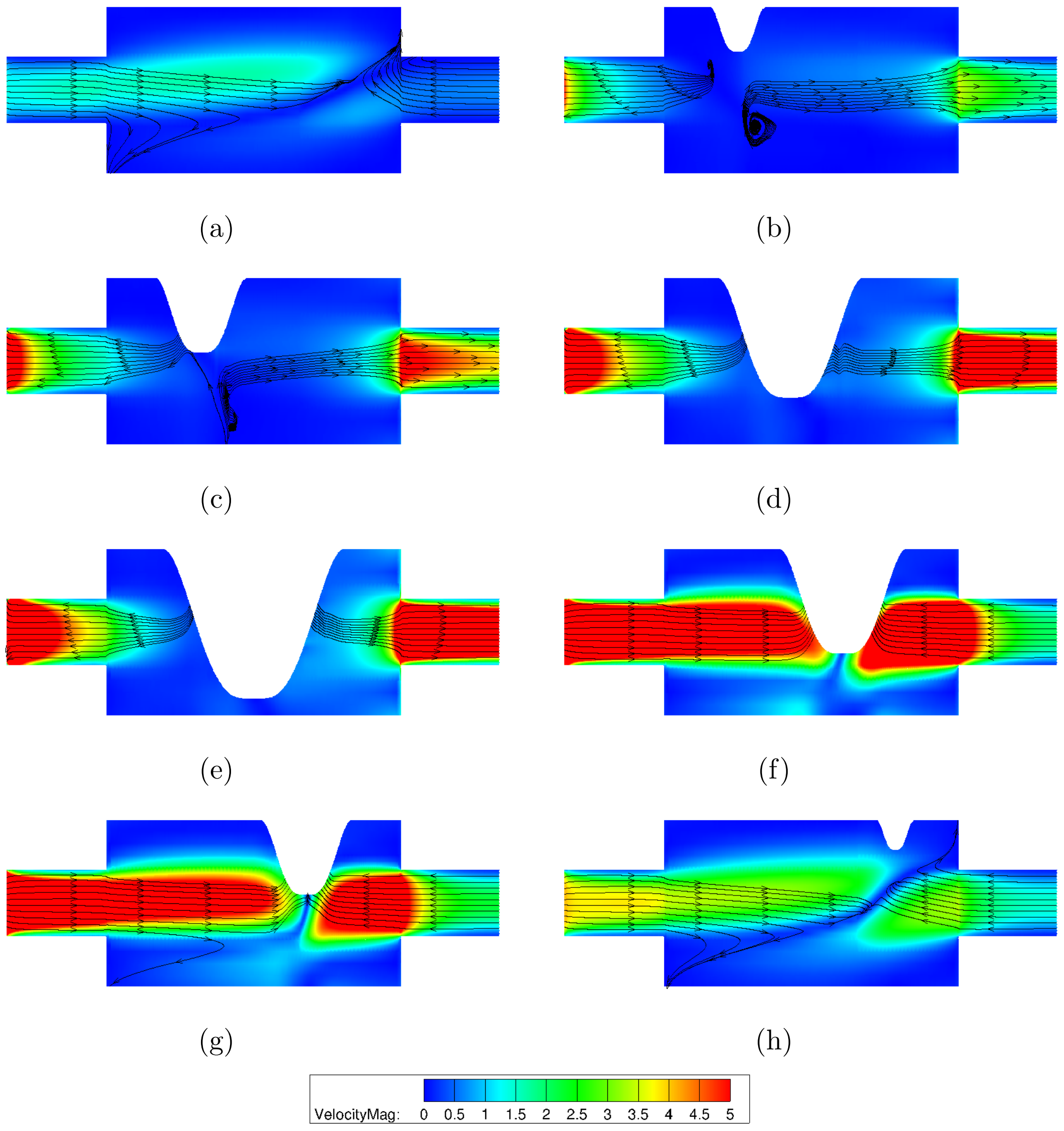


Figure 5.15: Laminar flow of an incompressible fluid inside a diaphragm pump: Velocity iso-areas and streamlines at 8 time instants within a single period of the periodic flow. Axes not in scale.

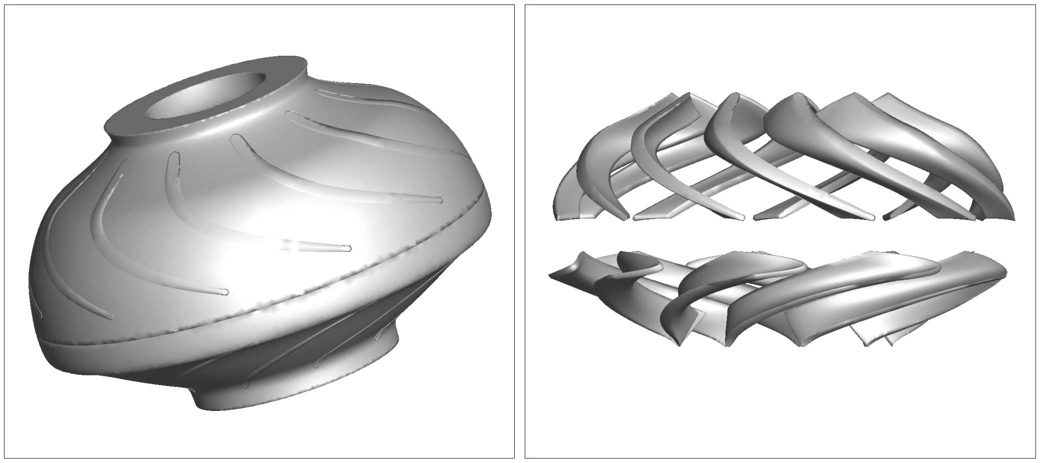
5.4 Compressible Flow inside an Electrical Submersible Pump Stage

The Electrical Submersible Pump (ESP) is an efficient and reliable method for extracting moderate to high volumes of fluids from wellbores. Its operation is a subject of great importance, especially in the oil industry. More than 90% of the worldwide oil-producing wells require an artificial lift to increase the flow from wells when a reservoir has no longer sufficient energy to induce flow towards the ground [171]. ESPs are one of the most versatile and efficient artificial lift methods. They comprise multiple centrifugal pump stages positioned in a series within a proper induction motor that can achieve rotational speeds of more than 5000 *rpm* [324]. Each stage consists of an impeller and a bladed diffuser. At each stage, the fluid's total pressure slightly increases until the exit of the multistage arrangement, where the fluid should have gained enough energy to travel through the well until its exit.

In this section, the numerical simulation of a compressible fluid within an EPS is presented. CFD is a key feature in the understanding of the complicated flow phenomena developed inside an ESP. During the last decades, considerable research has been conducted on pump stages, based almost exclusively on body-fitted meshes. Their use is accompanied by some difficulties arising from the relative motion between the impeller and the diffuser. This challenge is by-passed by employing the Multiple Reference Frame (MRF) technique [346], in which a steady-state solver simulates the flow within the impeller concerning a relative reference frame and adding proper source terms to the momentum equations simulating the rotating motion within a non-rotating mesh. Another widely used method is the comparatively more accurate Sliding Mesh technique [347], according to which the mesh generated for the impeller is rotated together with its blades, bringing the simulation closer to the real-world scenario.However, data must be exchanged through the non-matching interfaces between the rotating impeller's and stationary diffuser's mesh domains.

This work introduces the cut-cell method as a powerful alternative that overcomes the impeller-diffuser interface problems without using the techniques mentioned above. Although most papers in the literature on this kind of pumps deal with multiphase flows [52], this study makes the single-phase inviscid flow assumption, focusing more on the successful implementation of the developed cut-cell method. The study is part of a project funded by Schlumberger Cambridge Research Limited dealing with the optimization of a commercial EPS. The company also provided the pump's geometry and operating conditions. Fig. 5.16 shows the pump's casing and the impeller and diffuser blades. The flow enters from its bottom and exits from the top.

A single Cartesian mesh is generated around the impeller and the diffuser. In the impeller's vicinity, cells are combined or break anew as the mesh refinement follows the blades' rotation. The part of the mesh close to the diffuser remains unchanged. The impeller's blades are attached to the casing, the bottom half of which rotates together with the blades. Consequently, the casing consists of the rotating (bottom) and the stationary (top) parts. Their interface comprises two concentric circles, where the outer one is shown in fig. 5.16a. At each time step, edges of the casing's triangulated surface mesh that connect one node positioned at the interface with another one belonging to the rotating part are deleted. Then, the casing's bottom half is rotated by a small angle, and new edges are created connecting its two separated parts. Three periods were simulated. After the second one, periodicity has been established. Figs. 5.17a and 5.17b show velocity magnitude iso-areas on a slice perpendicular to the axis and streamlines, indicating the flow direction through the impeller and the diffuser, respectively. Slices are located as in fig. 5.18. Finally, figs. 5.19 and 5.20 show four time instants each, at an attempt to visualize the transient flow inside the impeller and the diffuser, before the flow becomes periodic.



(a) (b)

Figure 5.16: Inviscid flow of a compressible fluid inside an EPS stage: (a) casing and (b) impeller's and diffuser's blades of an ESP's stage.

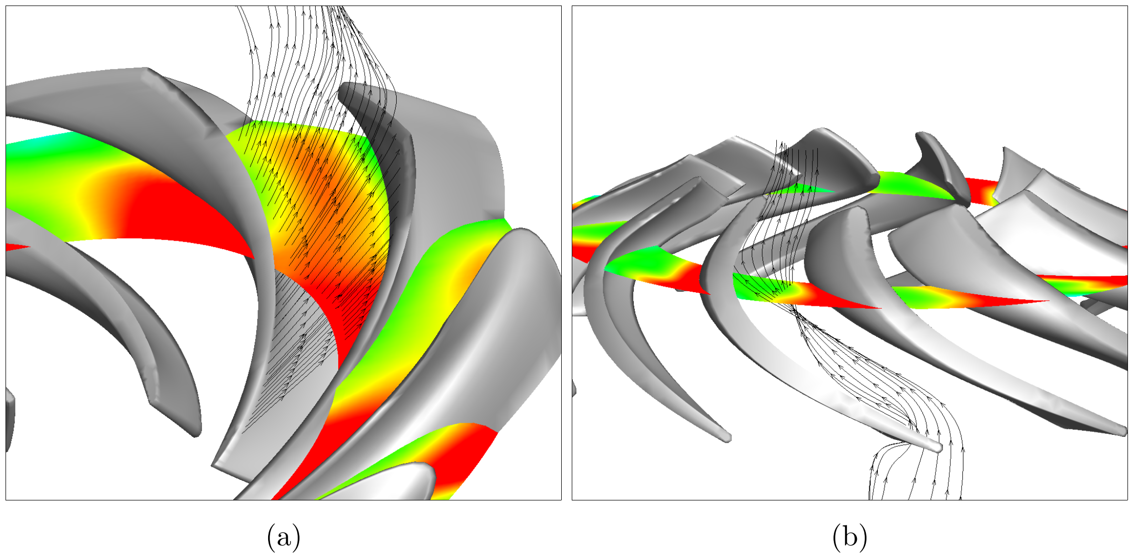


Figure 5.17: Inviscid flow of a compressible fluid inside an EPS stage: Streamlines and velocity magnitude iso-areas in (a) the impeller and (b) the diffuser. The slices' position is depicted in fig. 5.18.

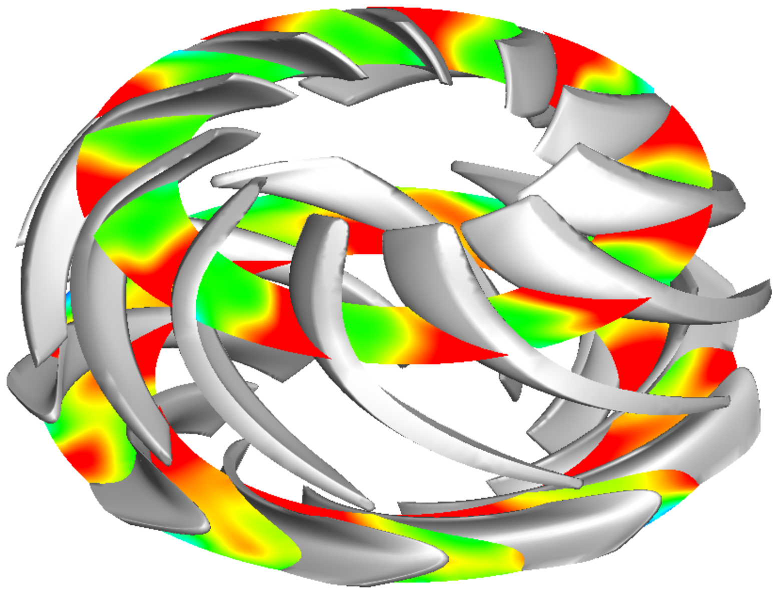


Figure 5.18: Inviscid flow of a compressible fluid inside an EPS stage: Impeller and diffuser slices, perpendicular to the axis. Velocity magnitude iso-areas are shown. Slices' close-up views are presented in fig. 5.17.

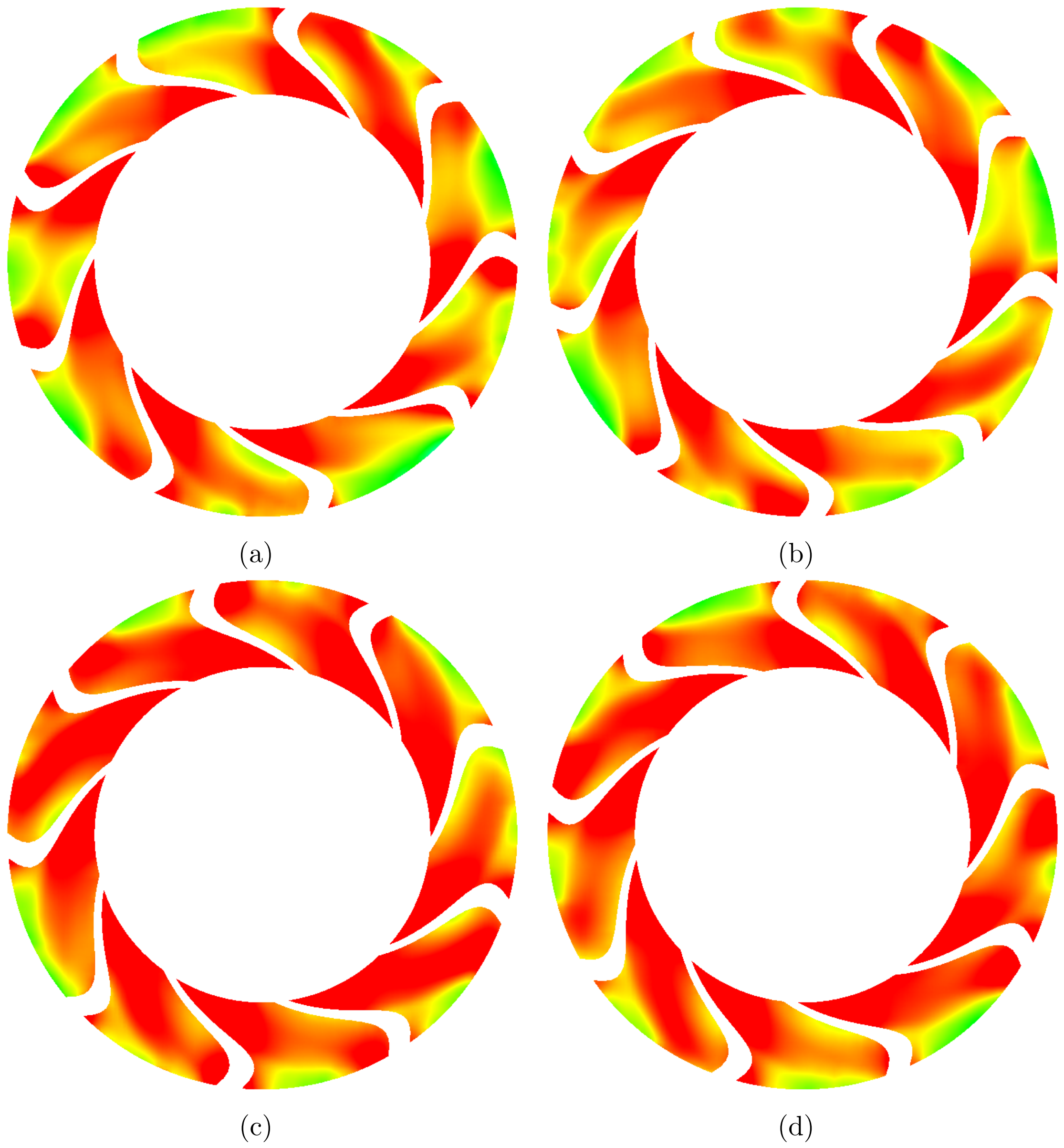


Figure 5.19: Inviscid flow of a compressible fluid inside an EPS stage: Velocity iso-areas on an impeller slice during the transient phenomenon computed in the pump's second operating period.

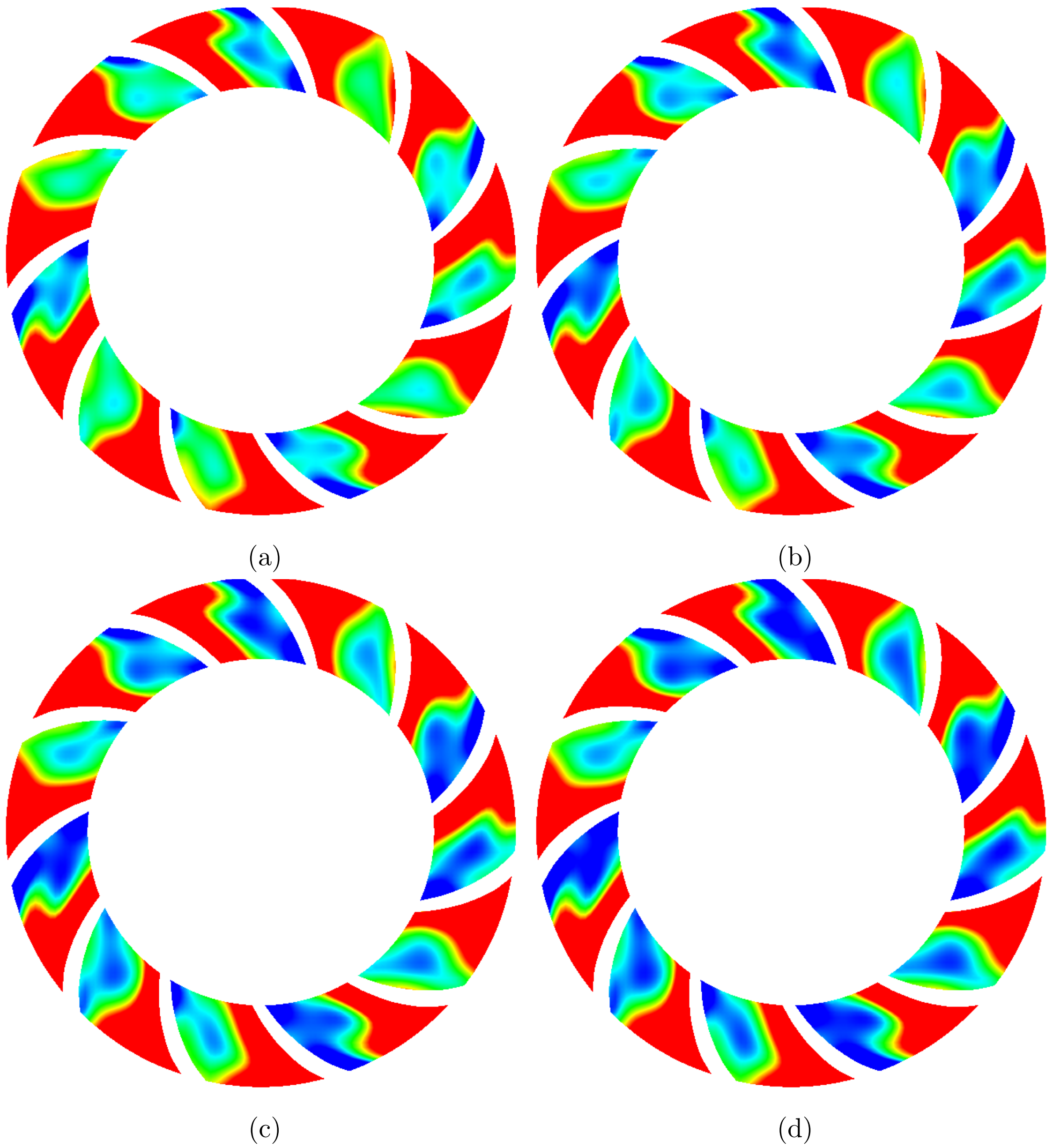


Figure 5.20: Inviscid flow of a compressible fluid inside an EPS stage: Velocity iso-areas on a diffuser slice during the transient phenomenon computed during the pump's second operating period. Time instants coincide with those shown in fig. 5.19.

Chapter 6

The Continuous Adjoint Method

This chapter is concerned with the mathematical development of numerical methods capable of solving shape optimization problems for academic and industrial flow applications. Typical examples are drag minimization in external aerodynamics and total pressure losses minimization in internal aerodynamics. From a mathematical perspective, an optimization method aims at maximizing or minimizing one or more targets called objective functions (F). This thesis deals only with single-target optimization problems. F is a function of N independent variables called design variables ($b_q,\ q = 1, \cdots, N$). In all optimization cases presented in this work, the design variables control the solid bodies' shape. They can be the coordinates of the control points of a shape parameterization tool (likely based on Bézier–Bernstein polynomials, Splines or NURBS) or nodal coordinates on the solid bodies' discretized surface.

This chapter focuses on computing the objective function's gradient to support shape optimization algorithms. Therefore the continuous adjoint method is developed for compressible and incompressible flows implemented by the cut-cell or the ghost-cell method. A discussion is also made about the adjoint Riemann problem definition and the discretization of the field adjoint equations. Thereafter, the unsteady method's variant is studied, and solutions to data storage problems are proposed. Readers interested in the discrete adjoint method are referred to chapter 7.

6.1 Mathematical Development of the Compressible Adjoint Method

This section presents the mathematical formulation of the continuous adjoint method for 3D compressible steady or unsteady, inviscid and laminar flows. The adjoint PDEs accompanied by the proper boundary conditions are defined, and the final expression of the objective's gradient w.r.t. $\vec{b}$ is given. Hereafter, this gradient noted as $\delta F/\delta b_q$ will be referred to as the sensitivity derivatives. Firstly, the more general unsteady laminar case is studied, and then, the resulting formulas are specified in steady or inviscid flows. In the following development, an assumption is made about the indices range. Specifically, indices k, l and m correspond to the 3 space dimensions, while h, i and j vary from 1 to the number of the governing equations. Index r is 1 or 2, and q is used only for the design variables enumeration. Flow quantities notation follows the definitions made in section 3.1.

6.1.1 Definition of the Total Derivative

Consider a parameterized flow domain Ω and its boundary S. An alteration in $\vec{b}$ causes the boundary displacement, which pushes each internal point, defined by its parametric coordinates, changing their Cartesian coordinates ($\vec{x}=\vec{x}(\vec{b})$). However, the boundary deformation also affects the flow variables stored at each point, allowing for the expression of any field quantity $\Phi(\vec{b},\vec{x})$ (e.g., the governing equation R_i) as a function of $\vec{b}$. Hence,

$$\Phi = \Phi\left(\vec{b},\vec{x}(\vec{b})\right)$$

In unsteady flows, the design variables are considered independent of time. When solid bodies are allowed to move, Φ is expressed as

$$\Phi = \Phi\left(\vec{b},t,\vec{x}(\vec{b},t)\right)$$

Its total derivative w.r.t. b_q and t is

$$\begin{aligned}\frac{\delta\Phi}{\delta b_q} &= \frac{\partial\Phi}{\partial b_q} + \frac{\partial\Phi}{\partial x_k}v_k^s\\ \frac{\delta\Phi}{\delta t} &= \frac{\partial\Phi}{\partial t} + \frac{\partial\Phi}{\partial x_k}v_k^g\end{aligned}$$

where $v_k^s = \partial x_k / \partial b_q$ is the surface deformation rate during the optimization process and $v_k^g = \partial x_k / \partial t$ is the surface velocity during the unsteady phenomenon. Their normal components will be denoted as $v_n^s = v_k^s n_k$ and $v_n^g = v_k^g n_k$. It follows that $\delta\Phi/\delta b_q$ is a combination of the partial derivative $\partial\Phi/\partial b_q$ representing its variation caused exclusively by changes in the flow variables due to the surface modification and, the term describing its change due to the point's displacement. Furthermore, the total derivative $\delta\Phi/\delta t$ is equivalent to the material derivative used in the Lagrangian description of fluids' motion. Since the partial derivative is independent of the Cartesian coordinates variation [237], [159],

$$\frac{\partial}{\partial b_q}\left(\frac{\partial \Phi}{\partial x_k}\right) = \frac{\partial}{\partial x_k}\left(\frac{\partial \Phi}{\partial b_q}\right)$$

As mentioned before, design variables are independent of time, which allows for the corresponding derivatives permutation,

$$\frac{\partial}{\partial b_q}\left(\frac{\partial \Phi}{\partial t}\right) = \frac{\partial}{\partial t}\left(\frac{\partial \Phi}{\partial b_q}\right)$$

A useful tool from the calculus of moving surfaces [119] is also introduced. The instantaneous rate of change of Φ in the normal direction of a moving surface is

$$\frac{\delta_s \Phi}{\delta_s b_q} = \frac{\partial \Phi}{\partial b_q} + \frac{\partial \Phi}{\partial n} v_n^s$$

which was originally defined by J. Hadamard. The term $\partial\Phi/\partial n = \partial\Phi/\partial x_k n_k$ represents the normal to the surface derivative. The δ_s derivative is applied only to points on the surface and differs from the δ derivative used only in the interior of the domain. The aforementioned derivatives play a central role in the differentiation of volume and surface integrals,

$$\begin{aligned} \frac{\delta}{\delta b_q}\int_\Omega \Phi d\Omega &= \int_\Omega \frac{\partial \Phi}{\partial b_q} d\Omega + \int_S \Phi v_n^s dS \\ \frac{\delta}{\delta b_q}\int_S \Phi dS &= \int_S \frac{\delta_s \Phi}{\delta_s b_q} dS - \int_S \Phi H v_n^s dS \end{aligned} \tag{6.1}$$

where H is the surface mean curvature. The first equation is the Reynolds transport theorem, and the second one was firstly published in [118] and assumes that S is closed or its contour is independent of $\vec{b}$. In both equations, the first integral on the r.h.s. corresponds to the rate of change of Φ and the second one represents the contribution of the surface motion. Based on the Hadamard's derivative, the surface differentiation is alternatively expressed as

$$\frac{\delta}{\delta b_q}\int_S \Phi dS = \int_S \frac{\partial \Phi}{\partial b_q} dS + \int_S \left(\frac{\partial \Phi}{\partial n} - \Phi H\right) v_n^s dS$$

6.1.2 Differentiation of the Objective Function

This thesis focuses on objective functions given in a volume or surface integral form. In a general unsteady case,

$$F = \int_{T_F}\int_{\Omega_F} F_\Omega d\Omega dt + \int_{T_F}\int_{S_F} F_{S_k} n_k dS dt$$

The time window T_F is equal to or part of the flow simulation's period. For example, in periodic flows, the governing equations are solved until periodicity is established. However, the design optimization usually focuses only on the periodic part, and thus, T_F is chosen equal to the last period of the simulation. Moreover, the following mathematical development assumes that $\Omega_F = \Omega$. The surface S_F is considered part of the boundary of the fluid domain. In applications studied in this thesis, the integrand F_Ω is a function of the conservative variables, and F_{S_k} is expressed w.r.t. velocity, pressure, temperature, and the stress force ($\tau_{lk}n_k$) exerted from the fluid to S_F. It is helpful to project the force onto the orthonormal basis $(\vec{n}, \vec{t}^1, \vec{t}^2)$, where $\vec{n}$, $\vec{t}^1$ and $\vec{t}^2$ are the local normal and tangent vectors to S_F,

$$\tau_{lk} n_k = \tau^n n_l + \tau^{t^r} t_l^r \tag{6.2}$$

where

$$\tau^n = \tau_{lk} n_k n_l$$
$$\tau^{t^r} = \tau_{lk} n_k t_l^r$$

Therefore,

$$\begin{aligned} F_\Omega &= F_\Omega(\vec{U}) \\ F_{S_k} &= F_{S_k}(\vec{v}, p, T, \tau^n, \tau^{t^r}) \end{aligned} \tag{6.3}$$

If F_{S_k} is defined along the wall in a viscous case, its dependency from $\vec{v}$ is replaced by $\vec{v}^g$. Attention should be paid to the way F depends on $\vec{b}$. If a design variable changes, the geometry's shape changes as well, causing an alteration in the flow variables, which should always satisfy the governing equations. Consequently, the dependency of F is direct through the geometrical terms included in its mathematical expression and indirect through the flow variables it depends on.

Based on theorems 6.1 the total derivative of F is

$$\begin{aligned} \frac{\delta F}{\delta b_q} =& \int_{T_F}\int_\Omega \frac{\partial F_\Omega}{\partial b_q} d\Omega dt + \int_{T_F}\int_{S_w} F_\Omega v_n^s dSdt \\ &+ \int_{T_F}\int_{S_F} \frac{\partial F_{S_k}}{\partial b_q} n_k dSdt + \int_{T_F}\int_{S_w} \left(\frac{\partial F_{S_k}}{\partial n} - F_{S_k} H\right) v_n^s n_k dSdt \\ &+ \int_{T_F}\int_{S_w} F_{S_k} \frac{\delta_s n_k}{\delta_s b_q} dSdt \end{aligned}$$

During the optimization only the solid boundary is modified. Therefore, $\delta_s n_k/\delta_s b_q = 0$ and $v_n^s = 0$ at the inlet or outlet, and terms containing these quantities are integrated only along S_w. Considering the objective function's dependencies, eqs. 6.3, the gradient becomes

$$\begin{aligned} \frac{\delta F}{\delta b_q} =& \int_{T_F}\int_\Omega \frac{\partial F_\Omega}{\partial U_j}\frac{\partial U_j}{\partial b_q} d\Omega dt + \int_{T_F}\int_{S_w} F_\Omega v_n^s dS + \int_{T_F}\int_{S_F} \frac{\partial F_{S_k}}{\partial v_m}\frac{\partial v_m}{\partial b_q} n_k dSdt \\ &+ \int_{T_F}\int_{S_F} \frac{\partial F_{S_k}}{\partial p}\frac{\partial p}{\partial b_q} n_k dSdt + \int_{T_F}\int_{S_F} \frac{\partial F_{S_k}}{\partial T}\frac{\partial T}{\partial b_q} n_k dSdt \\ &+ \int_{T_F}\int_{S_F} \frac{\partial F_{S_k}}{\partial \tau^n}\frac{\partial \tau^n}{\partial b_q} n_k dSdt + \int_{T_F}\int_{S_F} \frac{\partial F_{S_k}}{\partial \tau^{t^r}}\frac{\partial \tau^{t^r}}{\partial b_q} n_k dSdt \\ &+ \int_{T_F}\int_{S_w} \left(\frac{\partial F_{S_k}}{\partial n} - F_{S_k} H\right) v_n^s n_k dSdt + \int_{T_F}\int_{S_w} F_{S_k} \frac{\delta_s n_k}{\delta_s b_q} dSdt \end{aligned} \tag{6.4}$$

As expected, the gradient of F contains derivatives of flow quantities w.r.t. the design variables. A straightforward approach for their computation is the direct differentiation method, according to which the governing equations and boundary conditions are differentiated w.r.t. each b_q. Their numerical solution provides the

$\partial U_i/\partial b_q$ values by the cost of N equivalent flow simulations, which is unfeasible for industrial applications where N can be very high. This cost is drastically decreased to one equivalent flow solution using the adjoint method, which provides an alternative expression for $\delta F/\delta b_q$ calculation independent of $\partial U_i/\partial b_q$. This method is explained in subsection 6.1.3.

6.1.3 Definition of the Augmented Function

Adjoint theory's central concept is the optimization problem perception as a constrained one, where F should be maximized or minimized subject to the constraint of satisfying the flow equations ($\vec{R}=\vec{0}$). Inspired by the method of Lagrange multipliers, a Lagrangian or augmented function is introduced as

$$L = F + \int_{T_R}\int_{\Omega} \Psi_i R_i d\Omega dt$$

where $\vec{\Psi}$ are the adjoint or co-state variables, $\vec{R}$ stands for the unsteady flow equations and T_R is the flow simulation duration. The adjoint variant for steady flow problems will emerge as a particular case of the unsteady one in subsection 6.1.11. The variable T_R represents the time window from t_s to t_e at which the unsteady flow phenomenon is studied. The space-time integral is zero since it contains the residuals of the governing equations. Therefore, $L=F$. Differentiation of L w.r.t. b_q yields

$$\frac{\delta L}{\delta b_q} = \frac{\delta F}{\delta b_q} + \int_{T_R}\int_{\Omega} \Psi_i \frac{\partial R_i}{\partial b_q} d\Omega dt + \int_{T_R}\int_{S} \Psi_i R_i v_n^s dS dt$$

where the Reynolds transport theorem is used.

The target of the following mathematical development is to differentiate $\vec{R}$ and eliminate the resulting volume integrals containing the $\partial U_i/\partial b_q$ term by factorizing all $\partial U_i/\partial b_q$ derivatives and nullifying their multipliers, giving rise to the adjoint PDEs. The same method applied to the surface integrals leads to the introduction of the corresponding adjoint boundary conditions. In the following development, volume integrals noted as FAE contribute to the field adjoint equations formulation. Similarly, all surface integrals noted as ABC will be used for the adjoint boundary conditions definition. Finally, integrals annotated as SD contain geometrical variations and are part of the final sensitivity derivatives expression.

By using eqs. 6.4 and 3.1 one obtains

$$
\begin{aligned}
\frac{\delta L}{\delta b_q} = & \underbrace{\int_{T_F}\int_{\Omega} \frac{\partial F_\Omega}{\partial U_j}\frac{\partial U_j}{\partial b_q} d\Omega dt}_{FAE} + \underbrace{\int_{T_F}\int_{S_w} F_\Omega v_n^s dS dt}_{SD} + \underbrace{\int_{T_F}\int_{S_F} \frac{\partial F_{S_k}}{\partial v_m}\frac{\partial v_m}{\partial b_q} n_k dS dt}_{ABC/SD} \\
& + \underbrace{\int_{T_F}\int_{S_F} \frac{\partial F_{S_k}}{\partial p}\frac{\partial p}{\partial b_q} n_k dS dt}_{ABC} + \underbrace{\int_{T_F}\int_{S_F} \frac{\partial F_{S_k}}{\partial T}\frac{\partial T}{\partial b_q} n_k dS dt}_{ABC} + \underbrace{\int_{T_F}\int_{S_F} \frac{\partial F_{S_k}}{\partial \tau^n}\frac{\partial \tau^n}{\partial b_q} n_k dS dt}_{ABC} \\
& + \underbrace{\int_{T_F}\int_{S_F} \frac{\partial F_{S_k}}{\partial \tau^{t^r}}\frac{\partial \tau^{t^r}}{\partial b_q} n_k dS dt}_{ABC} + \underbrace{\int_{T_F}\int_{S_w} \left(\frac{\partial F_{S_k}}{\partial n} - F_{S_k} H\right) v_n^s n_k dS dt}_{SD} \\
& + \underbrace{\int_{T_F}\int_{S_w} F_{S_k}\frac{\delta_s n_k}{\delta_s b_q} dS dt}_{SD} + \underbrace{\int_{T_R}\int_{\Omega} \Psi_i \frac{\partial}{\partial b_q}\left(\frac{\partial U_i}{\partial t}\right) d\Omega dt}_{I^{temp}} \\
& + \underbrace{\int_{T_R}\int_{\Omega} \Psi_i \frac{\partial}{\partial b_q}\left(\frac{\partial f_{ik}^{inv}}{\partial x_k}\right) d\Omega dt}_{I^{inv}} - \underbrace{\int_{T_R}\int_{\Omega} \Psi_i \frac{\partial}{\partial b_q}\left(\frac{\partial f_{ik}^{vis}}{\partial x_k}\right) d\Omega dt}_{I^{vis}}
\end{aligned}
\tag{6.5}
$$

If the inlet or outlet boundaries are part of S_F, the under-braced integral with ABC/SD contributes to the inlet/outlet boundary conditions. On the other hand, in case S_F is defined only along the solid wall, the integral contributes to the sensitivity derivatives expression. Integrals I^{temp}, I^{inv} and I^{vis} are processed in subsections 6.1.4, 6.1.5, and 6.1.6, respectively.

6.1.4 Differentiation of the Temporal Term

The I^{temp} integral of eq. 6.5 is developed as

$$
\begin{aligned}
I^{temp} &= \int_{T_R}\int_{\Omega} \Psi_i \frac{\partial}{\partial t}\left(\frac{\partial U_i}{\partial b_q}\right) d\Omega dt = \int_{T_R}\int_{\Omega} \frac{\partial}{\partial t}\left(\Psi_i \frac{\partial U_i}{\partial b_q}\right) d\Omega dt - \int_{T_R}\int_{\Omega} \frac{\partial \Psi_i}{\partial t}\frac{\partial U_i}{\partial b_q} d\Omega dt \\
&= \int_{T_R} \frac{\delta}{\delta t}\int_{\Omega} \Psi_i \frac{\partial U_i}{\partial b_q} d\Omega dt - \int_{T_R}\int_{S_w} \Psi_i \frac{\partial U_i}{\partial b_q} v_n^g dS dt - \int_{T_R}\int_{\Omega} \frac{\partial \Psi_i}{\partial t}\frac{\partial U_i}{\partial b_q} d\Omega dt \\
&= \underbrace{\left[\int_{\Omega} \Psi_i \frac{\partial U_i}{\partial b_q} d\Omega\right]_{t_s}^{t_e}}_{ABC} - \underbrace{\int_{T_R}\int_{S_w} \Psi_i \frac{\partial U_i}{\partial b_q} v_n^g dS dt}_{ABC} - \underbrace{\int_{T_R}\int_{\Omega} \frac{\partial \Psi_i}{\partial t}\frac{\partial U_i}{\partial b_q} d\Omega dt}_{FAE}
\end{aligned}
\tag{6.6}
$$

6.1.5 Differentiation of the Convection Term

The I^{inv} integral of eq. 6.5 is developed as

$$I^{inv} = \int_{T_R}\int_{\Omega} \Psi_i \frac{\partial}{\partial x_k}\left(\frac{\partial f_{ik}^{inv}}{\partial b_q}\right) d\Omega dt = \int_{T_R}\int_{S} \Psi_i \frac{\partial f_{ik}^{inv}}{\partial b_q} n_k dSdt - \int_{T_R}\int_{\Omega} \frac{\partial \Psi_i}{\partial x_k}\frac{\partial f_{ik}^{inv}}{\partial b_q} d\Omega dt$$

where the Green-Gauss theorem is used. Since

$$\frac{\partial f_{ik}^{inv}}{\partial b_q} = \frac{\partial f_{ik}^{inv}}{\partial U_j}\frac{\partial U_j}{\partial b_q} = A_{ijk}\frac{\partial U_j}{\partial b_q}$$

the inviscid integral is expressed as

$$I^{inv} = \int_{T_R}\int_{S} \Psi_i \frac{\partial f_{ik}^{inv}}{\partial b_q} n_k dSdt - \int_{T_R}\int_{\Omega} \frac{\partial \Psi_i}{\partial x_k} A_{ijk}\frac{\partial U_j}{\partial b_q} d\Omega dt$$

Boundary S consists of the inlet and outlet parts (S_{IO}) and walls (S_w). The surface integral is split into these parts, which are treated differently,

$$\begin{aligned} I^{inv} = & \underbrace{\int_{T_R}\int_{S_w} \Psi_i \frac{\partial f_{ik}^{inv}}{\partial b_q} n_k dSdt}_{I_S^{inv}} + \underbrace{\int_{T_R}\int_{S_{IO}} \Psi_i A_{ijk}\frac{\partial U_j}{\partial b_q} n_k dSdt}_{ABC} \\ & - \underbrace{\int_{T_R}\int_{\Omega} \frac{\partial \Psi_i}{\partial x_k} A_{ijk}\frac{\partial U_j}{\partial b_q} d\Omega dt}_{FAE} \end{aligned} \tag{6.7}$$

Considering that only the solid boundary is modified during the optimization, $\vec{v}^s = \vec{0}$ and $\delta_s n_k/\delta_s b_q = 0$ along S_{IO}. Under these conditions, term I_S^{inv} becomes

$$\begin{aligned} I_S^{inv} &= \int_{T_R}\int_{S_w} \Psi_i \frac{\partial f_{ik}^{inv}}{\partial b_q} n_k dSdt = \int_{T_R}\int_{S_w} \Psi_i \left(\frac{\delta_s f_{ik}^{inv}}{\delta_s b_q} - \frac{\partial f_{ik}^{inv}}{\partial n} v_n^s\right) n_k dSdt \\ &= \int_{T_R}\int_{S_w} \Psi_i \frac{\delta_s f_{ik}^{inv}}{\delta_s b_q} n_k dSdt - \int_{T_R}\int_{S_w} \Psi_i \frac{\partial f_{ik}^{inv}}{\partial n} v_n^s n_k dSdt \\ &= \int_{T_R}\int_{S_w} \Psi_i \frac{\delta_s (f_{ik}^{inv} n_k)}{\delta_s b_q} dSdt - \int_{T_R}\int_{S_w} \Psi_i f_{ik}^{inv} \frac{\delta_s n_k}{\delta_s b_q} dSdt \\ &- \int_{T_R}\int_{S_w} \Psi_i \frac{\partial f_{ik}^{inv}}{\partial n} v_n^s n_k dSdt \end{aligned}$$

By using the wall flux expression, eq. 3.22,

$$\vec{f}_k^{inv,w} n_k^w = \begin{bmatrix} \rho^w v_n^g \\ \rho^w v_1^w v_n^g + p^w n_1^w \\ \rho^w v_2^w v_n^g + p^w n_2^w \\ \rho^w v_3^w v_n^g + p^w n_3^w \\ (\rho^w E^w + p^w) v_n^g \end{bmatrix} = \vec{U}^w v_n^g + p^w \begin{bmatrix} 0 \\ n_1^w \\ n_2^w \\ n_3^w \\ v_n^g \end{bmatrix}$$

the integral term becomes

$$\begin{aligned} I_S^{inv} = &\int_{T_R}\int_{S_w} \Psi_i \frac{\delta_s(U_i v_n^g)}{\delta_s b_q} dSdt + \int_{T_R}\int_{S_w} \Psi_{k+1} \frac{\delta_s(p n_k)}{\delta_s b_q} dSdt + \int_{T_R}\int_{S_w} \Psi_E \frac{\delta_s(p v_n^g)}{\delta_s b_q} dSdt \\ &- \int_{T_R}\int_{S_w} \Psi_i f_{ik}^{inv} \frac{\delta_s n_k}{\delta_s b_q} dSdt - \int_{T_R}\int_{S_w} \Psi_i \frac{\partial f_{ik}^{inv}}{\partial n} v_n^s n_k dSdt \end{aligned}$$

where $\Psi_E = \Psi_5$ is the adjoint variable corresponding to the energy equation. Geometry's normal velocity v_n^g may dependent on b_q (e.g., application 5.3). Thus, the above expression is written as

$$\begin{aligned} I_S^{inv} = &\underbrace{\int_{T_R}\int_{S_w} \left(\Psi_i v_n^g \frac{\partial U_i}{\partial b_q} + (\Psi_{k+1} n_k + \Psi_E v_n^g) \frac{\partial p}{\partial b_q} \right) dSdt}_{ABC} \\ &+ \underbrace{\int_{T_R}\int_{S_w} \left(\Psi_i v_n^g \frac{\partial U_i}{\partial n} + (\Psi_{k+1} n_k + \Psi_E v_n^g) \frac{\partial p}{\partial n} \right) v_n^s dSdt}_{SD} \\ &+ \underbrace{\int_{T_R}\int_{S_w} \left(\Psi_{k+1} p \frac{\delta_s n_k}{\delta_s b_q} + (\Psi_i U_i + \Psi_E p) \frac{\delta_s v_n^g}{\delta_s b_q} \right) dSdt}_{SD} \\ &- \underbrace{\int_{T_R}\int_{S_w} \Psi_i f_{ik}^{inv} \frac{\delta_s n_k}{\delta_s b_q} dSdt}_{SD} - \underbrace{\int_{T_R}\int_{S_w} \Psi_i \frac{\partial f_{ik}^{inv}}{\partial n} v_n^s n_k dSdt}_{SD} \end{aligned} \quad (6.8)$$

6.1.6 Differentiation of the Diffusion Term

A process similar to subsection 6.1.5 is followed for the viscous term,

$$\begin{aligned}
I^{vis} &= \int_{T_R}\int_{\Omega}\Psi_i\frac{\partial}{\partial x_k}\left(\frac{\partial f_{ik}^{vis}}{\partial b_q}\right)d\Omega dt = \int_{T_R}\int_{S}\Psi_i\frac{\partial f_{ik}^{vis}}{\partial b_q}n_k dSdt - \int_{T_R}\int_{\Omega}\frac{\partial \Psi_i}{\partial x_k}\frac{\partial f_{ik}^{vis}}{\partial b_q}d\Omega dt \\
&= \underbrace{\int_{T_R}\int_{S_w}\Psi_i\frac{\delta_s(f_{ik}^{vis}n_k)}{\delta_s b_q}dSdt}_{I_S^{vis}} - \underbrace{\int_{T_R}\int_{S_w}\Psi_i f_{ik}^{vis}\frac{\delta_s n_k}{\delta_s b_q}dSdt}_{SD} \\
&- \underbrace{\int_{T_R}\int_{S_w}\Psi_i\frac{\partial f_{ik}^{vis}}{\partial n}v_n^s n_k dSdt}_{SD} - \underbrace{\int_{T_R}\int_{\Omega}\frac{\partial \Psi_i}{\partial x_k}\frac{\partial f_{ik}^{vis}}{\partial b_q}d\Omega dt}_{I_\Omega^{vis}}
\end{aligned} \tag{6.9}$$

where the surface integral along S_{IO} containing the variation of the viscous terms is neglected. The viscous volume term reads

$$\begin{aligned}
I_\Omega^{vis} &= \int_{T_R}\int_{\Omega}\frac{\partial \Psi_{l+1}}{\partial x_k}\frac{\partial \tau_{lk}}{\partial b_q}d\Omega dt + \int_{T_R}\int_{\Omega}\frac{\partial \Psi_E}{\partial x_k}\frac{\partial (v_l\tau_{lk})}{\partial b_q}d\Omega dt + \int_{T_R}\int_{\Omega}\frac{\partial \Psi_E}{\partial x_k}\frac{\partial q_k}{\partial b_q}d\Omega dt \\
&= \int_{T_R}\int_{\Omega}\left(\frac{\partial \Psi_{l+1}}{\partial x_k}+\frac{\partial \Psi_E}{\partial x_k}v_l\right)\frac{\partial \tau_{lk}}{\partial b_q}d\Omega dt + \int_{T_R}\int_{\Omega}\frac{\partial \Psi_E}{\partial x_k}\tau_{lk}\frac{\partial v_l}{\partial b_q}d\Omega dt + \int_{T_R}\int_{\Omega}\frac{\partial \Psi_E}{\partial x_k}\frac{\partial q_k}{\partial b_q}d\Omega dt
\end{aligned}$$

A useful variable is defined as

$$h_{kl} = \frac{\partial \Psi_{l+1}}{\partial x_k} + \frac{\partial \Psi_E}{\partial x_k}v_l$$

By substituting the stress tensor and heat flux expressions from section 3.1 and using the Green-Gauss theorem, one gets

$$\begin{aligned}
I_\Omega^{vis} &= \int_{T_R}\int_{\Omega}\mu h_{kl}\frac{\partial}{\partial x_k}\left(\frac{\partial v_l}{\partial b_q}\right)d\Omega dt + \int_{T_R}\int_{\Omega}\mu h_{kl}\frac{\partial}{\partial x_l}\left(\frac{\partial v_k}{\partial b_q}\right)d\Omega dt \\
&- \frac{2}{3}\delta_{kl}\int_{T_R}\int_{\Omega}\mu h_{kl}\frac{\partial}{\partial x_m}\left(\frac{\partial v_m}{\partial b_q}\right)d\Omega dt \\
&+ \int_{T_R}\int_{\Omega}k\frac{\partial \Psi_E}{\partial x_k}\frac{\partial}{\partial x_k}\left(\frac{\partial T}{\partial b_q}\right)d\Omega dt + \int_{T_R}\int_{\Omega}\frac{\partial \Psi_E}{\partial x_k}\tau_{lk}\frac{\partial v_l}{\partial b_q}d\Omega dt
\end{aligned}$$

and thus,

$$
\begin{aligned}
I_{\Omega}^{vis} = &\int_{T_R}\int_S \mu h_{kl}\frac{\partial v_l}{\partial b_q}n_k dSdt + \int_{T_R}\int_S \mu h_{kl}\frac{\partial v_k}{\partial b_q}n_l dSdt - \frac{2}{3}\delta_{kl}\int_{T_R}\int_S \mu h_{kl}\frac{\partial v_m}{\partial b_q}n_m dSdt \\
&- \int_{T_R}\int_{\Omega} \mu \frac{\partial h_{kl}}{\partial x_k}\frac{\partial v_l}{\partial b_q}d\Omega dt - \int_{T_R}\int_{\Omega} \mu \frac{\partial h_{kl}}{\partial x_l}\frac{\partial v_k}{\partial b_q}d\Omega dt + \frac{2}{3}\delta_{kl}\int_{T_R}\int_{\Omega} \mu \frac{\partial h_{kl}}{\partial x_m}\frac{\partial v_m}{\partial b_q}d\Omega dt \\
&+ \int_{T_R}\int_S k\frac{\partial \Psi_E}{\partial x_k}\frac{\partial T}{\partial b_q}n_k dSdt - \int_{T_R}\int_{\Omega} k\frac{\partial^2 \Psi_E}{\partial x_k^2}\frac{\partial T}{\partial b_q}d\Omega dt + \int_{T_R}\int_{\Omega} \frac{\partial \Psi_E}{\partial x_k}\tau_{lk}\frac{\partial v_l}{\partial b_q}d\Omega dt
\end{aligned}
$$

By using the two identities,

$$
\begin{aligned}
\delta_{kl}h_{kl}\frac{\partial v_m}{\partial b_q}n_m &\equiv \delta_{kl}h_{mm}\frac{\partial v_l}{\partial b_q}n_k \\
\delta_{kl}\frac{\partial h_{kl}}{\partial x_m}\frac{\partial v_m}{\partial b_q} &\equiv \delta_{kl}\frac{\partial h_{mm}}{\partial x_k}\frac{\partial v_l}{\partial b_q}
\end{aligned}
$$

the volume term becomes

$$
\begin{aligned}
I_{\Omega}^{vis} = &\int_{T_R}\int_S \mu\left(h_{kl} + h_{lk} - \frac{2}{3}\delta_{kl}h_{mm}\right)\frac{\partial v_l}{\partial b_q}n_k dSdt \\
&- \int_{T_R}\int_{\Omega} \mu\left(\frac{\partial h_{kl}}{\partial x_k} + \frac{\partial h_{lk}}{\partial x_k} - \frac{2}{3}\delta_{kl}\frac{\partial h_{mm}}{\partial x_k}\right)\frac{\partial v_l}{\partial b_q}d\Omega dt \\
&+ \int_{T_R}\int_S k\frac{\partial \Psi_E}{\partial x_k}\frac{\partial T}{\partial b_q}n_k dSdt - \int_{T_R}\int_{\Omega} k\frac{\partial^2 \Psi_E}{\partial x_k^2}\frac{\partial T}{\partial b_q}d\Omega dt \\
&+ \int_{T_R}\int_{\Omega} \frac{\partial \Psi_E}{\partial x_k}\tau_{lk}\frac{\partial v_l}{\partial b_q}d\Omega dt
\end{aligned}
$$

The adjoint stress tensor and heat flux are defined as

$$
\begin{aligned}
\tau_{kl}^{A} &= \mu\left(h_{kl} + h_{lk} - \frac{2}{3}\delta_{kl}h_{mm}\right) \\
q_k^{A} &= k\frac{\partial \Psi_E}{\partial x_k}
\end{aligned}
$$

By neglecting the surface integrals along the inlet and outlet and taking the flow boundary conditions at the wall into account, I_Ω^{vis} becomes

$$I_\Omega^{vis} = \underbrace{\int_{T_R}\int_{S_w} q_k^A \frac{\partial T}{\partial b_q} n_k dSdt}_{ABC} + \underbrace{\int_{T_R}\int_{S_w} \tau_{kl}^A \frac{\delta_s v_l^g}{\delta_s b_q} n_k dSdt}_{SD} - \underbrace{\int_{T_R}\int_{S_w} \tau_{kl}^A \frac{\partial v_l}{\partial n} v_n^s n_k dSdt}_{SD}$$
$$- \underbrace{\int_{T_R}\int_{\Omega} \left[\left(\frac{\partial \tau_{kl}^A}{\partial x_k} - \frac{\partial \Psi_E}{\partial x_k}\tau_{lk} \right) \frac{\partial v_l}{\partial U_j} + \frac{\partial q_k^A}{\partial x_k}\frac{\partial T}{\partial U_j} \right] \frac{\partial U_j}{\partial b_q} d\Omega dt}_{FAE} \tag{6.10}$$

The viscous surface term is processed as

$$I_S^{vis} = \int_{T_R}\int_{S_w} \Psi_{l+1} \frac{\delta_s(\tau_{lk}n_k)}{\delta_s b_q} dSdt + \int_{T_R}\int_{S_w} \Psi_E \frac{\delta_s(v_l^g \tau_{lk}n_k)}{\delta_s b_q} dSdt + \int_{T_R}\int_{S_w} \Psi_E \frac{\delta_s(q_k n_k)}{\delta_s b_q} dSdt$$

where the $v_l = v_l^g$ boundary condition has been used. The last integral is canceled out since the solid wall is always considered adiabatic. By rearranging terms, I_S^{vis} becomes

$$I_S^{vis} = \int_{T_R}\int_{S_w} g_l \frac{\delta_s(\tau_{lk}n_k)}{\delta_s b_q} dSdt + \int_{T_R}\int_{S_w} \Psi_E \tau_{lk} n_k \frac{\delta_s v_l^g}{\delta_s b_q} dSdt$$

where

$$g_l = \Psi_{l+1} + \Psi_E v_l^g$$

The force $\tau_{lk}n_k$ is substituted from eq. 6.2 and term I_S^{vis} is written as

$$I_S^{vis} = \underbrace{\int_{T_R}\int_{S_w} g_l n_l \frac{\partial \tau^n}{\partial b_q} dSdt}_{ABC} + \underbrace{\int_{T_R}\int_{S_w} g_l n_l \frac{\partial \tau^n}{\partial n} v_n^s dSdt}_{SD} + \underbrace{\int_{T_R}\int_{S_w} g_l \tau^n \frac{\delta_s n_l}{\delta_s b_q} dSdt}_{SD}$$
$$+ \underbrace{\int_{T_R}\int_{S_w} g_l t_l^r \frac{\partial \tau^{t^r}}{\partial b_q} dSdt}_{ABC} + \underbrace{\int_{T_R}\int_{S_w} g_l t_l^r \frac{\partial \tau^{t^r}}{\partial n} v_n^s dSdt}_{SD} + \underbrace{\int_{T_R}\int_{S_w} g_l \tau^{t^r} \frac{\delta_s t_l^r}{\delta_s b_q} dSdt}_{SD}$$
$$+ \underbrace{\int_{T_R}\int_{S_w} \Psi_E \tau_{lk} n_k \frac{\delta_s v_l^g}{\delta_s b_q} dSdt}_{SD} \tag{6.11}$$

6.1.7 The Compressible Field Adjoint Equations

In the previous sections, all terms resulted from the differentiation of the governing equations has been developed and classified. The next step is gathering all FAE, ABC and SD terms from eqs. 6.5, 6.6, 6.7, 6.8, 6.9, 6.10, and 6.11 to give rise to the field adjoint equation, the adjoint boundary conditions, and the sensitivity derivatives. This section focuses on the FAE terms, which are

$$\int_{T_R}\int_{\Omega}\left(\frac{\partial F_\Omega}{\partial U_j}-\frac{\partial \Psi_j}{\partial t}-\frac{\partial \Psi_i}{\partial x_k}A_{ijk}-\left(\left(\frac{\partial \tau^A_{kl}}{\partial x_k}-\frac{\partial \Psi_E}{\partial x_k}\tau_{lk}\right)\frac{\partial v_l}{\partial U_j}+\frac{\partial q^A_k}{\partial x_k}\frac{\partial T}{\partial U_j}\right)\right)\frac{\partial U_j}{\partial b_q}d\Omega dt$$

The multiplier of $\partial U_j/\partial b_q$ is set to zero and this gives rise to the compressible adjoint equations

$$-\frac{\partial \Psi_i}{\partial t}-A_{jik}\frac{\partial \Psi_j}{\partial x_k}-T^{vis}_i+\frac{\partial F_\Omega}{\partial U_i}=0 \tag{6.12}$$

where

$$T^{vis}_i=\frac{\partial \tau^A_{kl}}{\partial x_k}\frac{\partial v_l}{\partial U_i}+\frac{\partial q^A_k}{\partial x_k}\frac{\partial T}{\partial U_i}-\frac{\partial \Psi_E}{\partial x_k}\tau_{lk}\frac{\partial v_l}{\partial U_i}$$

$$\begin{aligned}
\tau^A_{kl}&=\mu\left(h_{kl}+h_{lk}-\frac{2}{3}\delta_{kl}h_{mm}\right)\\
h_{kl}&=\frac{\partial \Psi_{l+1}}{\partial x_k}+\frac{\partial \Psi_E}{\partial x_k}v_l\\
q^A_k&=k\frac{\partial \Psi_E}{\partial x_k}
\end{aligned}$$

and

$$\frac{\partial \vec{v}}{\partial \vec{U}}=\frac{1}{\rho}\begin{bmatrix}-v_1 & 1 & 0 & 0 & 0\\ -v_2 & 0 & 1 & 0 & 0\\ -v_3 & 0 & 0 & 1 & 0\end{bmatrix}, \quad \frac{\partial T}{\partial \vec{U}}=\frac{1}{c_v\rho}\left[-\tfrac{p}{(\gamma-1)\rho}+\tfrac{1}{2}v_k^2 \quad v_1 \quad v_2 \quad v_3 \quad 1\right]^T$$

In the case of unsteady inviscid flows, $T^{vis}_i=0$. Eq. 6.12 represents a linear 5×5 system of PDEs. If the objective function is defined only by a surface integral, source term $\partial F_\Omega/\partial U_i$ vanishes. Moreover, if the time integration of F differs from the simulation's time window (i.e., $T_F\neq T_R$), the source term remains active only during the T_F period. The special meaning of the temporal and convective terms' minus sign is discussed in section 6.4. A similar to the one described in section 3.7 procedure is used for the discretized system solution process. The main difference is that the system's l.h.s. remains constant and, thus, is computed at the beginning of each time step.

6.1.8 The Inlet-Outlet Adjoint Boundary Conditions

The ABC terms are collected from expressions appeared in subsections 6.1.3, 6.1.4, 6.1.5, 6.1.6, and all together give

$$\int_{T_R}\int_{S_{IO}} \frac{\partial F_{S_k}}{\partial v_m}\frac{\partial v_m}{\partial b_q} n_k dS dt + \int_{T_R}\int_{S_{IO}} \frac{\partial F_{S_k}}{\partial p}\frac{\partial p}{\partial b_q} n_k dS dt$$
$$+ \int_{T_R}\int_{S_{IO}} \frac{\partial F_{S_k}}{\partial T}\frac{\partial T}{\partial b_q} n_k dS dt + \int_{T_R}\int_{S_{IO}} \Psi_i A_{ijk}\frac{\partial U_j}{\partial b_q} n_k dS dt$$

They can also be written as

$$\int_{T_R}\int_{S_{IO}} \left(\frac{\partial F_{S_k}}{\partial U_j} + \Psi_i A_{ijk}\right)\frac{\partial U_j}{\partial b_q} n_k dS dt$$

It has already been discussed in section 3.1 that a different number of flow variables are imposed as Dirichlet conditions at the inlet and outlet, depending also on the local Mach number. In each case, a new vector $\vec{Q} \in \mathbb{R}^5$ is introduced containing all the imposed flow quantities accompanied by the variables which are extrapolated from the interior of the domain. For example, at the inlet it could be $\vec{Q} = (p_t, T_t, \alpha_{pitch}, \alpha_{yaw}, |\vec{v}|)$ where, for stationary inlet, $\partial Q_i/\partial b_q = \delta Q_i/\delta b_q = 0,\ i = 1, \cdots, 4$. Therefore, the above terms can be written also as

$$\int_{T_R}\int_{S_{IO}} \left(\frac{\partial F_{S_k}}{\partial U_j} + \Psi_i A_{ijk}\right)\frac{\partial U_j}{\partial Q_h}\frac{\partial Q_h}{\partial b_q} n_k dS dt$$

and is eliminated by introducing the adjoint boundary conditions,

$$\Psi_i A_{ijk} n_k \frac{\partial U_j}{\partial Q_h} = -\frac{\partial F_{S_k}}{\partial Q_h} n_k \tag{6.13}$$

where for all h the condition $\partial Q_h/\partial b_q \neq 0$ is true. In the previous example, $h = 5$ and $Q_h = p$. The term $\partial F_{S_k}/\partial Q_h$ remains active only during the T_F time window and is zero in case the objective function is not defined at the inlet or outlet. Finally, details about the computation of $\partial U_j/\partial Q_h$ are given in Appendix J.

At a subsonic outlet, pressure is usually imposed as a Dirichlet condition, and, thus, $\vec{Q}$ is chosen to be equal to $\vec{V}$. Index h varies from 1 to 4. Hence, a system of 4 equations should be solved there. The $\vec{Q}$ choice should guarantee that the matrix $A_{ijk}n_k\partial U_j/\partial Q_h$ is invertible. However, this is not always possible. The interested reader can find such a case in Appendix K.

6.1.9 The Wall Adjoint Boundary Conditions

The ABC surface integrals along the wall appeared in subsections 6.1.3, 6.1.4, 6.1.5, and 6.1.6 are rewritten below,

$$\begin{aligned}
&\left[\int_{\Omega}\Psi_i\frac{\partial U_i}{\partial b_q}d\Omega\right]_{t_s}^{t_e}+\int_{T_R}\int_{S_w}\left(\Psi_{k+1}+\Psi_E v_k^g+\frac{\partial F_{S_k}}{\partial p}\right)\frac{\partial p}{\partial b_q}n_k dSdt\\
&+\int_{T_R}\int_{S_w}\left(\Psi_i v_k^g\frac{\partial U_i}{\partial b_q}-\Psi_i v_k^g\frac{\partial U_i}{\partial b_q}+q_k^A\frac{\partial T}{\partial b_q}+\frac{\partial F_{S_k}}{\partial T}\frac{\partial T}{\partial b_q}\right)n_k dSdt\\
&-\int_{T_R}\int_{S_w}\left(g_k-\frac{\partial F_{S_k}}{\partial \tau^n}\right)\frac{\partial \tau^n}{\partial b_q}n_k dSdt-\int_{T_R}\int_{S_w}\left(g_k t_k^r-\frac{\partial F_{S_k}}{\partial \tau^{t^r}}n_k\right)\frac{\partial \tau^{t^r}}{\partial b_q}dSdt
\end{aligned}$$

and rearranged as

$$\begin{aligned}
&\left[\int_{\Omega}\Psi_i\frac{\partial U_i}{\partial b_q}d\Omega\right]_{t_s}^{t_e}+\int_{T_R}\int_{S_w}\left(g_k+\frac{\partial F_{S_k}}{\partial p}\right)\frac{\partial p}{\partial b_q}n_k dSdt\\
&+\int_{T_R}\int_{S_w}\left(q_k^A+\frac{\partial F_{S_k}}{\partial T}\right)\frac{\partial T}{\partial b_q}n_k dSdt\\
&-\int_{T_R}\int_{S_w}\left(g_k-\frac{\partial F_{S_k}}{\partial \tau^n}\right)\frac{\partial \tau^n}{\partial b_q}n_k dSdt-\int_{T_R}\int_{S_w}\left(g_k t_k^r-\frac{\partial F_{S_k}}{\partial \tau^{t^r}}n_k\right)\frac{\partial \tau^{t^r}}{\partial b_q}dSdt
\end{aligned}$$

Elimination of the above surface terms leads to

$$\begin{aligned}
g_k n_k &= -\frac{\partial F_{S_k}}{\partial p}n_k\\
g_k n_k &= \ \ \frac{\partial F_{S_k}}{\partial \tau^n}n_k\\
g_k t_k^r &= \ \ \frac{\partial F_{S_k}}{\partial \tau^{t^r}}n_k\\
q_k^A n_k &= -\frac{\partial F_{S_k}}{\partial T}n_k
\end{aligned}$$

In order to get a single boundary condition for $g_k n_k$, the objective function should satisfy the condition

$$\left(\frac{\partial F_{S_k}}{\partial p} + \frac{\partial F_{S_k}}{\partial \tau^n} \right) n_k = 0 \tag{6.14}$$

This condition is true for most objective functions used in real-world optimization problems. Such an objective is the lift or the drag, which are common optimization targets in industrial applications. Consequently, the viscous adjoint wall boundary conditions are

$$\begin{aligned} \left(\Psi_{k+1} + v_k^g \Psi_E\right) n_k &= \frac{\partial F_{S_k}}{\partial \tau^n} n_k \\ \left(\Psi_{k+1} + v_k^g \Psi_E\right) t_k^r &= \frac{\partial F_{S_k}}{\partial \tau^{t^r}} n_k \\ k \frac{\partial \Psi_E}{\partial x_k} n_k &= -\frac{\partial F_{S_k}}{\partial T} n_k \end{aligned} \tag{6.15}$$

The corresponding boundary condition for the inviscid adjoint equations is

$$\Psi_{k+1} n_k = -v_n^g \Psi_E - \frac{\partial F_{S_k}}{\partial p} n_k \tag{6.16}$$

and the condition 6.14 is not necessary anymore.

Subsequently, the elimination of the volume integral is investigated, where two approaches are discussed. Firstly, the objective function is integrated for the whole simulation's time ($T_F = T_R$). Flow initialization at $t = ts$ remains the same at each optimization cycle, and thus, $\left.\frac{\partial U_i}{\partial b_q}\right|_{t_s} = 0$. Therefore, the $\Psi_i(t = t_e) =\ 0$ condition is defined.

On the other hand, in periodic phenomena, T_F is equal to the period of the flow (T_P) that is different from T_R, because the geometry is usually optimized without considering the flow development during the transition phase. Then, the adjoint equations are repetitively solved along the flow simulation's last period until periodicity is established. In other words, $\Psi_i(t_e) = \Psi_i(t_e - T_p)$ is imposed, eliminating the volume term.

6.1.10 Sensitivity Derivatives Expression

The appropriate definition of the field adjoint equations and boundary conditions eliminates all the FAE and ABC integrals. The remaining SD terms provide the formula for the augmented function's gradient computation. Since $\delta F/\delta b_q = \delta L/\delta b_q$, the final expression of the sensitivity derivatives reads

$$\begin{aligned}
\frac{\delta F}{\delta b_q} &= \int_{T_F}\int_{S_w} F_\Omega v_n^s dSdt + \int_{T_R}\int_{S_w} \frac{\partial F_{S_k}}{\partial v_m^g}\frac{\delta_s v_m^g}{\delta_s b_q} n_k dSdt \\
&+ \int_{T_F}\int_{S_w} \left(\frac{\partial F_{S_k}}{\partial n} - F_{S_k}H\right) v_n^s n_k dSdt + \int_{T_F}\int_{S_w} F_{S_k}\frac{\delta_s n_k}{\delta_s b_q} dSdt \\
&+ \int_{T_R}\int_{S_w} \left(\Psi_i v_n^g \frac{\partial U_i}{\partial n} + (\Psi_{k+1}n_k + \Psi_E v_n^g)\frac{\partial p}{\partial n}\right) v_n^s dSdt \\
&+ \int_{T_R}\int_{S_w} \left(p\Psi_{k+1}\frac{\delta_s n_k}{\delta_s b_q} + (\Psi_i U_i + p\Psi_E)\frac{\delta_s v_n^g}{\delta_s b_q}\right) dSdt - \int_{T_R}\int_{S_w} \Psi_i f_{ik}^{inv}\frac{\delta_s n_k}{\delta_s b_q} dSdt \\
&- \int_{T_R}\int_{S_w} \Psi_i \frac{\partial f_{ik}^{inv}}{\partial n} v_n^s n_k dSdt + \int_{T_R}\int_{S_w} \Psi_i f_{ik}^{vis}\frac{\delta_s n_k}{\delta_s b_q} dSdt + \int_{T_R}\int_{S_w} \Psi_i \frac{\partial f_{ik}^{vis}}{\partial n} v_n^s n_k dSdt \\
&+ \int_{T_R}\int_{S_w} \tau_{kl}^A \frac{\delta_s v_l^g}{\delta_s b_q} n_k dSdt - \int_{T_R}\int_{S_w} \tau_{kl}^A \frac{\partial v_l}{\partial n} v_n^s n_k dSdt - \int_{T_R}\int_{S_w} g_l n_l \frac{\partial \tau^n}{\partial n} v_n^s dSdt \\
&- \int_{T_R}\int_{S_w} g_l \tau^n \frac{\delta_s n_l}{\delta_s b_q} dSdt - \int_{T_R}\int_{S_w} g_l t_l^r \frac{\partial \tau^{t^r}}{\partial n} v_n^s dSdt - \int_{T_R}\int_{S_w} g_l \tau^{t^r}\frac{\delta_s t_l^r}{\delta_s b_q} dSdt \\
&- \int_{T_R}\int_{S_w} \Psi_E \tau_{lk} n_k \frac{\delta_s v_l^g}{\delta_s b_q} dSdt
\end{aligned}$$

Taking the viscous adjoint boundary conditions into account,eq. 6.15, the following terms

$$\begin{aligned}
&\int_{T_R}\int_{S_w} \left(\frac{F_{S_k}}{\partial n} n_k + (\Psi_{k+1}n_k + \Psi_E v_n^g)\frac{\partial p}{\partial n} - g_k n_k \frac{\partial \tau^n}{\partial n} - g_k t_k^r \frac{\partial \tau^{t^r}}{\partial n}\right) v_n^s dSdt \\
&= \int_{T_R}\int_{S_w} \left(\frac{F_{S_k}}{\partial n} - \frac{F_{S_k}}{\partial p}\frac{\partial p}{\partial n} - \frac{F_{S_k}}{\partial \tau^n}\frac{\partial \tau^n}{\partial n} - \frac{F_{S_k}}{\partial \tau^{t^r}}\frac{\partial \tau^{t^r}}{\partial n}\right) v_n^s dSdt \\
&= \int_{T_R}\int_{S_w} \frac{F_{S_k}}{\partial T}\frac{\partial T}{\partial n} v_n^s dSdt = 0
\end{aligned}$$

are canceled out. The last equality is true due to the imposed adiabatic wall condition ($\partial T/\partial n = 0$).

By rearranging the remaining terms, one gets

$$\begin{aligned}
\frac{\delta F}{\delta b_q} &= \int_{T_R}\int_{S_w}\left(\Psi_i v_n^g \frac{\partial U_i}{\partial n} - \Psi_i \frac{\partial f_{ik}}{\partial n} n_k - \tau_{kl}^A n_k \frac{\partial v_l}{\partial n} + F_\Omega - F_{S_k} n_k H\right) v_n^s dSdt \\
&+ \int_{T_R}\int_{S_w}\left(F_{S_k} + p\Psi_{k+1} - \Psi_i f_{ik} - g_k \tau^n\right)\frac{\delta_s n_k}{\delta_s b_q} dSdt - \int_{T_R}\int_{S_w} g_k \tau^{t^r} \frac{\delta_s t_k^r}{\delta_s b_q} dSdt \\
&+ \int_{T_R}\int_{S_w}\left(\frac{\partial F_{S_k}}{\partial v_l^g} + \tau_{kl}^A - \Psi_E \tau_{kl}\right)\frac{\delta_s v_l^g}{\delta_s b_q} n_k dSdt \\
&+ \int_{T_R}\int_{S_w}\left(\Psi_i U_i + p\Psi_E\right)\frac{\delta_s v_n^g}{\delta_s b_q} dSdt
\end{aligned} \tag{6.17}$$

where $g_k = \Psi_{k+1} + v_k^g \Psi_E$ and $\vec{f}_k = \vec{f}_k^{inv} - \vec{f}_k^{vis}$. The required normal and tangent derivatives are

$$\begin{aligned}
\frac{\delta_s n_k}{\delta_s b_q} &= -t_k^r \nabla_r u_n^s \\
\frac{\delta_s t_k^r}{\delta_s b_q} &= n_k \nabla_r u_n^s
\end{aligned}$$

where ∇_r is the surface covariant derivative [119]. The corresponding sensitivity derivatives expression for inviscid flows is

$$\begin{aligned}
\frac{\delta F}{\delta b_q} &= \int_{T_R}\int_{S_w}\left(\Psi_i v_n^g \frac{\partial U_i}{\partial n} - \Psi_i \frac{\partial f_{ik}^{inv}}{\partial n} n_k + F_\Omega - F_{S_k} n_k H\right) v_n^s dSdt \\
&+ \int_{T_R}\int_{S_w}\left(F_{S_k} + p\Psi_{k+1} - \Psi_i f_{ik}^{inv}\right)\frac{\delta_s n_k}{\delta_s b_q} dSdt \\
&+ \int_{T_R}\int_{S_w}\left(\Psi_i U_i + p\Psi_E\right)\frac{\delta_s v_n^g}{\delta_s b_q} dSdt
\end{aligned}$$

The continuous adjoint method applied in inviscid flows cannot handle objective functions integrated along the wall depending on flow variables different from pressure. Thus, the $\partial F_{S_k}/\partial v_l^g n_k$ term has been neglected from the expression above.

6.1.11 The Continuous Adjoint Method for Steady Flows

In regard to steady flows, the main differences between the mathematical development presented in subsections 6.1.2 to 6.1.10 for unsteady flows are the lack of the temporal term in the flow equations and the condition $\vec{v}^g = \vec{0}$ along the solid boundaries, which results in $\delta v_k^g/\delta b_q = 0$. Based on eqs. 6.12, 6.13, and 6.15 the field adjoint equations are

$$-A_{jik}\frac{\partial \Psi_j}{\partial x_k} - T_i^{vis} + \frac{\partial F_\Omega}{\partial U_i} = 0$$

accompanied by the proper adjoint boundary conditions at the inlet and outlet,

$$\Psi_i A_{ijk} n_k \frac{\partial U_j}{\partial Q_h} = -\frac{\partial F_{S_k}}{\partial Q_h} n_k$$

and the solid wall,

$$\begin{aligned}
\Psi_{k+1} n_k &= \frac{\partial F_{S_k}}{\partial \tau^n} n_k \\
\Psi_{k+1} t_k^r &= \frac{\partial F_{S_k}}{\partial \tau^{t^r}} n_k \\
k\frac{\partial \Psi_E}{\partial x_k} n_k &= -\frac{\partial F_{S_k}}{\partial T} n_k
\end{aligned}$$

Considering eq. 6.16, the adjoint wall condition for inviscid cases is

$$\Psi_{k+1} n_k = -\frac{\partial F_{S_k}}{\partial p} n_k$$

According to eq. 6.17, the corresponding sensitivity derivatives expression for laminar flows is

$$\begin{aligned}
\frac{\delta F}{\delta b_q} = &- \int_{S_w} \left(\Psi_i \frac{\partial f_{ik}}{\partial n} n_k + \tau_{kl}^A n_k \frac{\partial v_l}{\partial n} - F_\Omega + F_{S_k} n_k H \right) v_n^s dS \\
&+ \int_{S_w} \left(F_{S_k} + p\Psi_{k+1} - \Psi_i f_{ik} - \Psi_{k+1}\tau^n \right) \frac{\delta_s n_k}{\delta_s b_q} dS \\
&- \int_{S_w} \Psi_{k+1} \tau^{t^r} \frac{\delta_s t_k^r}{\delta_s b_q} dS
\end{aligned}$$

and for inviscid flows is

$$\begin{aligned}
\frac{\delta F}{\delta b_q} = &- \int_{S_w} \left(\Psi_i \frac{\partial f_{ik}^{inv}}{\partial n} n_k - F_\Omega + F_{S_k} n_k H \right) v_n^s dS \\
&+ \int_{S_w} \left(F_{S_k} + p\Psi_{k+1} - \Psi_i f_{ik}^{inv} \right) \frac{\delta_s n_k}{\delta_s b_q} dS
\end{aligned}$$

6.2 Mathematical Development of the Incompressible Adjoint Method

This section formulates the continuous adjoint method for steady or unsteady viscous incompressible flows. It presents many similarities with the corresponding mathematical development for the compressible flows presented in section 6.1, and thus, it is kept shorter by just laying emphasis to its the key points. The used flow quantities notation follows the definitions made in section 3.4. The continuous adjoint implementation is based on the differential operators defined in subsection 6.1.1, and the assumptions of section 6.1 are made for indices applying summation.

6.2.1 Definition and Differentiation of the Objective and Augmented Functions

Similarly to subsection 6.1.2, the objective function is defined by a volume or surface integral. Both integrants' dependencies are

$$\begin{aligned} F_\Omega &= F_\Omega(\vec{V}) \\ F_{S_k} &= F_{S_k}(\vec{v}, p, \tau^n, \tau^{t^r}) \end{aligned}$$

The objective function's total derivative w.r.t. b_q is

$$\begin{aligned} \frac{\delta F}{\delta b_q} = & \int_{T_F}\int_\Omega \frac{\partial F_\Omega}{\partial V_j}\frac{\partial V_j}{\partial b_q} d\Omega dt + \int_{T_F}\int_{S_w} F_\Omega v_n^s dS \\ & + \int_{T_F}\int_{S_F} \frac{\partial F_{S_k}}{\partial v_m}\frac{\partial v_m}{\partial b_q} n_k dS dt + \int_{T_F}\int_{S_F} \frac{\partial F_{S_k}}{\partial p}\frac{\partial p}{\partial b_q} n_k dS dt \\ & + \int_{T_F}\int_{S_F} \frac{\partial F_{S_k}}{\partial \tau^n}\frac{\partial \tau^n}{\partial b_q} n_k dS dt + \int_{T_F}\int_{S_F} \frac{\partial F_{S_k}}{\partial \tau^{t^r}}\frac{\partial \tau^{t^r}}{\partial b_q} n_k dS dt \\ & + \int_{T_F}\int_{S_w} \left(\frac{\partial F_{S_k}}{\partial n} - F_{S_k} H\right) v_n^s n_k dS dt + \int_{T_F}\int_{S_w} F_{S_k}\frac{\delta_s n_k}{\delta_s b_q} dS dt \end{aligned} \tag{6.18}$$

The objective function is defined as in subsection 6.1.3 and its differentiation leads to

$$\begin{aligned}\frac{\delta L}{\delta b_q} = &\underbrace{\int_{T_F}\int_{\Omega}\frac{\partial F_\Omega}{\partial V_j}\frac{\partial V_j}{\partial b_q}d\Omega dt}_{FAE} + \underbrace{\int_{T_F}\int_{S_w} F_\Omega v_n^s dSdt}_{SD} + \underbrace{\int_{T_F}\int_{S_F}\frac{\partial F_{S_k}}{\partial v_m}\frac{\partial v_m}{\partial b_q}n_k dSdt}_{ABC/SD} \\ &+ \underbrace{\int_{T_F}\int_{S_F}\frac{\partial F_{S_k}}{\partial p}\frac{\partial p}{\partial b_q}n_k dSdt}_{ABC} + \underbrace{\int_{T_F}\int_{S_F}\frac{\partial F_{S_k}}{\partial \tau^n}\frac{\partial \tau^n}{\partial b_q}n_k dSdt}_{ABC} \\ &+ \underbrace{\int_{T_F}\int_{S_F}\frac{\partial F_{S_k}}{\partial \tau^{t^r}}\frac{\partial \tau^{t^r}}{\partial b_q}n_k dSdt}_{ABC} + \underbrace{\int_{T_F}\int_{S_w}\left(\frac{\partial F_{S_k}}{\partial n} - F_{S_k}H\right)v_n^s n_k dSdt}_{SD} \\ &+ \underbrace{\int_{T_F}\int_{S_w} F_{S_k}\frac{\delta_s n_k}{\delta_s b_q}dSdt}_{SD} + \underbrace{\int_{T_R}\int_{\Omega}\Psi_i\frac{\partial}{\partial b_q}\left(\frac{\partial (M_{ij}V_i)}{\partial t}\right)d\Omega dt}_{I^{temp}} \\ &+ \underbrace{\int_{T_R}\int_{\Omega}\Psi_i\frac{\partial}{\partial b_q}\left(\frac{\partial f_{ik}^{inv}}{\partial x_k}\right)d\Omega dt}_{I^{inv}} - \underbrace{\int_{T_R}\int_{\Omega}\Psi_i\frac{\partial}{\partial b_q}\left(\frac{\partial f_{ik}^{vis}}{\partial x_k}\right)d\Omega dt}_{I^{vis}} \end{aligned} \tag{6.19}$$

where the governing equation's residual is given by eq. 3.4.

6.2.2 Differentiation of the Temporal Term

The I^{temp} integral of eq. 6.19 is processed as

$$I^{temp} = \int_{T_R}\int_{\Omega}\frac{\partial}{\partial t}\left(\Psi_i M_{ij}\frac{\partial V_j}{\partial b_q}\right)d\Omega dt - \int_{T_R}\int_{\Omega}\frac{\partial(\Psi_i M_{ij})}{\partial t}\frac{\partial V_j}{\partial b_q}d\Omega dt$$

By defining the vector

$$\vec{\bar{\Psi}} = M\vec{\Psi}$$

the integral term becomes

$$\begin{aligned} I^{temp} &= \int_{T_R}\frac{\delta}{\delta t}\int_{\Omega}\bar{\Psi}_i\frac{\partial V_i}{\partial b_q}d\Omega dt - \int_{T_R}\int_{S_w}\bar{\Psi}_i\frac{\partial V_i}{\partial b_q}v_n^g dSdt - \int_{T_R}\int_{\Omega}\frac{\partial \bar{\Psi}_i}{\partial t}\frac{\partial V_i}{\partial b_q}d\Omega dt \\ &= \underbrace{\left[\int_{\Omega}\bar{\Psi}_i\frac{\partial V_i}{\partial b_q}d\Omega\right]_{t_s}^{t_e}}_{ABC} - \underbrace{\int_{T_R}\int_{S_w}\bar{\Psi}_i\frac{\partial V_i}{\partial b_q}v_n^g dSdt}_{ABC} - \underbrace{\int_{T_R}\int_{\Omega}\frac{\partial \bar{\Psi}_i}{\partial t}\frac{\partial V_i}{\partial b_q}d\Omega dt}_{FAE} \end{aligned} \tag{6.20}$$

6.2.3 Differentiation of the Convection Term

The I^{inv} integral of eq. 6.19 is developed as

$$\begin{aligned}
I^{inv} &= \int_{T_R}\int_{\Omega} \Psi_i \frac{\partial}{\partial x_k}\left(\frac{\partial f_{ik}^{inv}}{\partial b_q}\right) d\Omega dt \\
&= \underbrace{\int_{T_R}\int_{S_w} \Psi_i \frac{\partial f_{ik}^{inv}}{\partial b_q} n_k dSdt}_{I_S^{inv}} + \underbrace{\int_{T_R}\int_{S_{IO}} \Psi_i A_{ijk} \frac{\partial V_j}{\partial b_q} n_k dSdt}_{ABC} \\
&- \underbrace{\int_{T_R}\int_{\Omega} \frac{\partial \Psi_i}{\partial x_k} A_{ijk} \frac{\partial V_j}{\partial b_q} d\Omega dt}_{FAE} \qquad (6.21)
\end{aligned}$$

where

$$\frac{\partial f_{ik}^{inv}}{\partial b_q} = A_{ijk} \frac{\partial V_j}{\partial b_q}$$

Term I_S^{inv} becomes

$$\begin{aligned}
I_S^{inv} &= \int_{T_R}\int_{S_w} \Psi_i \frac{\delta_s (f_{ik}^{inv} n_k)}{\delta_s b_q} dSdt - \int_{T_R}\int_{S_w} \Psi_i f_{ik}^{inv} \frac{\delta_s n_k}{\delta_s b_q} dSdt \\
&- \int_{T_R}\int_{S_w} \Psi_i \frac{\partial f_{ik}^{inv}}{\partial n} v_n^s n_k dSdt
\end{aligned}$$

By using the wall flux expression,

$$\vec{f}_k^{inv,w} n_k^w = \begin{bmatrix} v_n^g \\ v_1^w v_n^g + p^w n_1^w \\ v_2^w v_n^g + p^w n_2^w \\ v_3^w v_n^g + p^w n_3^w \end{bmatrix} = M\vec{V}^w v_n^g + \begin{bmatrix} v_n^g \\ p^w n_1^w \\ p^w n_2^w \\ p^w n_3^w \end{bmatrix}$$

the integral term becomes

$$\begin{aligned}
I_S^{inv} &= \int_{T_R}\int_{S_w} \Psi_i \frac{\delta_s (M_{ij} V_j v_n^g)}{\delta_s b_q} dSdt + \int_{T_R}\int_{S_w} \Psi_p \frac{\delta_s v_n^g}{\delta_s b_q} dSdt \\
&+ \int_{T_R}\int_{S_w} \Psi_{k+1} \frac{\delta_s (p n_k)}{\delta_s b_q} dSdt \\
&- \int_{T_R}\int_{S_w} \Psi_i f_{ik}^{inv} \frac{\delta_s n_k}{\delta_s b_q} dSdt - \int_{T_R}\int_{S_w} \Psi_i \frac{\partial f_{ik}^{inv}}{\partial n} v_n^s n_k dSdt
\end{aligned}$$

Finally,

$$\begin{aligned}
I_S^{inv} =& \underbrace{\int_{T_R}\int_{S_w}\left(\bar{\Psi}_i v_n^g \frac{\partial V_i}{\partial b_q} + \Psi_{k+1} n_k \frac{\partial p}{\partial b_q}\right) dSdt}_{ABC} \\
&+ \underbrace{\int_{T_R}\int_{S_w}\left(\bar{\Psi}_i v_n^g \frac{\partial V_i}{\partial n} + \Psi_{k+1} n_k \frac{\partial p}{\partial n}\right) v_n^s dSdt}_{SD} \\
&+ \underbrace{\int_{T_R}\int_{S_w}\left(\Psi_{k+1} p \frac{\delta_s n_k}{\delta_s b_q} + \left(\bar{\Psi}_i V_i + \Psi_p\right) \frac{\delta_s v_n^g}{\delta_s b_q}\right) dSdt}_{SD} \\
&- \underbrace{\int_{T_R}\int_{S_w} \Psi_i f_{ik}^{inv} \frac{\delta_s n_k}{\delta_s b_q} dSdt}_{SD} - \underbrace{\int_{T_R}\int_{S_w} \Psi_i \frac{\partial f_{ik}^{inv}}{\partial n} v_n^s n_k dSdt}_{SD} \qquad (6.22)
\end{aligned}$$

6.2.4 Differentiation of the Diffusion Term

A development that is similar to that of subsection 6.1.6 is followed for the viscous integral term,

$$\begin{aligned}
I^{vis} =& \int_{T_R}\int_{\Omega} \Psi_i \frac{\partial}{\partial x_k}\left(\frac{\partial f_{ik}^{vis}}{\partial b_q}\right) d\Omega dt \\
=& \underbrace{\int_{T_R}\int_{S_w} \Psi_i \frac{\delta_s (f_{ik}^{vis} n_k)}{\delta_s b_q} dSdt}_{I_S^{vis}} - \underbrace{\int_{T_R}\int_{S_w} \Psi_i f_{ik}^{vis} \frac{\delta_s n_k}{\delta_s b_q} dSdt}_{SD} \\
&- \underbrace{\int_{T_R}\int_{S_w} \Psi_i \frac{\partial f_{ik}^{vis}}{\partial n} v_n^s n_k dSdt}_{SD} - \underbrace{\int_{T_R}\int_{\Omega} \frac{\partial \Psi_i}{\partial x_k}\frac{\partial f_{ik}^{vis}}{\partial b_q} d\Omega dt}_{I_\Omega^{vis}} \qquad (6.23)
\end{aligned}$$

where the surface integral along S_{IO} has been neglected. The viscous volume term reads

$$\begin{aligned}
I_\Omega^{vis} =& \int_{T_R}\int_{\Omega} \frac{\partial \Psi_{l+1}}{\partial x_k}\frac{\partial \tau_{lk}}{\partial b_q} d\Omega dt \\
=& \int_{T_R}\int_{\Omega} \mu \frac{\partial \Psi_{l+1}}{\partial x_k}\frac{\partial}{\partial x_k}\left(\frac{\partial v_l}{\partial b_q}\right) d\Omega dt + \int_{T_R}\int_{\Omega} \mu \frac{\partial \Psi_{l+1}}{\partial x_k}\frac{\partial}{\partial x_l}\left(\frac{\partial v_k}{\partial b_q}\right) d\Omega dt
\end{aligned}$$

The Green-Gauss theorem is used at each of the above integrals leading to

$$\begin{aligned}I_{\Omega}^{vis} =& \int_{T_R}\int_{S} \mu \frac{\partial \Psi_{l+1}}{\partial x_k}\frac{\partial v_l}{\partial b_q} n_k dSdt + \int_{T_R}\int_{S} \mu \frac{\partial \Psi_{l+1}}{\partial x_k}\frac{\partial v_k}{\partial b_q} n_l dSdt \\ &- \int_{T_R}\int_{\Omega} \mu \frac{\partial^2 \Psi_{l+1}}{\partial x_k^2}\frac{\partial v_l}{\partial b_q} d\Omega dt - \int_{T_R}\int_{\Omega} \mu \frac{\partial^2 \Psi_{l+1}}{\partial x_k \partial x_l}\frac{\partial v_k}{\partial b_q} d\Omega dt\end{aligned}$$

By rearranging terms, it becomes

$$\begin{aligned}I_{\Omega}^{vis} =& \int_{T_R}\int_{S} \mu \left(\frac{\partial \Psi_{l+1}}{\partial x_k} + \frac{\partial \Psi_{k+1}}{\partial x_l}\right)\frac{\partial v_l}{\partial b_q} n_k dSdt \\ &- \int_{T_R}\int_{\Omega} \mu \left(\frac{\partial^2 \Psi_{l+1}}{\partial x_k^2} + \frac{\partial \Psi_{k+1}}{\partial x_k \partial x_l}\right)\frac{\partial v_l}{\partial b_q} d\Omega dt\end{aligned}$$

The adjoint stress tensor in incompressible flows is defined as

$$\tau_{kl}^{A} = \mu \left(\frac{\partial \Psi_{k+1}}{\partial x_l} + \frac{\partial \Psi_{l+1}}{\partial x_k}\right)$$

By neglecting the surface integrals along the inlet and outlet and taking the wall conditions into account, this becomes

$$I_{\Omega}^{vis} = \underbrace{\int_{T_R}\int_{S_w} \tau_{kl}^{A} \frac{\delta_s v_l^g}{\delta_s b_q} n_k dSdt}_{SD} - \underbrace{\int_{T_R}\int_{S_w} \tau_{kl}^{A} \frac{\partial v_l}{\partial n} v_n^s n_k dSdt}_{SD} - \underbrace{\int_{T_R}\int_{\Omega} \frac{\partial \tau_{kl}^{A}}{\partial x_k}\frac{\partial v_l}{\partial b_q} d\Omega dt}_{FAE} \tag{6.24}$$

The viscous surface term is written as

$$\begin{aligned}I_{S}^{vis} =& \int_{T_R}\int_{S_w} \Psi_{l+1} \frac{\delta_s(\tau_{lk} n_k)}{\delta_s b_q} dSdt \\ =& \underbrace{\int_{T_R}\int_{S_w} \Psi_{l+1} n_l \frac{\partial \tau^n}{\partial b_q} dSdt}_{ABC} + \underbrace{\int_{T_R}\int_{S_w} \Psi_{l+1} n_l \frac{\partial \tau^n}{\partial n} v_n^s dSdt}_{SD} + \underbrace{\int_{T_R}\int_{S_w} \Psi_{l+1} \tau^n \frac{\delta_s n_l}{\delta_s b_q} dSdt}_{SD} \\ &+ \underbrace{\int_{T_R}\int_{S_w} \Psi_{l+1} t_l^r \frac{\partial \tau^{t^r}}{\partial b_q} dSdt}_{ABC} + \underbrace{\int_{T_R}\int_{S_w} \Psi_{l+1} t_l^r \frac{\partial \tau^{t^r}}{\partial n} v_n^s dSdt}_{SD} + \underbrace{\int_{T_R}\int_{S_w} \Psi_{l+1} \tau^{t^r} \frac{\delta_s t_l^r}{\delta_s b_q} dSdt}_{SD}\end{aligned} \tag{6.25}$$

6.2.5 The Incompressible Field Adjoint Equations

The FAE terms are gathered from eqs. 6.19, 6.20, 6.21, 6.22, 6.23, 6.24, and 6.25, and can be written as

$$\int_{T_R}\int_{\Omega}\left[\left(\frac{\partial F_\Omega}{\partial V_j}-\frac{\partial \bar{\Psi}_j}{\partial t}-\frac{\partial \Psi_i}{\partial x_k}A_{ijk}\right)\frac{\partial V_j}{\partial b_q}-\frac{\partial \tau^A_{kl}}{\partial x_k}\frac{\partial v_l}{\partial b_q}\right]$$
$$=\int_{T_R}\int_{\Omega}\left(\frac{\partial F_\Omega}{\partial V_j}-\frac{\partial \bar{\Psi}_j}{\partial t}-\frac{\partial \Psi_i}{\partial x_k}A_{ijk}-\frac{\partial \tau^A_{kl}}{\partial x_k}M_{lj}\right)\frac{\partial V_j}{\partial b_q}d\Omega dt$$

The multiplier of $\partial V_j/\partial b_q$ is set to zero defining the incompressible field adjoint equations,

$$-M_{ij}\frac{\partial \Psi_j}{\partial t}-A_{jik}\frac{\partial \Psi_j}{\partial x_k}-\frac{\partial f^{vis,A}_k}{\partial x_k}+\frac{\partial F_\Omega}{\partial V_i}=0 \qquad (6.26)$$

where

$$\vec{f}^{vis,A}_k=\left(\ 0,\ \tau^A_{1k},\ \tau^A_{2k},\ \tau^A_{3k}\ \right)$$

They represent a linear 4×4 system of PDEs in 3D. The source term $\partial F_\Omega/\partial V_i$ is active only during the T_F period. The discretization of eq. 6.26 results in a linear algebraic system which is solved according to the method described in section 3.7.

6.2.6 The Inlet-Outlet Adjoint Boundary Conditions

The ABC terms are aggregated from expressions appeared in subsections 6.2.1, 6.2.2, 6.2.3, 6.2.4,

$$\int_{T_R}\int_{S_{IO}}\frac{\partial F_{S_k}}{\partial v_m}\frac{\partial v_m}{\partial b_q}n_k dSdt+\int_{T_R}\int_{S_{IO}}\frac{\partial F_{S_k}}{\partial p}\frac{\partial p}{\partial b_q}n_k dSdt$$
$$+\int_{T_R}\int_{S_{IO}}\Psi_i A_{ijk}\frac{\partial V_j}{\partial b_q}n_k dSdt$$

where the stress tensor's variation is considered negligible. The same terms are rewritten as

$$\int_{T_R}\int_{S_{IO}}\left(\frac{\partial F_{S_k}}{\partial V_j}+\Psi_i A_{ijk}\right)\frac{\partial V_j}{\partial b_q}n_k dSdt$$

Vector $\vec{Q} \in \mathbb{R}^4$ defined in subsection 6.1.8 is also introduced in the incompressible case. Therefore,

$$\int_{T_R}\int_{S_{IO}} \left(\frac{\partial F_{S_k}}{\partial V_j} + \Psi_i A_{ijk}\right) \frac{\partial V_j}{\partial Q_h}\frac{\partial Q_h}{\partial b_q} n_k dSdt$$

This term is eliminated by introducing the incompressible adjoint boundary conditions,

$$\Psi_i A_{ijk} n_k \frac{\partial V_j}{\partial Q_h} = -\frac{\partial F_{S_k}}{\partial Q_h} n_k \tag{6.27}$$

where for every index h is true that $\partial Q_h/\partial b_q \neq 0$. The term $\partial F_{S_k}/\partial Q_h$ remains active only during the T_F time window and only if the surface S_F comprises the inlet and/or outlet boundaries. Finally, expression $\partial V_j/\partial Q_h$ depends on the implemented boundary condition, which differs for each application.

6.2.7 The Adjoint Wall Conditions

The ABC surface integrals along the wall appeared in subsections 6.2.1, 6.2.2, 6.2.3, and 6.2.4 are written as

$$\left[\int_\Omega \bar{\Psi}_i \frac{\partial V_i}{\partial b_q} d\Omega\right]_{t_s}^{t_e} + \int_{T_R}\int_{S_w} \left(\Psi_{k+1} + \frac{\partial F_{S_k}}{\partial p}\right)\frac{\partial p}{\partial b_q} n_k dSdt$$
$$-\int_{T_R}\int_{S_w} \left(\Psi_{k+1} - \frac{\partial F_{S_k}}{\partial \tau^n}\right)\frac{\partial \tau^n}{\partial b_q} n_k dSdt - \int_{T_R}\int_{S_w} \left(\Psi_{k+1} t_k^r - \frac{\partial F_{S_k}}{\partial \tau^{t^r}} n_k\right)\frac{\partial \tau^{t^r}}{\partial b_q} dSdt$$

Elimination of the above surface terms leads to

$$\begin{aligned} \Psi_{k+1} n_k &= -\frac{\partial F_{S_k}}{\partial p} n_k \\ \Psi_{k+1} n_k &= \frac{\partial F_{S_k}}{\partial \tau^n} n_k \\ \Psi_{k+1} t_k^r &= \frac{\partial F_{S_k}}{\partial \tau^{t^r}} n_k \end{aligned}$$

The above equations lead to a single boundary expression only in case the objective function satisfies condition 6.14. Consequently, the adjoint wall conditions are

$$\begin{aligned} \Psi_{k+1}n_k &= \frac{\partial F_{S_k}}{\partial \tau^n}n_k \\ \Psi_{k+1}t_k^r &= \frac{\partial F_{S_k}}{\partial \tau^{t^r}}n_k \end{aligned} \tag{6.28}$$

The volume integral is eliminated by applying the method discussed in subsection 6.1.9.

6.2.8 Sensitivity Derivatives Expression

The appropriate definition of the field adjoint equations and boundary conditions presented in subsections 6.2.5, 6.2.6, 6.2.7 eliminates all the FAE and ABC integrals. The remaining SD terms provide the formula for the augmented function's gradient computation. Since $\delta F/\delta b_q = \delta L/\delta b_q$, the final expression of the sensitivity derivatives reads

$$\begin{aligned} \frac{\delta F}{\delta b_q} &= \int_{T_F}\int_{S_w} F_\Omega v_n^s dSdt + \int_{T_R}\int_{S_w} \frac{\partial F_{S_k}}{\partial v_m^g}\frac{\delta_s v_m^g}{\delta_s b_q}n_k dSdt \\ &+ \int_{T_F}\int_{S_w}\left(\frac{\partial F_{S_k}}{\partial n} - F_{S_k}H\right)v_n^s n_k dSdt + \int_{T_F}\int_{S_w} F_{S_k}\frac{\delta_s n_k}{\delta_s b_q}dSdt \\ &+ \int_{T_R}\int_{S_w}\left(\bar{\Psi}_i v_n^g \frac{\partial V_i}{\partial n} + \Psi_{k+1}n_k\frac{\partial p}{\partial n}\right)v_n^s dSdt \\ &+ \int_{T_R}\int_{S_w}\left(p\Psi_{k+1}\frac{\delta_s n_k}{\delta_s b_q} + (\bar{\Psi}_i V_i + \Psi_p)\frac{\delta_s v_n^g}{\delta_s b_q}\right)dSdt - \int_{T_R}\int_{S_w}\Psi_i f_{ik}^{inv}\frac{\delta_s n_k}{\delta_s b_q}dSdt \\ &- \int_{T_R}\int_{S_w}\Psi_i\frac{\partial f_{ik}^{inv}}{\partial n}v_n^s n_k dSdt + \int_{T_R}\int_{S_w}\Psi_i f_{ik}^{vis}\frac{\delta_s n_k}{\delta_s b_q}dSdt + \int_{T_R}\int_{S_w}\Psi_i\frac{\partial f_{ik}^{vis}}{\partial n}v_n^s n_k dSdt \\ &+ \int_{T_R}\int_{S_w}\tau_{kl}^A\frac{\delta_s v_l^g}{\delta_s b_q}n_k dSdt - \int_{T_R}\int_{S_w}\tau_{kl}^A\frac{\partial v_l}{\partial n}v_n^s n_k dSdt - \int_{T_R}\int_{S_w}\Psi_{l+1}n_l\frac{\partial \tau^n}{\partial n}v_n^s dSdt \\ &- \int_{T_R}\int_{S_w}\Psi_{l+1}\tau^n\frac{\delta_s n_l}{\delta_s b_q}dSdt - \int_{T_R}\int_{S_w}\Psi_{l+1}t_l^r\frac{\partial \tau^{t^r}}{\partial n}v_n^s dSdt - \int_{T_R}\int_{S_w}\Psi_{l+1}\tau^{t^r}\frac{\delta_s t_l^r}{\delta_s b_q}dSdt \end{aligned}$$

Taking the viscous adjoint boundary conditions into consideration, the following terms

$$\begin{aligned}
&\int_{T_R}\int_{S_w}\left(\frac{F_{S_k}}{\partial n}n_k+\Psi_{k+1}n_k\frac{\partial p}{\partial n}-\Psi_{k+1}n_k\frac{\partial \tau^n}{\partial n}-\Psi_{k+1}t_k^r\frac{\partial \tau^{t^r}}{\partial n}\right)v_n^s dSdt\\
=&\int_{T_R}\int_{S_w}\left(\frac{F_{S_k}}{\partial n}-\frac{F_{S_k}}{\partial p}\frac{\partial p}{\partial n}-\frac{F_{S_k}}{\partial \tau^n}\frac{\partial \tau^n}{\partial n}-\frac{F_{S_k}}{\partial \tau^{t^r}}\frac{\partial \tau^{t^r}}{\partial n}\right)v_n^s dSdt=0
\end{aligned}$$

are canceled out.

By rearranging the remaining terms, one gets

$$\begin{aligned}
\frac{\delta F}{\delta b_q}=&\int_{T_R}\int_{S_w}\left(M_{ij}\Psi_j v_n^g\frac{\partial V_i}{\partial n}-\Psi_i\frac{\partial f_{ik}}{\partial n}n_k-\tau_{kl}^A n_k\frac{\partial v_l}{\partial n}+F_\Omega-F_{S_k}n_k H\right)v_n^s dSdt\\
&+\int_{T_R}\int_{S_w}\left(F_{S_k}+p\Psi_{k+1}-\Psi_i f_{ik}-\Psi_{k+1}\tau^n\right)\frac{\delta_s n_k}{\delta_s b_q}dSdt-\int_{T_R}\int_{S_w}\Psi_{k+1}\tau^{t^r}\frac{\delta_s t_k^r}{\delta_s b_q}dSdt\\
&+\int_{T_R}\int_{S_w}\left(\frac{\partial F_{S_k}}{\partial v_l^g}+\tau_{kl}^A\right)\frac{\delta_s v_l^g}{\delta_s b_q}n_k dSdt+\int_{T_R}\int_{S_w}\left(M_{ij}\Psi_j V_i+\Psi_p\right)\frac{\delta_s v_n^g}{\delta_s b_q}dSdt
\end{aligned}\tag{6.29}$$

where $\vec{f_k}=\vec{f}_k^{inv}-\vec{f}_k^{vis}$. The formulas for the required normal and tangent derivatives computation are given is subsection 6.1.10.

6.2.9 The Continuous Adjoint Method for Steady Flows

Based on the already presented demonstration for the unsteady case, the incompressible field adjoint equations are

$$-A_{jik}\frac{\partial \Psi_j}{\partial x_k}-\frac{\partial f_k^{vis,A}}{\partial x_k}+\frac{\partial F_\Omega}{\partial V_i}=0$$

The accompanied adjoint boundary conditions coincide with the unsteady ones and are mentioned in subsections 6.2.6 and 6.2.7. The corresponding sensitivity derivatives are expressed as

$$\begin{aligned}\frac{\delta F}{\delta b_q} = &-\int_{S_w}\left(\Psi_i\frac{\partial f_{ik}}{\partial n}n_k + \tau^A_{kl}n_k\frac{\partial v_l}{\partial n} - F_\Omega + F_{S_k}n_kH\right)v^s_n dS\\ &+\int_{S_w}\left(F_{S_k} + p\Psi_{k+1} - \Psi_i f_{ik} - \Psi_{k+1}\tau^n\right)\frac{\delta_s n_k}{\delta_s b_q}dS\\ &-\int_{S_w}\Psi_{k+1}\tau^{t^r}\frac{\delta_s t^r_k}{\delta_s b_q}dS\end{aligned}$$

6.3 Discretization of the Steady Adjoint Equations

The adjoint equations present similarities with the corresponding primal flow PDEs, and, consequently, similar techniques can be used for their numerical solution. However, studies on the proper discretization of the continuous adjoint PDEs are limited in the literature. In this thesis, three different discretization methods are developed based on the Roe [262], HLLC [307], and FVS [291] schemes. The section starts with the definition of the adjoint flux and the study of specific properties of the adjoint equations necessary for the discretization schemes formulation. Then, it focuses only on Roe's approach since it is the mainly used scheme in this thesis. The mathematical development of the rest schemes can be found in Appendix L. The adjoint PDEs numerical solution presents similarities with the theory developed for the flow equations in subsections 3.2.1 and 3.2.2. A significant part of the following analysis is described in detail in Appendix E for conservative hyperbolic systems. Thus, attention is mainly paid to parts of significant importance.

The following study focuses on the 1D steady conservative system,

$$A_{ij}\frac{\partial U_j}{\partial x} = 0, \quad i = 1,\cdots,N \tag{6.30}$$

before its generalization to 3D in subsection 6.3.3. According to Appendix E, the system of eqs. 6.30 can be expressed in the form of N wave equations representing the motion of the characteristic waves inside the domain. Their velocity is equal to the matrix coefficient (A) eigenvalues. The wave propagation direction agrees with the number of conditions imposed along each boundary. For example, in compressible subsonic flows where $N=3$, two waves move from the inlet to the interior, and one in the opposite direction. An equal number of Dirichlet conditions are imposed at the inlet and outlet, respectively.

The continuous adjoint method, presented in sections 6.1 and 6.2 implemented in the 1D problem, results in the following adjoint equations,

$$A_{ji}\frac{\partial \Psi_j}{\partial x}=0, \quad i=1,\cdots,N \tag{6.31}$$

The matrix coefficient of the adjoint problem is the transpose of the primal one, meaning that the adjoint characteristic waves' velocity is equal to the primal one. However, the number of the boundary conditions accompanying the adjoint problem at each border of the 1D domain is $N-N_{BC}$, where N_{BC} is the number of the corresponding primal boundary conditions. In the example of the 1D compressible subsonic adjoint problem, one Dirichlet condition should be imposed at the inlet and two at the outlet. Therefore, the waves associated with the adjoint field should have the opposite direction of the primal ones to comply with the corresponding adjoint boundary conditions. Therefore, the adjoint equations are multiplied with -1 to reverse the eigenvalues' sign. Finally, an artificial time derivative is added to the l.h.s. of eqs. 6.31 to convert them to a system of hyperbolic PDEs,

$$\frac{\partial \Psi_i}{\partial t}-A_{ji}\frac{\partial \Psi_j}{\partial x}=0 \tag{6.32}$$

The corresponding initial values defining the adjoint Riemann problem are

$$\Psi_i(x,t=0)=\begin{cases}\Psi_i^L,\ U_i^L & x\leqslant 0\\ \Psi_i^R,\ U_i^R & x>0\end{cases}$$

The flow initialization $\vec{U}^L$ and $\vec{U}^R$ is considered constant in time, while the adjoint field proceeds in time. According to Godunov's method, the time-averaged flux at $x=0$ should be found.

At first, the adjoint flux is introduced. Contrary to the flow equations, the adjoint convection is not conservative, and the definition of the adjoint flux is not straightforward. The suggested definition aims to overcome the convection term's spatial integration challenge and is expressed as

$$f_i^L(x,t)=-\int_{S^LT}^{x}A_{ji}\frac{\partial \Psi_j}{\partial x}dx-A_{ji}^L\Psi_j^L$$

or

$$f_i^R(x,t) = \int_x^{S^R T} A_{ji}\frac{\partial \Psi_j}{\partial x} dx - A_{ji}^R \Psi_j^R$$

where

$$S^L = -max\{\lambda_i^{L,R}\}$$
$$S^R = -min\{\lambda_i^{L,R}\}$$

and λ_i are the eigenvalues of the Jacobian matrices A^L and A^R. Then, the time-averaged flux is

$$f_i^{L0} = \frac{1}{T}\int_0^T f_i^L(x=0,t)dt$$

or

$$f_i^{R0} = \frac{1}{T}\int_0^T f_i^R(x=0,t)dt$$

Generally, $\vec{f}^{L0} \neq \vec{f}^{R0}$, due to the loss of the conservative property. Firstly, $\vec{f}^{L0}$ is computed by integrating eq. 6.32 in the $[S^L T, 0] \times [0,T]$ spatial-temporal domain.

$$\begin{aligned}
&\int_0^T \int_{S^L T}^0 \frac{\partial \Psi_i}{\partial t} dxdt - \int_0^T \int_{S^L T}^0 A_{ji}\frac{\partial \Psi_j}{\partial x} dxdt = 0 \Leftrightarrow \\
&\int_{S^L T}^0 \left(\Psi_i(x,T) - \Psi_i(x,0)\right) dx + \int_0^T \left(f_i^L(0,t) + A_{ji}^L \Psi_j^L\right) dt = 0 \Leftrightarrow \\
&\int_{S^L T}^0 \Psi_i(x,T)dx + \Psi_i^L S^L T + f_i^{L0} T + A_{ji}^L \Psi_j^L T = 0 \Leftrightarrow \\
&f_i^{L0} = -A_{ji}^L \Psi_j^L - \Psi_i^L S^L - \frac{1}{T}\int_{S^L T}^0 \Psi_i(x,T)dx
\end{aligned} \tag{6.33}$$

Similarly, by integrating eq. 6.32 in the $[0, S^R T] \times [0,T]$ domain, one gets

$$f_i^{R0} = -A_{ji}^R \Psi_j^R - \Psi_i^R S^R + \frac{1}{T}\int_0^{S^R T} \Psi_i(x,T)dx \tag{6.34}$$

Computing the unknown integrals of eqs. 6.33 and 6.34 is challenging, but can be facilitated by making a number of simplifications. Based on these assumptions, different discretization schemes appear. In the following subsections and Appendix L, three variants are presented inspired by the Roe, the HLLC, and the FVS schemes.

6.3.1 The Adjoint Roe Scheme

In this subsection, the unknown integrals of eqs. 6.33 and 6.34 are computed by introducing an approximation to the adjoint Riemann problem inspired by the Roe approach. It is based on the assumption that the flow field is constant throughout the 1D domain and equal to the Roe averages ($\vec{\tilde{U}}$) of $\vec{U}^L$ and $\vec{U}^R$. The corresponding equation is

$$\frac{\partial \hat{\Psi}_i}{\partial t} - \tilde{A}_{ji}\frac{\partial \hat{\Psi}_j}{\partial x} = 0 \quad i = 1, \cdots, N \tag{6.35}$$
$$\hat{\Psi}_i(x, t=0) = \begin{cases} \Psi_i^L, & x \leqslant 0 \\ \Psi_i^R, & x > 0 \end{cases}$$

where $\tilde{A} = A(\vec{\tilde{U}})$. In the following expressions, eigenvalues S^L and S^R correspond to the newly defined matrix $\tilde{A}$. Its integration in the $[S^L, 0] \times [0, T]$ control volume gives

$$\int_0^T \int_{S^LT}^0 \frac{\partial \hat{\Psi}_i}{\partial t} dxdt - \int_0^T \int_{S^LT}^0 \tilde{A}_{ji}\frac{\partial \hat{\Psi}_j}{\partial x} dxdt = 0 \Leftrightarrow$$
$$\int_{S^LT}^0 \hat{\Psi}_i(x, T)dx = -\tilde{A}_{ji}\Psi_j^L T + \tilde{A}_{ji}\hat{\Psi}_j^0 T - \Psi_i^L S^L T \tag{6.36}$$

where

$$\hat{\Psi}_i^0 = \frac{1}{T}\int_0^T \hat{\Psi}_i(0, t)dt$$

Substituting eq. 6.36 into eq. 6.33 one gets

$$f_i^{L0} = -A_{ji}^L \Psi_j^L - \tilde{A}_{ji}\left(\hat{\Psi}_j^0 - \Psi_j^L\right) \tag{6.37}$$

Similarly,

$$f_i^{R0} = -A_{ji}^R \Psi_j^R - \tilde{A}_{ji}\left(\hat{\Psi}_j^0 - \Psi_j^R\right) \tag{6.38}$$

The final step is the $\vec{\hat{\Psi}}^0$ computation, which is derived from the solution of the approximate adjoint Riemann problem. It arises by firstly diagonalizing matrix $\tilde{A}$,

$$\tilde{A} = \tilde{P}\tilde{\Lambda}\tilde{P}^{-1}$$

where $\tilde{\Lambda} = diag(\tilde{\lambda}_1, \cdots, \tilde{\lambda}_N)$ with $\tilde{\lambda}_i$ being the eigenvalues of $\tilde{A}$ in ascending order, and columns of $\tilde{P}$ are the right eigenvectors of $\tilde{A}$. Apparently, $S^L = -\tilde{\lambda_N}$ and $S^R = -\tilde{\lambda_1}$. Thereafter, the adjoint Riemann invariants are introduced as

$$\vec{W} = \tilde{P}^T \vec{\hat{\Psi}}$$

Then, eq. 6.35 becomes

$$\frac{\partial \hat{\Psi}_i}{\partial t} - \tilde{P}^{-1}_{mi} \tilde{\Lambda}_{nm} \tilde{P}_{jn} \frac{\partial \hat{\Psi}_j}{\partial x} = 0 \Leftrightarrow \tilde{P}_{mi} \frac{\partial \hat{\Psi}_m}{\partial t} - \tilde{\Lambda}_{mi} \tilde{P}_{jm} \frac{\partial \hat{\Psi}_j}{\partial x} = 0 \Leftrightarrow$$
$$\frac{\partial W_i}{\partial t} - \tilde{\Lambda}_{im} \frac{\partial W_m}{\partial x} = 0$$

The last equation describes the already mentioned motion of N characteristic waves with a velocity of $-\tilde{\lambda}_i$. In other words, W_i remains constant along the characteristic curve $dx/dt = -\tilde{\lambda}_i$. Since $\tilde{\lambda}_i$ is constant, the three curves are the straight lines $x = -\tilde{\lambda}_i t + x_0$, which leads to

$$W_i(x,t) = W_i(x + \hat{\lambda}_i t) = W_i(x_0) = W_i^0$$

Thus, $\vec{W}$ and $\vec{\hat{\Psi}}$ are 1D functions of x/t. Their value along the t-axis is constant and equal to $\vec{W}^0$ and $\vec{\hat{\Psi}}^0$, respectively. They are related through the definition of $\vec{W}$,

$$\hat{\Psi}_i^0 = \sum_{m=1}^{N} \tilde{P}^{-1}_{mi} W_m^0 \Leftrightarrow \hat{\Psi}_i^0 = \sum_{m=1}^{\hat{m}} \tilde{P}^{-1}_{mi} W_m^L + \sum_{m=\hat{m}+1}^{N} \tilde{P}^{-1}_{mi} W_m^R \tag{6.39}$$

where

$$\vec{W}^L = \tilde{P}^{L,T} \vec{\hat{\Psi}}^L$$
$$\vec{W}^R = \tilde{P}^{R,T} \vec{\hat{\Psi}}^R$$

Index $\hat{m}$ is the maximum integer for which $\tilde{\lambda}_{\hat{m}} < 0$. Eq. 6.39 is expressed in two alternative ways,

$$\hat{\Psi}_i^0 = \Psi_i^L + \sum_{m=\hat{m}+1}^{N} \tilde{P}^{-1}_{mi} (W_m^R - W_m^L) \tag{6.40}$$

$$\hat{\Psi}_i^0 = \Psi_i^R - \sum_{m=1}^{\hat{m}} \tilde{P}^{-1}_{mi} (W_m^R - W_m^L) \tag{6.41}$$

By substituting these expressions into eqs 6.37 and 6.38, one concludes that

$$f_i^{L0} = -A_{ji}^L \Psi_j^L - \sum_{m=\hat{m}+1}^{N} \tilde{\lambda}_m P_{mi}^{-1} \left(W_m^R - W_m^L \right)$$

$$f_i^{R0} = -A_{ji}^R \Psi_j^R + \sum_{m=1}^{\hat{m}} \tilde{\lambda}_m P_{mi}^{-1} \left(W_m^R - W_m^L \right)$$

Contrary to the adjoint eq. 6.32, the approximate adjoint Riemann problem of eq. 6.35 is governed by a conservative PDE. Therefore, $f_i^{L0} = f_i^{R0} = f_i^0$. By adding the expressions of $\vec{f}^{L0}$ and $\vec{f}^{R0}$, the final adjoint Roe scheme becomes

$$f_i^0 = \frac{1}{2}(-A_{ji}^L \Psi_j^L - A_{ji}^R \Psi_j^R) - \frac{1}{2} \sum_{m=1}^{N} \tilde{P}_{mi}^{-1} |\tilde{\lambda}_m| \tilde{P}_{jm} (\Psi_j^R - \Psi_j^L) \tag{6.42}$$

where

$$\tilde{\lambda}_m = \begin{cases} -|\tilde{\lambda}_m|, & m \leqslant \hat{m} \\ |\tilde{\lambda}_m|, & m > \hat{m} \end{cases}$$

has been used.

6.3.2 The Corrected Adjoint Roe Scheme

The discretization scheme developed in subsection 6.3.1 does not consider the non-conservative nature of the adjoint equations. This simplification is eliminated by adding a correction term to the flux expression of eq. 6.42 and creating a more accurate discretization scheme as explained in subsection 7.3.1. So, the conservative form of the convection term is added and subtracted from eq. 6.32,

$$\frac{\partial \Psi_i}{\partial t} - A_{ji} \frac{\partial \Psi_j}{\partial x} = 0 \Leftrightarrow \underbrace{\left[\frac{\partial \Psi_i}{\partial t} - \frac{\partial (A_{ji} \Psi_j)}{\partial x} \right]}_{T_h} + \underbrace{\left[\frac{\partial (A_{ji} \Psi_j)}{\partial x} - A_{ji} \frac{\partial \Psi_j}{\partial x} \right]}_{T_c} = 0$$

The first bracket (T_h) corresponds to a conservative hyperbolic equation, and the method developed in subsection 6.3.1 is applied. Its discretized expression at node

i and time instant n in a mesh of equally distributed nodes of spacing Δx is

$$
\begin{aligned}
I_h &= \int_0^T \int_{-\frac{\Delta x}{2}}^{+\frac{\Delta x}{2}} T_h dxdt = \int_0^T \int_{-\frac{\Delta x}{2}}^{+\frac{\Delta x}{2}} \frac{\partial \vec{\Psi}}{\partial t} dxdt - \int_0^T \int_{-\frac{\Delta x}{2}}^{+\frac{\Delta x}{2}} \frac{\partial (A^T \vec{\Psi})}{\partial x} dxdt \\
&= (\vec{\Psi}_i^{n+1} - \vec{\Psi}_i^n)\Delta x + (\vec{f}_{i,i+1}^n - \vec{f}_{i-1,i}^n)\Delta t
\end{aligned}
$$

where Δt is the chosen time step, and $\vec{f}_{i,j}^n$ stands for the flux between nodes i, j which correspond to the left (L) and right (R) states of eq. 6.42. Term T_c stands for the difference between the conservative and non-conservative convection term, representing the necessary correction for the already developed scheme. Its discretization is based on central finite differences,

$$
\begin{aligned}
I_c &= \int_0^T \int_{-\frac{\Delta x}{2}}^{+\frac{\Delta x}{2}} T_c dxdt = \int_0^T \int_{-\frac{\Delta x}{2}}^{+\frac{\Delta x}{2}} \frac{\partial (A^T \vec{\Psi})}{\partial x} dxdt - \int_0^T \int_{-\frac{\Delta x}{2}}^{+\frac{\Delta x}{2}} A^T \frac{\partial \vec{\Psi}}{\partial x} dxdt \\
&= \left.\frac{\partial (A^T \vec{\Psi})}{\partial x}\right|_{i,n} \Delta x \Delta t - A_i^T \left.\frac{\partial \vec{\Psi}}{\partial x}\right|_{i,n} \Delta x \Delta t \\
&= \frac{1}{2}(A_{i+1}^{Tn}\Psi_{i+1}^n - A_{i-1}^{Tn}\Psi_{i-1}^n)\Delta t - \frac{1}{2}A_i^{Tn}(\Psi_{i+1}^n - \Psi_{i-1}^n)\Delta t
\end{aligned}
$$

After rearranging terms, equation $I_h + I_c = 0$ becomes

$$
\begin{aligned}
&(\vec{\Psi}_i^{n+1} - \vec{\Psi}_i^n)\Delta x \\
&+ \left[\left(\vec{f}_{i,i+1}^n + \frac{1}{2}(A_{i+1}^{Tn} - A_i^{Tn})\vec{\Psi}_{i+1}^n\right) - \left(\vec{f}_{i-1,i}^n + \frac{1}{2}(A_{i-1}^{Tn} - A_i^{Tn})\vec{\Psi}_{i-1}^n\right)\right]\Delta t = 0
\end{aligned}
$$

Both parentheses suggest a new non-conservative expression for the flux between the L and R nodes,

$$
\begin{aligned}
f_i^{L0'} &= f_i^0 + \frac{1}{2}(A_{ji}^R - A_{ji}^L)\Psi_j^R \\
f_i^{R0'} &= f_i^0 + \frac{1}{2}(A_{ji}^L - A_{ji}^R)\Psi_j^L
\end{aligned}
$$

concluding with the final expressions,

$$
f_i^{L0'} = -\frac{1}{2}A_{ji}^L(\hat{\Psi}_j^L + \hat{\Psi}_j^R) - \frac{1}{2}\sum_{m=1}^{N} \tilde{P}_{mi}^{-1}|\tilde{\lambda}_m|\tilde{P}_{jm}(\hat{\Psi}_j^R - \hat{\Psi}_j^L) \tag{6.43}
$$

$$
f_i^{R0'} = -\frac{1}{2}A_{ji}^R(\hat{\Psi}_j^L + \hat{\Psi}_j^R) - \frac{1}{2}\sum_{m=1}^{N} \tilde{P}_{mi}^{-1}|\tilde{\lambda}_m|\tilde{P}_{jm}(\hat{\Psi}_j^R - \hat{\Psi}_j^L) \tag{6.44}
$$

6.3.3 The 3D Adjoint Solver

Although all the discretization schemes described in this thesis have the same behavior in terms of convergence, they result in slightly different adjoint fields leading to deviations in the computation of the sensitivity derivatives. Thus, choosing the appropriate discretization scheme is of high importance because it substantially affects the optimization process. Usually, the adjoint version of the flow discretization scheme is the most suitable choice. Therefore, in this thesis, the non-conservative adjoint Roe scheme, eq. 6.43, is preferred, which in 3D applications is expressed as

$$\bar{f}_{ik}^{A,inv,m} n_k^{PQ} = -\frac{1}{2} A_{jik}^{L} \left(\Psi_j^L + \Psi_j^R\right) n_k^{PQ} - \frac{1}{2} \left|\tilde{A}_{jik} n_k^{PQ}\right| \left(\Psi_j^R - \Psi_j^L\right)$$

or

$$\bar{f}_{ik}^{A,inv,m} n_k^{PQ} = -\frac{1}{2} A_{jik}^{P} \left(\Psi_j^P + \Psi_j^Q\right) n_k^{PQ} - \frac{1}{2} \left|\tilde{A}_{jik} n_k^{PQ}\right| \left(\Psi_j^R - \Psi_j^L\right)$$

Its second-order discretization is achieved by applying the MUSCL scheme, as explained in subsection 3.2.3 for the flow equations, where the same limiters, presented in subsection 3.2.4, can be used. The required derivatives of the adjoint variables are given by the Least Square Method described in subsection 3.2.5. Regarding the viscous adjoint terms, they are treated as in subsection 3.2.7. Finally, the pseudo-time step is computed from eq. 3.18. Moreover, the incompressible adjoint equations' discretization is based on methods discussed in section 3.5. Hence, relatively little effort is needed to develop the adjoint solver since it shows many similarities with the corresponding flow solver. Its straightforward implementation is one of the continuous adjoint method's most significant advantages, making it a suitable choice for complex flow software handling industrial applications.

6.4 The Adjoint Method Implemented in Unsteady Flows

The formulation of the continuous adjoint method for unsteady compressible flows results in the adjoint PDEs, eq. 6.12, where the negative sign in front of the temporal, the convection, and diffusion terms is of significant importance. Section 6.3 describes the physical meaning of the convection's negative sign and explains why its negligence drives the adjoint solver to divergence. Therefore, the multiplier of the other two terms should also remain negative. To emphasize the point, the viscous term T_i^{vis}, which represents the elliptic part of the equation modeling diffusive phenomena, should always have a negative sign whenever placed on the l.h.s. of any PDE. First-order forward finite differences discretize the temporal term as

$$-\frac{\partial \Psi_i}{\partial t} = -\frac{\Psi_i^{n+1} - \Psi_i^n}{\Delta t}$$

where Δt is the chosen time step. The negative multiplier $-1/\Delta t$ of the unknown quantity Ψ_i^{n+1} causes serious stability issues preventing the adjoint solver's convergence. Thus, the discretized adjoint equation should be solved w.r.t. Ψ_i^n, which implies the following discretization scheme for stationary geometries,

$$\frac{\Psi_i^n - \Psi_i^{n+1}}{\Delta t}\Omega^n + \frac{\Psi_i^{n,q+1} - \Psi_i^{n,q}}{\Delta \tau}\Omega^n + \sum_{m=1}^{M}\left(f_{ik}^{A,m} n_k^m \Delta S^m\right)^{n,q+1} = 0$$

which imposes that the adjoint equations must be solved backwards in time. The reader is referred to section 3.3 for the used notation definition. Subsection 6.1.9 proves that the initial conditions for non-periodic adjoint problems are defined at the end of the simulation time, which comes to an agreement with the backward in-time integration of the PDEs. However, the same argument also states for periodic flows, even though no initial boundary conditions are imposed.

Hence, the backward-in-time marching adjoint algorithm should start once the solution of the flow PDEs, via the forward-in-time marching algorithm, has been integrated. Computing the instantaneous adjoint field at any time step requires the corresponding flow field at the same time step to be available. In view of the above, the full storage of the flow field's entire time series seems mandatory, although it is not always feasible due to the huge storage requirements. Therefore, alternative methods have been proposed in the literature based on the unsteady flow field recomputation, up to the instant the adjoint equations are solved. The optimal check-pointing techniques, [116], [329], are some of the most widely used methods, according to which the flow field is partially recomputed from selectively stored instantaneous fields. Another alternative of lower computational cost is based on approximations to the already computed unsteady fields built during the forward-in-time marching flow simulation. Some of them efficiently compress and store the flow time series neglecting the less important details of the flow field. Two approaches of this kind were developed and appropriately adjusted to cooperate with the cut-cell and the ghost-cell methods. The first one is based on the Singular Value Decomposition (SVD) [112], [113], [323] and the second one uses the Proper Generalized Decomposition (PGD) [60], [9], [170]. More details can be found in Appendices M, N and section 9.5.

Chapter 7

The Discrete Adjoint Method

This chapter discusses the discrete adjoint to the steady and unsteady solvers based on the cut-cell method. Discrete adjoint codes can be implemented by either differentiating the discretized primal residuals “by hand” or using Algorithmic Differentiation. Even though the latter automates the generation of the discrete adjoint code, the resulting software tends to have significant memory requirements compared to hand-differentiated codes. Therefore, this thesis deals with the hand differentiation of the viscous flow equations of both compressible and incompressible fluids. However, their complex discretized form makes the differentiation a challenging process that hinders the development of the discrete adjoint software. Therefore, various assumptions have been proposed in the literature, simplifying the process at the cost of reducing the accuracy of the computed sensitivity derivatives.

Contrary to the usual practice, no assumptions are made in the following analysis, leading to the exact discrete adjoint expressions. The resulting terms are compared with the corresponding discretization schemes proposed for the continuous adjoint equations. Special treatment is given to the differentiation of the temporal term, taking the cells’ appearance and disappearance into account in applications that involve moving solid bodies. Hereafter, the expressions for the sensitivity derivatives computation are derived. Finally, emphasis is laid on smoothing techniques reducing the high-frequency signals that appear on the sensitivity map, making it capable of optimizing geometrically complex cases.

In the following sections, great effort is made for the clear mathematical development defining new quantities, which allow for a more compact presentation of the resulting expressions. The used notation is based on chapter 6, where the objective function (F) and the design variables (b_q) are defined as well. Moreover, an assumption is made about the indices variation for the rest of the chapter. Indices k, λ, and μ are used for the Cartesian directions, whereas i, j, and l are for the flow variables or equations. Finally, m and n are used for cells, time steps, and geometrical quantities. Contrary to the other chapters, the Einstein notation does not apply here. Whenever summation is implied, the symbol $\sum$ will be used instead, neglecting its upper and lower limits for the sake of brevity. For example, $\sum\limits_{k}$ is the shortcut for $\sum\limits_{k=1}^{3}$ in 3D or $\sum\limits_{k=1}^{2}$ in 2D cases.

7.1 The Discrete Field Adjoint Equation and Sensitivity Derivatives

Consider a computational domain covered by a Cartesian mesh of N cells. The flow PDEs $\vec{R}=\vec{0}$ are discretized at each cell $C \in [1, N]$, forming a set of algebraic equations, $\vec{R}^C=\vec{0}$. According to the discretization method presented in chapter 3, residuals are functions of the unknown flow variables $\vec{U}^n$ stored at the centroid of cells enumerated by index $n \in [1, N]$. Moreover, the residual expressions contain geometrical quantities listed in $\vec{G}^m$, where m is the counter of the list. Its members are the coordinates of the nodes of the triangulated solid surfaces, the normal vector, area, centroid, and velocity of the mesh faces as well as the volume and centroid of the cells.

The boundary displacement due to a change in $\vec{b}$ implies a mesh deformation affecting the aforementioned geometrical quantities. However, the variation in mesh quantities also modifies the flow solution. Thus, $\vec{R}^C$ can be written as a function of $\vec{b}$ in the form,

$$\vec{R}^C = \vec{R}^C\left(\vec{G}^m(\vec{b}), \vec{U}^n(\vec{G}^m(\vec{b}))\right) = \vec{R}^C\left(\vec{G}^m(\vec{b}), \vec{U}^n(\vec{b})\right)$$

Likewise, the objective function, can be expressed in the same way,

$$F = F\left(\vec{G}^m(\vec{b}), \vec{U}^n(\vec{b})\right)$$

Similarly to the continuous adjoint, computing the gradient of F is reduced to the cost of one equivalent flow simulation by introducing the Lagrangian function,

$$L = F + \sum_C \vec{\Psi}^C \cdot \vec{R}^C$$

where $\vec{\Psi}^C$ are the adjoint variables corresponding to cell C. The Lagrangian and the objective function are equal, so do their derivatives. An infinitesimal change in $\vec{b}$ causes a boundary displacement, which is always small enough to prevent the cells' or faces' appearance or disappearance. Therefore, functions $\vec{G}^m(\vec{b})$ are continuous and differentiable in the neighborhood of point $\vec{b}$. The Lagrangian function's derivative is

$$\frac{\delta L}{\delta b_q} = \frac{\delta F}{\delta b_q} + \sum_C \vec{\Psi}^C \cdot \frac{\delta \vec{R}^C}{\delta b_q}$$

where δ is used for the partial derivative w.r.t. b_q. The already presented indirect dependency of F and R on the design variables yields

$$\frac{\delta L}{\delta b_q} = \sum_m \frac{\partial F}{\partial \vec{G}^m} \cdot \frac{\delta \vec{G}^m}{\delta b_q} + \sum_C \sum_m \vec{\Psi}^C \cdot \left(\frac{\partial \vec{R}^C}{\partial \vec{G}^m} \frac{\delta \vec{G}^m}{\delta b_q} \right) + \sum_n \left(\frac{\partial F}{\partial \vec{U}^n} + \sum_C \sum_i \Psi_i^C \frac{\partial R_i^C}{\partial \vec{U}^n} \right) \cdot \frac{\delta \vec{U}^n}{\delta b_q}$$

The high computational cost of term $\delta \vec{U}^n / \delta b_q$ is avoided by eliminating its multiplier giving rise to N discrete adjoint equations,

$$\sum_C \sum_i \Psi_i^C \frac{\partial R_i^C}{\partial \vec{U}^n} + \frac{\partial F}{\partial \vec{U}^n} = \vec{0}, \quad n = 1, \cdots, N \tag{7.1}$$

which form a linear $N \times N$ system. After finding the unknown adjoint variables by solving the aforementioned system, the sensitivity derivatives are computed as

$$\frac{\delta F}{\delta b_q} = \sum_m \frac{\partial F}{\partial \vec{G}^m} \cdot \frac{\delta \vec{G}^m}{\delta b_q} + \sum_C \sum_m \vec{\Psi}^C \cdot \left(\frac{\partial \vec{R}^C}{\partial \vec{G}^m} \frac{\delta \vec{G}^m}{\delta b_q} \right) \tag{7.2}$$

A great advantage of using a Cartesian mesh is that derivatives $\delta \vec{G}^m / \delta b_q$ are non-zero only in cut-cells, drastically reducing the cost of computing the objective function's gradient. The next sections focus on the computation of $\sum_C \sum_i \Psi_i^C (\partial R_i^C / \partial \vec{U}^n)$ and $\sum_C \sum_i \Psi_i^C (\partial R_i^C / \partial \vec{G}^m)$.

Before moving to this analysis, a short investigation is made about the physical meaning of the adjoint variables. Hence, a hypothetical scenario is adopted, according to which term $s_i^{C_0}$ is added on the r.h.s. of residual $R_i^{C_0}$, where C_0 is an arbitrarily chosen cell. Depending on the value of i, this term acts as a source of mass, force, or source of energy. Additionally, consider an optimization problem, according to which the design variables affect only $s_i^{C_0}$. Therefore, an equivalent relation to eq. 7.2 can be derived, where $s_i^{C_0}$ is used instead of $\vec{G}^m$,

$$\frac{\delta F}{\delta b_q} = \frac{\partial F}{\partial s_i^{C_0}} \frac{\delta s_i^{C_0}}{\delta b_q} + \sum_C \vec{\Psi}^C \cdot \left(\frac{\partial \vec{R}^C}{\partial s_i^{C_0}} \frac{\delta s_i^{C_0}}{\delta b_q} \right)$$

However, F and $\vec{R}^C$, where $C \neq C_0$, are not functions of $s_i^{C_0}$, simplifying the above expression to

$$\frac{\delta F}{\delta b_q} = \vec{\Psi}^{C_0} \cdot \left(\frac{\partial \vec{R}^{C_0}}{\partial s_i^{C_0}} \frac{\delta s_i^{C_0}}{\delta b_q} \right) = \Psi_i^{C_0} \frac{\partial R_i^{C_0}}{\partial s_i^{C_0}} \frac{\delta s_i^{C_0}}{\delta b_q} = -\Psi_i^{C_0} \frac{\delta s_i^{C_0}}{\delta b_q}$$

If $s_i^{C_0}$ is the only design variable of this hypothetical optimization problem, the objective's gradient can be written as

$$\frac{\delta F}{\delta s_i^{C_0}} = -\Psi_i^{C_0}$$

which implies that each adjoint variable of a cell C_0 indicates the infinitesimal variation of the objective function caused by an corresponding infinitesimally small source term placed on C_0. An example of this conclusion is given in section 8.1.

7.2 The Discrete Adjoint Flux

The programming of the discrete adjoint method requests the reformation of the field adjoint equations to seem like the discretized flow equations. Hence, the adjoint flux should be defined, which is achieved by the following mathematical development based on a steady flow consideration. Firstly, some new indices are introduced. Index C' refers to all neighboring cells of cell C. Moreover, $\sum\limits_{C'}$ stands for the summation over all neighbors of C. Index F indicates each mesh face, and F_C is a local enumerator usually used for summation over all faces of cell C, which is written as $\sum\limits_{F_C}$. Additionally, if the face belongs to the solid wall boundary, enumerator F_C^w is preferred for stationary faces and F_C^v for moving faces. Each quantity computed on the face can be written either with F or F_C. For example, the normal to the face vector appears as $\vec{n}^F$ or $\vec{n}^{F_C}$. Based on this notation and according to chapter 3, the flow residual can be written as

$$\vec{R}^C = \sum_{F_C} \sum_k \vec{f}_k^{F_C} n_k^{F_C} \Delta S^{F_C}$$

Then, eq. 7.1 suggests that the n^{th} steady adjoint residual is

$$\vec{R}^{A,n} = \sum_C \sum_{F_C} \sum_i \sum_k \Psi_i^C \frac{\partial f_{ik}^{F_C}}{\partial \vec{U}^n} n_k^{F_C} \Delta S^{F_C} + \frac{\partial F}{\partial \vec{U}^n} \tag{7.3}$$

Each inner face F lays between two neighboring cells called P and Q. The face is identified differently by the local enumeration of each cell. Let F be the ${F_P}^{th}$ face of P and the ${F_Q}^{th}$ face of Q. Though F_P and F_Q correspond to the same face, two geometrical or flow quantities q^{F_P} and q^{F_Q} are not always equal. The conservative nature of the flow equations implies

$$\vec{f}_k^{F_P} n_k^{F_P} \Delta S^{F_P} = -\vec{f}_k^{F_Q} n_k^{F_Q} \Delta S^{F_Q}$$

According to eq. 7.3, the flux through each inner face contributes twice in the summation as part of cells P and Q. These terms are

$$\Psi_i^P \frac{\partial f_{ik}^{F_P}}{\partial \vec{U}^n} n_k^{F_P} \Delta S^{F_P} + \Psi_i^Q \frac{\partial f_{ik}^{F_Q}}{\partial \vec{U}^n} n_k^{F_Q} \Delta S^{F_Q} = \left(\Psi_i^P - \Psi_i^Q\right) \frac{\partial f_{ik}^{F_P}}{\partial \vec{U}^n} n_k^{F_P} \Delta S^{F_P}$$

Then, the adjoint residual is reformed by substituting the double sum over C and F_C with a single sum over all mesh faces,

$$\vec{R}^{A,C} = \sum_F \sum_i \sum_k \left(\Psi_i^P - \Psi_i^Q\right) \frac{\partial f_{ik}^F}{\partial \vec{U}^C} n_k^F \Delta S^F + \frac{\partial F}{\partial \vec{U}^C} \tag{7.4}$$

where $\vec{\Psi}^Q = \vec{0}$ is imposed at each boundary face. Moreover, the normal vector n_k^F is defined to point from P to Q. The same convention is also adopted for any other notation of the normal vector (e.g., $\vec{n}^{F_C}$) for the rest of this chapter. The last expression suggests that all fluxes containing $\vec{U}^C$ contribute to the adjoint equation of cell C. These faces can be separated into two categories. In the first one belong faces that separate C from its neighbors. The second one includes faces lying between the first and second neighbors of C. The following example explains the reason why the fluxes through these faces contribute to $\vec{R}^{A,C}$.

Consider cell P, its neighbor Q, and a neighbor of Q called R. Cell R is not a direct neighbor of P. The flux through the face between Q and R depends on the flow variables and their derivatives stored at Q and R. The spatial derivative of $\vec{U}^Q$ is computed by the Least Squares Method, subsection 3.2.5, which uses the flow variables of all direct neighbors of Q, including P. Therefore, the flux between the first (Q) and the second (R) neighbors of P depend on $\vec{U}^P$.

The sum over mesh faces of eq. 7.4 is split into these two categories,

$$\begin{aligned}
\vec{R}^{A,C} &= \sum_{F\in C}\sum_i\sum_k \left(\Psi_i^P - \Psi_i^Q\right)\frac{\partial f_{ik}^F}{\partial \vec{U}^C} n_k^F \Delta S^F \\
&+ \sum_{F\in C'}\sum_i\sum_k \left(\Psi_i^P - \Psi_i^Q\right)\frac{\partial f_{ik}^F}{\partial \vec{U}^C} n_k^F \Delta S^F + \frac{\partial F}{\partial \vec{U}^C} \\
&= \sum_{F_C}\sum_k \vec{f}_k^{A,F_C} n_k^{F_C}\Delta S^{F_C} + \sum_{C'}\vec{B}_C^{C'} + \frac{\partial F}{\partial \vec{U}^C}
\end{aligned}$$

where $\vec{f}_k^{A,F_C}$ is the adjoint flux on face F_C and its components are defined as

$$f_{jk}^{A,F_C} = \sum_i \frac{\partial f_{ik}^{F_P}}{\partial U_j^C}\left(\Psi_i^P - \Psi_i^Q\right) \tag{7.5}$$

If $C = P$, then $\vec{f}_k^{A,F_P}$ is part of the adjoint residual of P. Similarly, if $C = Q$, then $\vec{f}_k^{A,F_Q}$ is part of $\vec{R}^{A,Q}$. The above definition implies that $\vec{f}_k^{A,F_P} \neq \vec{f}_k^{A,F_Q}$ which expresses the non-conservative nature of the field adjoint equations. The primal flux derivative is computed by considering that all flow variables and their derivatives stored at all cells but C are constant. The rest dependencies are taken into account by the following term.

The new vector $\vec{B}_C^{C'}$ will be called the B-term and represents the contribution of each neighbor C' to C. Its components are defined as

$$B_{j\,C}^{\,C'} = \sum_{F_{C'}}\sum_i\sum_k \frac{\partial f_{ik}^{F_{P'}}}{\partial U_j^C}\left(\Psi_i^{P'} - \Psi_i^{Q'}\right) n_k^{F_{C'}}\Delta S^{F_{C'}} \tag{7.6}$$

where P' and Q' are the two cells separated by face $F_{C'}$. Its upper index shows the cell at which the vector is stored, while the lower index signifies the cell's adjoint equation in which $\vec{B}_C^{C'}$ is part of. This vector plays a central role in software parallelization. As mentioned before, the adjoint residual computation at C needs also information from neighbors other than its direct ones. However, the parallelization of CFD software is usually designed to solve discretized equations that exchange only the flow variables and their derivatives between neighbors. The proposed algorithm shown below is designed to overpass these complexities.

Algorithm 13: Discrete Adjoint Residual Computation

```
1  $\vec{R}^{A,C} \leftarrow \vec{0}$
2  foreach mesh face $F$ do
3      $\vec{R}^{A,P} \leftarrow \vec{R}^{A,P} + \vec{f}_k^{A,F_P} n_k^F \Delta S^F$
4      $\vec{R}^{A,Q} \leftarrow \vec{R}^{A,Q} + \vec{f}_k^{A,F_Q} n_k^F \Delta S^F$
5  end
6  foreach mesh cell $C$ do
7      $\vec{R}^{A,C} \leftarrow \vec{R}^{A,C} + \frac{\partial F}{\partial \vec{U}^C}$
8      foreach cell's neighbor $C'$ do
9          $\vec{B}_{C'}^{C} \leftarrow \vec{0}$
10         foreach cell's face $F_C$ do
11             $\vec{B}_{C'}^{C} \leftarrow \vec{B}_{C'}^{C} + \sum_i \sum_k \frac{\partial f_{ik}^{F_C}}{\partial \vec{U}^{C'}} \left(\Psi_i^P - \Psi_i^Q\right) n_k^{F_C} \Delta S^{F_C}$
12         end
13     end
14 end
15 exchange $\vec{B}_{C'}^{C}$ $\forall C, C'$ between processors
16 foreach mesh cell $C$ do
17     foreach cell's neighbor $C'$ do
18         $\vec{R}^{A,C} \leftarrow \vec{R}^{A,C} + \vec{B}_C^{C'}$
19     end
20 end
```

It starts with a loop over all mesh faces and computes the adjoint fluxes of cells P and Q, imitating the flow solver's algorithm. Afterwards, each mesh cell C is responsible for computing the objective function's source term and the $\vec{B}_{C'}^{C}$ contributions for its direct neighbors C'. For each and every pair (C, C'), the vector's computation requires the loop over all faces F_C. Then, vectors $\vec{B}^C_{C'}$ are exchanged between the processors' boundary cells and added to the adjoint equation's residual of the appropriate cell.

7.3 The Compressible Discrete Adjoint Equation

According to eq. 7.1, the discretized flow equations and the objective function should be differentiated w.r.t. the vector of the flow variables, which, in the case of compressible flows, are the conservative variables $\vec{U}^n$ defined in section 3.1. However, the differentiation w.r.t. the primitive variables $\vec{V}^n$ is much easier and leads to the same field adjoint equation. This statement is easily shown by rewriting eq. 7.1 as

$$\sum_C \sum_i \Psi_i^C \frac{\partial R_i^C}{\partial \vec{U}^n} + \frac{\partial F}{\partial \vec{U}^n} = 0 \Leftrightarrow$$
$$\sum_C \sum_i \sum_j \Psi_i^C \frac{\partial R_i^C}{\partial V_j^n} \frac{\partial V_j^n}{\partial \vec{U}^n} + \sum_j \frac{\partial F}{\partial V_j^n} \frac{\partial V_j^n}{\partial \vec{U}^n} = 0$$

Matrix $\partial\vec{V}/\partial\vec{U}$ is always invertible, which allows for the equation's simplification,

$$\sum_C \sum_i \Psi_i^C \frac{\partial R_i^C}{\partial \vec{V}^n} + \frac{\partial F}{\partial \vec{V}^n} = 0$$

Therefore, for the reasons mentioned above, the residual derivation w.r.t. $\vec{V}^n$ is preferred.

The computation of the adjoint flux and B-term is the target of the following subsections. The class of matrices introduced below plays a central role in this process,

$$D_{iX}^{\;Y} = \frac{\partial p_i^Y}{\partial q_i^X}$$

where $\vec{p}^X$ and $\vec{q}^Y$ are any flow quantities. Indices X and Y represent either cells (C, C') or faces (F_C, F_C^w, F_C^v, F') and indicate the position at which each quantity is computed. If neither X nor Y is a dotted index (C', F'), the symbol $\mathbb{D}$ is used instead, and the Jacobian matrix is defined as

$$\mathbb{D}^Y_{ij} = \frac{\partial p^Y_i}{\partial q^X_j}$$

Therefore, matrix $\mathbb{D}$ is used for the adjoint flux computation while D contributes to the B-term's expression. During the diffusion differentiation, quantity p represents a matrix instead of a vector. Then, the above definitions are transformed to

$$D_{ik}{}^Y_X = \frac{\partial p^Y_{ik}}{\partial q^X_i}$$

and

$$\mathbb{D}^Y_{ijk} = \frac{\partial p^Y_{ik}}{\partial q^X_j}$$

In case q is a geometrical quantity, a tilde is added on top of the matrix (e.g., $\tilde{D}_i{}^Y_X$).

7.3.1 Differentiation of the Convection Term

According to eq. 7.5, the adjoint inviscid flux is

$$f^{inv,A,F_C}_{jk} = \sum_i \frac{\partial f^{inv,F_P}_{ik}}{\partial V^C_j} \left(\Psi^P_i - \Psi^Q_i\right)$$

where $\vec{f}^{inv,F_P}_k$, defined by eq. 3.5, is a function of the flow variables stored at P and Q centroids. However, according to the MUSCL scheme, subsection 3.2.3, the flux is also a function of the flow variables extrapolated from the P and Q to the centroid of F, given by eq. 3.9. These variables will be referred to as $\hat{\vec{V}}^{F_P}$ and $\hat{\vec{V}}^{F_Q}$, where $\hat{\vec{V}}^{F_P} \neq \hat{\vec{V}}^{F_Q}$. By combining eqs. 3.9 and 3.13, every extrapolated flow variable $\hat{\vec{V}}^{F_C}$ can be written as

$$\hat{V}^{F_C}_i = V^C_i + \phi^C_i \sum_k \eth V^C_{ik} \Delta x^{F_C}_k = V^C_i + \phi^C_i \sum_k \sum_{C'} \mathbb{C}_k{}^C_{C'} \left(V^{C'}_i - V^C_i\right) \Delta x^{F_C}_k$$

where $\eth V^C_{ik}$ stands for the spatial derivative of V^C_i w.r.t. x_k, $\vec{\mathbb{C}}^C_{C'}$ is an alternative notation for $\vec{W}$ defined in eq. 3.12, $\Delta\vec{x}^{F_C}$ is a vector positioned at C pointing to F centroids and $\vec{\phi}^C$ stands for the used limiter. Only the limiter by Barth and Jespersen, is differentiated and used in the discrete adjoint formulation. This is computed as

$$\phi_i^C = \frac{V_i^{\bar{C}} - V_i^C}{\hat{V}_i^{\bar{F}_C} - V_i^C}$$

According to the notation used in subsection 3.2.4 for the limiter's definition, $V_i^{\bar{C}}$ is either $V_{i_{max}}$ or $V_{i_{min}}$, and $\hat{V}_i^{\bar{F}_C}$ is equal to V_i^f. For more information, see eq. 3.10.

Moreover, four matrices, necessary for the following mathematical development, are defined as

$$\begin{aligned} \mathbb{D}_{ij}^{F_C} &= \left.\frac{\partial \hat{V}_i^{F_C}}{\partial V_j^C}\right|_{\vec{\phi}^C}, & \mathbb{D}L_{ij}^{F_C} &= \frac{\partial \hat{V}_i^{F_C}}{\partial V_j^C} \\ D_{iC'}^{F_C} &= \left.\frac{\partial \hat{V}_i^{F_C}}{\partial V_i^{C'}}\right|_{\vec{\phi}^C}, & DL_{iC'}^{F_C} &= \frac{\partial \hat{V}_i^{F_C}}{\partial V_i^{C'}} \end{aligned} \tag{7.7}$$

where the sidebar denotes that the limiter is considered constant during the corresponding differentiation.

The final expressions for the inviscid adjoint flux and the B-term are

$$\begin{aligned} f_{jk}^{inv,A,F_C} &= \sum_i \left.\frac{\partial f_{ik}^{inv,F_P}}{\partial V_j^C}\right|_{\vec{\hat{V}}^C} \left(\Psi_i^P - \Psi_i^Q\right) + \sum_i \sum_l \frac{\partial f_{ik}^{inv,F_P}}{\partial \hat{V}_l^C} \left(\Psi_i^P - \Psi_i^Q\right) \mathbb{D}L_{lj}^{F_C} \\ B_{j\ \ C'}^{inv\,C} &= \sum_{F_C} \sum_i \sum_k \frac{\partial f_{ik}^{inv,F_P}}{\partial \hat{V}_j^C} \left(\Psi_i^P - \Psi_i^Q\right) DL_{jC'}^{F_C}\; n_k^{F_C} \Delta S^{F_C} \end{aligned}$$

The first term of the adjoint flux expression is computed by considering $\vec{\hat{V}}^C$ constant and is non-zero only if eq. 3.5 is used instead of eq. 3.4 for the flow equations discretization. The matrices defined in eq. 7.7 are

$$\begin{aligned} \mathbb{D}L_{ij}^{F_C} &= \left(\frac{\partial V_i^{\bar{C}}}{\partial V_j^C} - \mathbb{D}_{ij}^{\bar{F}_C}\right) T_{i\bar{F}}^{F_C} + \mathbb{D}_{ij}^{F_C} \\ DL_{iC'}^{F_C} &= \left(\delta_{\bar{C}C'} - D_{iC'}^{\bar{F}_C}\right) T_{i\bar{F}}^{F_C} + D_{iC'}^{F_C} \\ \mathbb{D}_{ij}^{F_C} &= \left[1 - \phi_i^C \sum_k \left(\sum_{C'} \mathbb{C}_{kC'}^{\ C}\right) \Delta x_k^{F_C}\right] \delta_{ij} + \phi_i^C \sum_k \left(\sum_{C'}^{BF} \mathbb{C}_{kC'}^{\ C} Q_{ij}^{C'}\right) \Delta x_k^{F_C} \\ D_{iC'}^{F_C} &= \phi_i^C \sum_k \mathbb{C}_{kC'}^{\ C} \Delta x_k^{F_C} \\ T_{i\bar{F}}^{F_C} &= \left(\sum_k \eth V_{ik}^C \Delta x_k^{F_C}\right) / \left(\sum_k \eth V_{ik}^C \Delta x_k^{\bar{F}_C}\right) \end{aligned} \tag{7.8}$$

and

$$\frac{\partial V_i^{\bar{C}}}{\partial V_j^{C}} = \begin{cases} \delta_{ij}\delta_{\bar{C}C}, & \exists\ \bar{C} \\ Q_{ij}^{\bar{C}}, & \nexists\ \bar{C} \end{cases}$$

The above expressions can be applied in all but wall faces. Symbol $\sum\limits_{C'}^{BF}$ stands for the summation over all boundary faces of C. The $Q_{ij}^{\bar{C}}$ and $Q_{ij}^{C'}$ appear in boundary faces where cells $\bar{C}$ or C' do not exist and correspond to the differentiation of the imposed boundary conditions on that face expressed as $\partial V_i^{BC}/\partial V_j^{C}$, where $\vec{V}^{BC}$ is defined in section 3.1.

The previously described discretization can be simplified by avoiding the limiter differentiation. In this case, $\mathbb{D}L_{ij}^{F_C} = \mathbb{D}_{ij}^{F_C}$ and $DL_{iC'}^{F_C} = D_{iC'}^{F_C}$. Matrices $\mathbb{D}_{ij}^{F_C}$ and $D_{iC'}^{F_C}$ are responsible for discretizing the adjoint flux and the B-term with second order accuracy. A first order discretization is possible by setting $\mathbb{D}L_{ij}^{F_C} = \mathbb{D}_{ij}^{F_C} = \delta_{ij}$ and $DL_{iC'}^{F_C} = D_{iC'}^{F_C} = 0$, which significantly reduces the computational cost of the adjoint solution process.

Derivative $\partial f_k^{inv,F_P}/\partial \vec{\hat{V}}^{C}$ emerges by differentiating eq. 3.4 w.r.t. $\vec{\hat{V}}^{P}$ and $\vec{\hat{V}}^{Q}$,

$$\begin{aligned} \frac{\partial f_{ik}^{inv,F_P}}{\partial \hat{V}_j^{P}} n_k^F &= \sum_k \bar{A}_{ijk}^{P} n_k^F + \frac{1}{2}\sum_l \sum_k |\tilde{A}_{ilk} n_k^F| W_{lj}^{P} - \frac{1}{2} A_{ij}^{d,P} \\ \frac{\partial f_{ik}^{inv,F_P}}{\partial \hat{V}_j^{Q}} n_k^F &= \sum_k \bar{A}_{ijk}^{Q} n_k^F - \frac{1}{2}\sum_l \sum_k |\tilde{A}_{ilk} n_k^F| W_{lj}^{Q} - \frac{1}{2} A_{ij}^{d,Q} \end{aligned} \tag{7.9}$$

where $\bar{A}$ is the Jacobian matrix of the flux derivative w.r.t. the primitive flow variables and

$$W_{ij}^{C} = \frac{\partial U_i^{C}}{\partial V_j^{C}}$$

$$A_{ij}^{d,C} = \sum_l \sum_k \frac{\partial |\tilde{A}_{ilk} n_k^F|}{\partial V_j^{C}} \left(U_l^{Q} - U_l^{P}\right) \tag{7.10}$$

Matrix $A^{d,C}$ arises from the differentiation of the absolute Jacobian matrix, which is presented in Appendix O. The computational effort needed for its computation is very high. Numerical examples in inviscid flows around isolated airfoils using coarse meshes signify that its elimination affects at most the third significant digit of the sensitivity derivatives. Thus, avoiding its computation is advantageous when the derivatives high accuracy is not crucial for the optimization implementation.

An essential outcome of the previous mathematical development is the relation of the discrete adjoint equation with the adjoint Roe scheme, shown in subsection 6.3.3, which is used in the continuous adjoint method. The comparison is more clear if a first order consideration is made. By additionally setting $A^{d,C} = 0$, the two schemes coincide, showing the remarkable correlation between the continuous and adjoint variants. This comparison also suggests that the corrected adjoint Roe scheme developed in subsection 6.3.2 is probably the best choice among the adjoint schemes presented in this thesis, provided that the Roe scheme discretizes the flow equations.

Finally, the adjoint wall flux is

$$\sum_k f_{jk}^{inv,A,F_C^w} n_k^{F_C^w} = \sum_i \sum_l \sum_k \frac{\partial(f_{ik}^{F_C^w} n_k^{F_C^w})}{\partial \hat{V}_l^C} \left(\Psi_i^P - \Psi_i^Q\right) \mathbb{D}L_{lj}^{F_C^w}$$

with $\mathbb{D}L_{ij}^{F_C^w}$ computed like $\mathbb{D}L_{ij}^{F_C}$ and

$$\begin{aligned} \frac{\partial(\vec{f}_k^{F_C^w} n_k^{F_C^w})}{\partial \hat{V}_m^C} &= \vec{0}, \quad \forall\, m \neq 5, \\ \frac{\partial(\vec{f}_k^{F_C^w} n_k^{F_C^w})}{\partial \hat{V}_5^C} &= \left(0, \quad n_1, \quad n_2, \quad n_3, \quad \sum_k \hat{v}_k^g n_k\right) \end{aligned}$$

where $\hat{v}_k^g$ is defined in subsection 3.3.1.

7.3.2 Differentiation of the Diffusion Term

The adjoint viscous flux and B-term are defined as

$$\begin{aligned} f_{jk}^{vis,A,F_C} &= \sum_i \frac{\partial f_{ik}^{vis,F_C}}{\partial V_j^C} \left(\Psi_i^P - \Psi_i^Q\right) \\ B_{j\ \ C'}^{vis\,C} &= \sum_{F_C} \sum_i \sum_k \frac{\partial f_{ik}^{vis,F_C}}{\partial V_j^{C'}} \left(\Psi_i^P - \Psi_i^Q\right) n_k^{F_C} \Delta S^{F_C} \end{aligned}$$

where $f_{ik}^{vis,F}$ computation is presented in subsection 3.2.7 and depends on $\vec{V}^P$ and $\vec{V}^Q$ as well as the spatial derivatives of the primitive flow variables on F_C, which are denoted by

$$\left.\frac{\partial V_i}{\partial x_k}\right|_F = \eth V_{ik}^F$$

for the sake of simplicity. They are given by the orthogonal correction formula, eq. 3.17, which is rewritten as

$$\eth V_{ik}^F = \eth V_{ik}^w - \left(\sum_\lambda \eth V_{i\lambda}^w \alpha_\lambda^F + \Delta^{F_P} V_i^P + \Delta^{F_Q} V_i^Q\right) \alpha_k^F \tag{7.11}$$

The newly presented quantities are

$$\begin{aligned} \eth V_{ik}^w &= w^{F_P} \eth V_{ik}^P + w^{F_Q} \eth V_{ik}^Q, \\ w^{F_P} &= w^F, \quad w^{F_Q} = 1 - w^F \\ \Delta^{F_P} &= \frac{s^P}{|\vec{x}^Q - \vec{x}^P|}, \quad \Delta^{F_Q} = \frac{s^Q}{|\vec{x}^Q - \vec{x}^P|}, \\ s^P &= 1, \quad s^Q = -1 \end{aligned}$$

and w^F, $\vec{\alpha}^F$ are equal to the weight w and vector $\vec{\alpha}$ defined in subsection 3.2.7. The spatial derivatives $\eth V_{ik}^P$ and $\eth V_{ik}^Q$ of V_i^P and V_i^Q are computed by the Least Square Method explained in subsection 3.2.5. Two matrices are defined

$$\begin{aligned} \mathbb{D}_{ijk}^C &= \frac{\partial \eth V_{ik}^C}{\partial V_j^C} \\ D_{ik}{}_{C'}^{C} &= \frac{\partial \eth V_{ik}^C}{\partial V_i^{C'}} \end{aligned} \tag{7.12}$$

following the same process as in subsection 7.3.1. Then, the final expressions for the adjoint viscous flux and the B-term are

$$f_{jk}^{vis,A,F_C} = \sum_\lambda \tau^A_{\lambda jk}{}^{F_C}_{C} \left(\Psi_{\lambda+1}^P - \Psi_{\lambda+1}^Q\right) + f^E_{jk}{}^{F_C}_{C} \left(\Psi_5^P - \Psi_5^Q\right) \tag{7.13}$$

$$B_j{}^C_{C'} = \sum_{F_C} \sum_\lambda \sum_k \left[\tau^A_{\lambda jk}{}^{F_C}_{C'} \left(\Psi_{\lambda+1}^P - \Psi_{\lambda+1}^Q\right) + f^E_{jk}{}^{F_C}_{C'} \left(\Psi_5^P - \Psi_5^Q\right)\right] n_k^{F_C} \Delta S^{F_C} \tag{7.14}$$

The adjoint stress tensors $\tau^A_{\lambda jk}{}^{F_C}_{C}$ and $\tau^A_{\lambda jk}{}^{F_C}_{C'}$ are given by a similar formula. Moreover, variables $f^E_{jk}{}^{F_C}_{C}$ and $f^E_{jk}{}^{F_C}_{C'}$ denote the contribution of the energy diffusion differentiation to the adjoint equations and are computed analogously. Expressions like these remain similar for cells C and C' and make use of the symbol $C(')$. Therefore,

$$\tau^{A\ \ F_C}_{\lambda jk C(')} = \mu \left(\frac{\partial đ V^{F_C}_{\lambda+1,k}}{\partial V^{C(')}_j} + \frac{\partial đ V^{F_C}_{k+1,\lambda}}{\partial V^{C(')}_j} - \frac{2}{3} \sum_\mu \frac{\partial đ V^{F_C}_{\mu+1,\mu}}{\partial V^{C(')}_j} \delta_{\lambda k} \right)$$

$$f^{E\ F_C}_{jk C(')} = \sum_\lambda \left(V^{F_C}_{\lambda+1}\ \tau^{A\ \ F_C}_{\lambda jk C(')} + \frac{\partial V^{F_C}_{\lambda+1}}{\partial V^{C(')}_j} \tau_{\lambda k} \right) + q^{A\ F_C}_{jk C(')}$$

The newly introduced adjoint heat flux $q^{A\ F_C}_{jk C(')}$ is

$$\begin{aligned} q^{A\ F_C}_{jk C(')} = kT^{F_C} \Bigg[& \frac{1}{V^{F_C}_5} \frac{\partial đ V^{F_C}_{5k}}{\partial V^{C(')}_j} - \frac{1}{V^{F_C}_1} \frac{\partial đ V^{F_C}_{1k}}{\partial V^{C(')}_j} \\ & - \frac{1}{(V^{F_C}_5)^2} đ V^{F_C}_{5k} \frac{\partial V^{F_C}_5}{\partial V^{C(')}_j} + \frac{1}{(V^{F_C}_1)^2} đ V^{F_C}_{1k} \frac{\partial V^{F_C}_1}{\partial V^{C(')}_j} \\ & + đ T^{F_C}_k \left(\frac{1}{V^{F_C}_5} \frac{\partial V^{F_C}_5}{\partial V^{C(')}_j} - \frac{1}{V^{F_C}_1} \frac{\partial V^{F_C}_1}{\partial V^{C(')}_j} \right) \Bigg] \end{aligned}$$

in all faces, except those belonging to solid boundaries, where $q^{A\ F_C}_{jk C(')} = 0$ and $f^{E\ F_C}_{jk C(')} = \sum_\lambda V^{F_C}_{\lambda+1}\ \tau^{A\ \ F_C}_{\lambda jk C(')}$. Term $\partial đ V^{F_C}_{ik} / \partial V^{C(')}_j$ is computed by differentiating the orthogonal correction formula given by eq. 7.11 w.r.t. the primal variables of C and C', which reads

$$\frac{\partial đ V^{F_C}_{ik}}{\partial V^C_j} = w^{F_C} \mathbb{D}^C_{ijk} - \alpha^F_k w^{F_C} \left(\sum_\lambda \mathbb{D}^C_{ij\lambda} \alpha^F_\lambda \right) - \alpha^F_k \Delta^{F_C} \delta_{ij} + \alpha^F_k \Delta^{F_C} \sum_l Q^C_{il} \mathbb{D}L^{F_C}_{lj}$$

$$\frac{\partial đ V^{F_C}_{ik}}{\partial V^{C'}_j} = w^{F_C} D_{ik}{}^{C}_{C'} \delta_{ij} - \alpha^F_k w^{F_C} \left(\sum_\lambda D_{i\lambda}{}^{C}_{C'} \alpha^F_\lambda \right) \delta_{ij} + \alpha^F_k \Delta^{F_C} Q^C_{ij} DL_j{}^{F_C}_{C'}$$

The last term in both equations appears only if F_C is a boundary face and contains the matrices $\mathbb{D}L^{F_C}_{ij}$ and $DL_{ik}{}^{F_C}_{C'}$ defined in eq. 7.7 and Q^C, representing the differentiation of the boundary conditions w.r.t. $\vec{V}^C$. It is also reminded that, for these faces, $w^{F_C} = 1$. The used matrices defined in eq. 7.12 are

$$\mathbb{D}^C_{ijk} = - \left(\sum_{C'} \mathbb{C}_k{}^C_{C'} \right) \delta_{ij} + \sum_{C'}^{BC} \mathbb{C}_k{}^C_{C'} Q^{C'}_{ij}$$

$$D_{ik}{}^C_{C'} = \mathbb{C}_k{}^C_{C'} \quad \forall\ i$$

Finally, the flow variables $V_i^{F_C}$ are given by eq. 3.16 and their derivative w.r.t. $V_j^{C(')}$ is

$$\frac{\partial V_i^{F_C}}{\partial V_j^C} = \sum_l \frac{\partial V_i^{F_C}}{\partial \hat{V}_l^C} \mathbb{D}L_{lj}^{F_C}, \quad \frac{\partial V_i^{F_C}}{\partial V_j^{C'}} = \frac{\partial V_i^{F_C}}{\partial \hat{V}_j^{C'}} DL_j{}_{C'}^{F_C}$$

and

$$\frac{\partial \vec{V}^{F_C}}{\partial \hat{\vec{V}}^{C(')}} = \frac{1}{2V_1^{F_C}} \begin{bmatrix} V_1^{F_C} & 0 & 0 & 0 & 0 \\ \hat{V}_2^{C(')} - V_2^{F_C} & \hat{V}_1^{C(')} & 0 & 0 & 0 \\ \hat{V}_3^{C(')} - V_3^{F_C} & 0 & \hat{V}_1^{C(')} & 0 & 0 \\ \hat{V}_4^{C(')} - V_4^{F_C} & 0 & 0 & \hat{V}_1^{C(')} & 0 \\ 0 & 0 & 0 & 0 & V_1^{F_C} \end{bmatrix}$$

7.3.3 Differentiation of the Temporal Term

In unsteady applications with moving geometries, the temporal term differentiation needs special treatment. In such cases, the mesh is adjusted at each time iteration, which necessitates the flow transportation from the mesh of the previous time step (n) to the next one ($n+1$) defining an intermediate flow field ($n+1/2$), which is used in the temporal term discretization. This transportation also takes the appearing and disappearing cells into consideration, which is discussed in subsection 3.3.3. From a mathematical perspective, the conservative variables' vector $\vec{U}_{n+\frac{1}{2}}^C$ stored at cell C is a linear combination of flow variables stored at a group of cells from time step n, enumerated by $\bar{C}$. Thus,

$$\vec{U}_{n+\frac{1}{2}}^C = \sum_{\bar{C}} Z_{n+1}{}_{\bar{C}}^{C} \vec{U}_n^{\bar{C}} \tag{7.15}$$

where coefficients $Z_n{}_{\bar{C}}^{C}$ depend only on geometrical quantities. The discretization of the temporal term, shown in eq. 3.21 can be rewritten as

$$\vec{T}_n^C = \frac{\Omega_n^C \vec{U}_n^C - \Omega_{n-\frac{1}{2}}^C \vec{U}_{n-\frac{1}{2}}^C}{\Delta t} \tag{7.16}$$

The discrete adjoint equation of cell C, eq. 7.1, for unsteady flows at time step n is

$$\sum_n \sum_m \vec{\Psi}_n^m \frac{\partial \vec{R}_n^m}{\partial \vec{V}_n^C} + \frac{\partial F}{\partial \vec{V}_n^C} = 0$$

with index m enumerating mesh cells. Therefore, only the flow equations containing the vector $\vec{V}_n^C$ in the discretization of their temporal term contribute to the adjoint equation of C. Apparently, this is true for the term $\vec{T}_n^C$ of equation $\vec{R}_n^C$. Furthermore, $\vec{V}_n^C$ appears in multiple equations of time step $n+1$, enumerated by the index $\hat{C}$, as part of the flow variables formulating the field at $n+1/2$. Then, the adjoint equations' temporal term becomes

$$\vec{T}_n^{A,C} = \vec{\Psi}_n^C \frac{\partial \vec{T}_n^C}{\partial \vec{V}_n^C} + \sum_{\hat{C}} \vec{\Psi}_{n+1}^{\hat{C}} \frac{\partial \vec{T}_{n+1}^{\hat{C}}}{\partial \vec{V}_n^C}$$

The proper differentiation of the above temporal terms yields

$$\vec{T}_n^{A,C} = (W^C)^T \frac{\Omega_n^C \vec{\Psi}_n^C - \Omega_{n+\frac{1}{2}}^C \vec{\Psi}_{n+\frac{1}{2}}^C}{\Delta t}$$

It is reminded that

$$W_{ij}^C = \frac{\partial U_i^C}{\partial V_j^C}$$

It is highlighted that the adjoint field $\vec{\Psi}_{n+\frac{1}{2}}^C$ is not computed similarly to the flow field at the same intermediate step. Instead, the differentiation of the flow equations indicates that

$$\vec{\Psi}_{n+\frac{1}{2}}^C = \frac{1}{\Omega_{n+\frac{1}{2}}^C} \sum_{\hat{C}} \Omega_{n+1/2}^{\hat{C}} \; Z_{n+1}{}_C^{\hat{C}} \; \Psi_{n+1}^{\hat{C}} \tag{7.17}$$

The discretization of the temporal term designates the need for the inverse time integration of the discrete adjoint equations, which totally agrees with the proposed time discretization in section 6.4 for the continuous field adjoint equations.

7.4 Sensitivity Derivatives for Compressible Flows

The sensitivity derivatives of an arbitrary objective function F are given by eq. 7.2, which requires the computation of several quantities listed below. Firstly, $\delta F/\delta b_q$ depend on the derivatives of F w.r.t. $\vec{G}$, the computation of which is usually straightforward. Secondly, the adjoint variables $\vec{\Psi}$ are necessary, which are computed by solving the discrete field adjoint equations shown in section 7.3. Moreover, the $\delta\vec{G}/\delta b_q$ terms are needed, which are given by expressions presented in section 2.9. Finally, the derivatives of the flow equations w.r.t. to $\vec{G}$ are required, the computation of which is the goal of this section.

The term of eq. 7.2 containing the derivatives of the flow residual is developed as

$$\frac{\partial \vec{R}^C}{\partial \vec{G}^m}\frac{\delta \vec{G}^m}{\delta b_q} = \sum_n \frac{\partial \vec{R}^C}{\partial \vec{G}^m}\frac{\partial \vec{G}^m}{\partial \vec{x}^n}\frac{\delta \vec{x}^n}{\delta b_q} \tag{7.18}$$

where $\vec{x}^n$ are the nodes constituting the triangulated solid surface. The first two terms of the r.h.s. can further be decomposed as

$$\begin{aligned}\frac{\partial \vec{R}^C}{\partial \vec{G}^m}\frac{\partial \vec{G}^m}{\partial \vec{x}^n} = &\sum_{C_w}\frac{\partial \vec{R}^C}{\partial \vec{x}^{C_w}}\frac{\partial \vec{x}^{C_w}}{\partial \vec{x}^n} + \sum_F \frac{\partial \vec{R}^C}{\partial \vec{x}^F}\frac{\partial \vec{x}^F}{\partial \vec{x}^n} + \sum_F \frac{\partial \vec{R}^C}{\partial \vec{N}^F}\frac{\partial \vec{N}^F}{\partial \vec{x}^n}\\ &+ \sum_{C_w}\frac{\partial \vec{R}^C}{\partial \Omega^{C_w}}\frac{\partial \Omega^{C_w}}{\partial \vec{x}^n} + \sum_{F_w}\frac{\partial \vec{R}^C}{\partial \vec{v}^{F_w}}\frac{\partial \vec{v}^{F_w}}{\partial \vec{x}^n}\end{aligned} \tag{7.19}$$

Index C_w in the first term sums the contributions of centroids $\vec{x}^{C_w}$ of all cells cut by the wall. The rest of the cell centroids remain intact by the variation of b_q in a Cartesian mesh. The second term describes the flow residual's change due to the variation of face centroids ($\vec{x}^F$). There exist two kinds of faces, the centroid of which are affected by the geometry's shape modification in a Cartesian mesh. In the first kind belong all wall faces, and the second one consists of inner faces cut by the solid boundary. The following term corresponds to the face normal vectors contribution. Once again, only the wall and cut faces participate in the summation. The $\vec{N}$ vector is defined as the normal unit vector $\vec{n}$ multiplied by the area ΔS of each face. The last two terms exist only in unsteady cases, including moving solid bodies where the cut-cells volume Ω^{C_w} and the velocity $\vec{v}^{F_w}$ of each wall face F_w depend on b_q.

The following sections discuss the differentiation of the convection and diffusion terms to result in expressions used to compute the terms included in eq. 7.19.

7.4.1 Differentiation of the Convection Term

The discussion starts from the computation of the first term of eq. 7.19. According to the proposed discretization of the flow equations, a slight change in $\vec{x}^{C_w}$ modifies the primitive variables' spatial derivative at cell C and its neighbors C'. Therefore, the residuals of C, its first and second neighbors are affected, which allows the development of a similar method to the one shown in section 7.2. The equivalent terms to the adjoint flux and B-term are defined as

$$T_\lambda^{inv,C} = \sum_{F_C}\sum_i\sum_j\sum_k \frac{\partial f_{ik}^{inv,F_P}}{\partial \hat{V}_j^C}\left(\Psi_i^P - \Psi_i^Q\right)\tilde{\mathbb{D}}L_{j\lambda}^{F_C} n_k^{F_C}\Delta S^{F_C}$$

$$\tilde{B}_{\lambda\ C'}^{visC} = \sum_{F_C}\sum_i\sum_j\sum_k \frac{\partial f_{ik}^{inv,F_P}}{\partial \hat{V}_j^C}\left(\Psi_i^P - \Psi_i^Q\right)\tilde{D}L_{j\lambda}{}_{C'}^{F_C}\ n_k^{F_C}\Delta S^{F_C}$$

where

$$\tilde{\mathbb{D}}L_{ik}^{F_C} = \frac{\partial \hat{V}_i^{F_C}}{\partial x_k^C} = \tilde{\mathbb{D}}_{ik}^{F_C} - T_{i\bar{F}}^{F_C}\tilde{\mathbb{D}}_{ik}^{\bar{F}_C}$$

$$\tilde{D}L_{ik}{}_{C'}^{F_C} = \frac{\partial \hat{V}_i^{F_C}}{\partial x_k^{C'}} = \tilde{D}_{ik}{}_{C'}^{F_C} - T_{i\bar{F}}^{F_C}\tilde{D}_{ik}{}_{C'}^{F_C}$$

and

$$\tilde{\mathbb{D}}_{ik}^{F_C} = \left.\frac{\partial \hat{V}_i^{F_C}}{\partial x_k^C}\right|_{\vec{\phi}^C} = \phi_i^C\sum_\lambda\sum_{C'}\frac{\partial \mathbb{C}_{\lambda}{}_{C'}^{C}}{\partial x_k^C}\left(V_i^{C'} - V_i^C\right)\Delta x_\lambda^{F_C} - \phi_i^C \eth V_{ik}^C$$

$$\tilde{D}_{ik}{}_{C'}^{F_C} = \left.\frac{\partial \hat{V}_i^{F_C}}{\partial x_k^{C'}}\right|_{\vec{\phi}^C} = \phi_i^C\sum_\lambda\sum_{\hat{C}}\frac{\partial \mathbb{C}_{\lambda}{}_{\hat{C}}^{C}}{\partial x_k^{C'}}\left(V_i^{\hat{C}} - V_i^C\right)\Delta x_\lambda^{F_C}$$

The $T_{\bar{F}}^{F_C}$ term is computed by eq. 7.8. The $\partial \mathbb{C}_{\lambda}{}_{C'}^{C}/\partial x_k^C$ and $\partial \mathbb{C}_{\lambda}{}_{\hat{C}}^{C}/\partial x_k^{C'}$ terms are computed by differentiating the Least Squares Method w.r.t. $\vec{x}^C$. The computation of coefficients $\mathbb{C}_{\lambda}{}_{C'}^{C}$ in the non-weighted version of the method is based on vector $\vec{b}_{C'}^C = \vec{x}^{C'} - \vec{x}^C$ and matrix A^C defined in subsection 3.2.5,

$$A_{k\lambda}^C = \sum_{C'} b_{k}{}_{C'}^{C} b_{\lambda}{}_{C'}^{C}$$

The mathematical development leads to

$$\frac{\partial \mathbb{C}_{\lambda}{}_{C'}^{C}}{\partial x_k^C} = (A^{-1})_{\lambda k}^C\left(\sum_\mu S_\mu^C \mathbb{C}_{\mu}{}_{C'}^{C} - 1\right) + \mathbb{C}_{k}{}_{C'}^{C}\sum_\mu S_\mu^C A_{\lambda\mu}^{-1C}$$

$$\frac{\partial \mathbb{C}_{\lambda}{}_{\hat{C}}^{C}}{\partial x_k^{C'}} = (A^{-1})_{\lambda k}^C\left(\delta_{C'\hat{C}} - \sum_\mu b_{\mu}{}_{C'}^{C}\mathbb{C}_{\mu}{}_{\hat{C}}^{C}\right) - \mathbb{C}_{k}{}_{\hat{C}}^{C}\sum_\mu b_{\mu}{}_{C'}^{C} A_{\lambda\mu}^{-1C}$$

where

$$S_k^C = \sum_{C'} b_{k}{}_{C'}^{C}$$

The corresponding algorithm is

Algorithm 14: Contribution of cell centroids to sensitivity derivatives

```
1 foreach cut-cell $C_w$ do
2     $\frac{\delta F}{\delta b_q} \leftarrow \frac{\delta F}{\delta b_q} + \sum_\lambda T_\lambda^{C_w} \frac{\delta x_\lambda^{C_w}}{\delta b_q}$
3     foreach cell's neighbor $C'$ do
4         $\frac{\delta F}{\delta b_q} \leftarrow \frac{\delta F}{\delta b_q} + \sum_\lambda B_{\lambda}{}_{C_w}^{C'} \frac{\delta x_\lambda^{C_w}}{\delta b_q}$
5     end
6 end
```

Then, the convection differentiation w.r.t. the face centroids is discussed. According to the MUSCL scheme, the second order flux computation requests the flow variables extrapolation from the cells to the faces centroids. Therefore, the centroids' position affects the extrapolated variables and the corresponding flux. Furthermore, centroids of the wall faces also influence the computation of $\mathbb{C}_{\mu}{}_{C'}^{C}$. Both residual dependencies are modeled by introducing a similar to $T_\lambda^{C_w}$ term,

$$T_\lambda^{inv,F_C} = \sum_{F'} \sum_i \sum_j \sum_k \frac{\partial f_{ik}^{inv,F'}}{\partial \hat{V}_j^{F_C}} \left(\Psi_i^P - \Psi_i^Q\right) \tilde{D}L_{j\lambda}{}_{F'}^{F_C} n_k^{F'} \Delta S^{F'} \tag{7.20}$$

where F' enumerates the faces of cell C. The $\tilde{D}L_{F'}^{F_C}$ matrix is computed by differentiating the extrapolation scheme and the Least Squares system,

$$\begin{aligned}
\tilde{D}L_{j\lambda}{}_{F'}^{F_C} &= \frac{\partial \hat{V}_i^{F_C}}{\partial x_k^{F'}} = \tilde{D}_{j\lambda}{}_{F'}^{F_C} - T_{i}{}_{\bar{F}}^{F_C} \tilde{D}_{j\lambda}{}_{F'}^{\bar{F}_C} \\
\tilde{D}_{ik}{}_{F'}^{F_C} &= \left.\frac{\partial \hat{V}_i^{F_C}}{\partial x_k^{F'}}\right|_{\vec{\phi}^C} = \phi_i^C \sum_\lambda \sum_{C'} \frac{\partial \mathbb{C}_{\lambda}{}_{C'}^{C}}{\partial x_k^{F'}} \left(V_i^{C'} - V_i^C\right) \Delta x_k^F + \phi_i^C \text{đ}V_{ik}^C \delta_{F'F} \\
\frac{\partial \mathbb{C}_{\lambda}{}_{C'}^{C}}{\partial x_k^F} &= (A^{-1})_{\lambda k}^C \left(\delta_{C'F} - \sum_\mu b_{\mu}{}_F^C \mathbb{C}_{\mu}{}_{C'}^{C}\right) + \mathbb{C}_{k}{}_{C'}^{C} \sum_\mu b_{\mu}{}_F^C (A^{-1})_{\lambda\mu}^C \\
b_{k}{}_F^C &= x_k^F - x_k^C
\end{aligned}$$

The algorithm which gathers all contributions is

Algorithm 15: Contribution of face centroids to sensitivity derivatives

```
1 foreach wall or cut face F do
2     if inner face then
3         $\frac{\delta F}{\delta b_q} \leftarrow \frac{\delta F}{\delta b_q} + \sum_\lambda T_\lambda^{F_P} \frac{\delta x_\lambda^F}{\delta b_q}$
4         $\frac{\delta F}{\delta b_q} \leftarrow \frac{\delta F}{\delta b_q} + \sum_\lambda T_\lambda^{F_Q} \frac{\delta x_\lambda^F}{\delta b_q}$
5     else
6         $\frac{\delta F}{\delta b_q} \leftarrow \frac{\delta F}{\delta b_q} + \sum_\lambda T_\lambda^{F_C} \frac{\delta x_\lambda^F}{\delta b_q}$
7     end
8 end
```

Subsequently, the normal vector $\vec{N}$ contribution to the sensitivity derivatives is presented. It has already been mentioned that $\vec{N}$ is modified only in the wall and cut faces. However, the vector corresponding to the inner faces of a Cartesian mesh changes only in magnitude, a.k.a. the face area ΔS, significantly simplifying the residual differentiation process. Therefore, the contribution of each inner face to the sensitivity derivatives is

$$\sum_k f_{ik}^F n_k^F \frac{\partial \Delta S}{\partial b_q}$$

The effect of the solid faces' normal vector on the spatial flow derivatives and the limiter of the corresponding cut-cell is expressed through term T_λ^{inv,F_C^w}. It is non-zero only when the slip wall condition is used, and it is computed as

$$\begin{aligned}
T_\lambda^{inv,F_C^w} &= \sum_{F_C}\sum_i\sum_j\sum_k \frac{\partial f_{ik}^{inv,F_C}}{\partial \hat{V}_j^{F_C^w}} \left(\Psi_i^P - \Psi_i^Q\right) \tilde{D}L_{j\lambda}{}_{F_C}^{F_C^w}\, n_k^{F_C} \Delta S^{F_C} \\
\tilde{D}L_{ik}{}_{F_C}^{F_C^w} &= \tilde{D}_{ik}{}_{F_C}^{F_C^w} + \left(\frac{\partial V_i^{F_C^w}}{\partial n_k^{F_C^w}} \delta_{F_C F_C^w} - \tilde{D}_{ik}{}_{\bar{F}_C}^{F_C^w}\right) T_{i\bar{F}}^{F_C} \\
\tilde{D}_{ik}{}_{F_C}^{F_C^w} &= \phi_i^C \sum_\lambda \mathbb{C}_{\lambda}{}_{F_C^w}^{C} \frac{\partial V_i^{F_C^w}}{\partial n_k^{F_C^w}} \Delta x_k^{F_C} \\
\frac{\partial V_i^{F_C^w}}{\partial n_k^{F_C^w}} &= -\frac{1}{\Delta S^{F_C^w}} \left[v_n^C \delta_{ik} + n_i^{F_C^w} (V_{k+1}^C - 2 v_n^{F_C^w} n_k^{F_C^w}) \right]
\end{aligned}$$

where $V_i^{F_C^w}$ is the velocity on the face after the imposition of the no-penetration wall condition and $v_n^{F_C^w} = \sum_k V_{k+1}^C n_k^{F_C^w}$.

7.4.2 Differentiation of the Diffusion Term

The differentiation of the diffusive term of compressible equations w.r.t. geometrical quantities included in its discretization scheme presents similarities with the method discussed in subsection 7.4.1. Terms of eq. 7.19 are successively examined.

The residual differentiation w.r.t. $\vec{G}$ gives rise to the $\vec{T}^{vis}$ term, which is

$$T^{vis,C(F)}_{\lambda} = \sum_{F_C}\sum_{\mu}\sum_{k}\left[\frac{f^{vis,F_C}_{\mu+1,k}}{\partial G^{C(F)}_{\lambda}}\left(\Psi^{P}_{\mu+1}-\Psi^{Q}_{\mu+1}\right)+\frac{f^{vis,F_C}_{5k}}{\partial G^{C(F)}_{\lambda}}\left(\Psi^{P}_{5}-\Psi^{Q}_{5}\right)\right]n^{F_C}_{k}\Delta S^{F_C} \tag{7.21}$$

Vector $\vec{G}$ represents $\vec{x}^{C}$, $\vec{x}^{F}$, $\vec{n}^{F}$, or $\vec{v}^{F^{w}_{C}}$ stored in the cell or face centroids which exemplifies the use of symbol $\vec{G}^{C(F)}$. The unknown derivatives associated with $T^{vis,C(F)}_{\lambda}$ are

$$\begin{aligned}\frac{f^{vis,F_C}_{\mu+1,k}}{\partial G^{C(F)}_{\lambda}} &= \sum_{k}\mu\left(\frac{\partial \text{đ}V^{F_C}_{\mu+1,k}}{\partial G^{C(F)}_{\lambda}}+\frac{\partial \text{đ}V^{F_C}_{k+1,\mu}}{\partial G^{C(F)}_{\lambda}}-\frac{2}{3}\sum_{m}\frac{\partial \text{đ}V^{F_C}_{m+1,m}}{\partial G^{C(F)}_{\lambda}}\delta_{\mu k}\right)\\ \frac{f^{vis,F_C}_{5k}}{\partial G^{C(F)}_{\lambda}} &= \sum_{\mu}\left(V^{F_C}_{\mu+1}\,\tau^{A}_{\mu\lambda k}{}^{F_C}_{C}+\sum_{j}\frac{\partial V^{F_C}_{\mu+1}}{\partial \hat{V}^{F_C}_{j}}\frac{\partial \hat{V}^{F_C}_{j}}{\partial G^{C(F)}_{\lambda}}\tau_{\mu k}\right)+\frac{\partial q^{F_C}_{k}}{\partial G^{C(F)}_{\lambda}}\end{aligned} \tag{7.22}$$

and

$$\begin{aligned}\frac{\partial q^{F_C}_{k}}{\partial G^{C(F)}_{\lambda}} = kT^{F_C}\Bigg[&\frac{1}{V^{F_C}_{5}}\frac{\partial \text{đ}V^{F_C}_{5k}}{\partial G^{C(F)}_{\lambda}}-\frac{1}{V^{F_C}_{1}}\frac{\partial \text{đ}V^{F_C}_{1k}}{\partial G^{C(F)}_{\lambda}}\\ &-\frac{1}{(V^{F_C}_{5})^{2}}\text{đ}V^{F_C}_{5k}\sum_{j}\frac{\partial V^{F_C}_{5}}{\partial \hat{V}^{F_C}_{j}}\frac{\partial \hat{V}^{F_C}_{j}}{\partial G^{C(F)}_{\lambda}}+\frac{1}{(V^{F_C}_{1})^{2}}\text{đ}V^{F_C}_{1k}\sum_{j}\frac{\partial V^{F_C}_{1}}{\partial \hat{V}^{C(F)}_{j}}\frac{\partial \hat{V}^{C(F)}_{j}}{\partial G^{C(F)}_{\lambda}}\\ &+\text{đ}T^{F_C}_{k}\left(\frac{1}{V^{F_C}_{5}}\frac{\partial V^{F_C}_{5}}{\partial \hat{V}^{F_C}_{5}}\frac{\partial \hat{V}^{F_C}_{5}}{\partial G^{C(F)}_{\lambda}}-\frac{1}{V^{F_C}_{1}}\frac{\partial V^{F_C}_{1}}{\partial \hat{V}^{F_C}_{1}}\frac{\partial \hat{V}^{F_C}_{1}}{\partial G^{C(F)}_{\lambda}}\right)\Bigg]\end{aligned} \tag{7.23}$$

The computation of all terms excluding the $\partial \text{đ}V^{F_C}_{ik}/\partial G^{C(F)}_{\lambda}$ derivatives has already been discussed in the previous subsections. The computation of the new terms depends on the geometrical quantity type represented by $\vec{G}^{C(F)}$. Firstly, the residual differentiation w.r.t. the cell centroids is presented, which, similarly to subsection 7.4.1, requires the definition of $\vec{T}^{vis,C}$ and $\vec{\tilde{B}}^{vis\,C}_{\;C'}$. Vector $\vec{T}^{vis,C}$ emerges by substituting index $C(F)$ with C and $G_{\lambda}^{C(F)}$ with x_{λ}^{C} in eq. 7.21, while the B-term is

$$B_{\lambda C'}^{\;C} = \sum_{F_C} \sum_{\mu} \sum_{k} \left[\frac{f_{\mu+1,k}^{vis,F_C}}{\partial x_{\lambda}^{C'}} \left(\Psi_{\mu+1}^{P} - \Psi_{\mu+1}^{Q} \right) + \frac{f_{5k}^{vis,F_C}}{\partial x_{\lambda}^{C'}} \left(\Psi_{5}^{P} - \Psi_{5}^{Q} \right) \right] n_k^{F_C} \Delta S^{F_C}$$

Moreover, the differentiation of the orthogonal correction scheme, eq. 7.11, for inner faces yields the expressions of the remaining unknown terms,

$$\frac{\partial đV_{i\lambda}^{F_C}}{\partial x_k^C} = \frac{\partial đV_{i\lambda}^{w}}{\partial x_k^C} - \frac{\partial \alpha_{\lambda}^{F}}{\partial x_k^C} \left(\sum_{\mu} đV_{i\lambda}^{w} \alpha_{\mu}^{F} - \frac{V_i^Q - V_i^P}{\Delta^{F_C}} \right) - \alpha_{\lambda}^{F} \left[\sum_{\mu} \left(\frac{\partial đV_{i\mu}^{w}}{\partial x_k^C} \alpha_{\mu}^{F} + đV_{i\mu}^{w} \frac{\partial \alpha_{\mu}^{F}}{\partial x_k^C} \right) - \left(V_i^Q - V_i^P \right) \frac{\partial}{\partial x_k^C} \left(\frac{1}{\Delta^{F_C}} \right) \right]$$

where

$$\frac{\partial \alpha_{\lambda}^{F}}{\partial x_k^C} = \frac{s^C}{\Delta^{F_C}} \left(\alpha_{\lambda}^{F} \alpha_{k}^{F} - \delta_{\lambda k} \right), \quad \frac{\partial}{\partial x_k^C} \left(\frac{1}{\Delta^{F_C}} \right) = \frac{s^C}{(\Delta^{F_C})^2} \alpha_k^F,$$

$$\frac{\partial đV_{i\lambda}^{w}}{\partial x_k^C} = w^{F_C} \tilde{\mathbb{D}}_{i\lambda k}^{F_C} + w^{F_C} \frac{b_{k\,F}^{\;C}}{d^C \left(d^P + d^Q \right)} \left(đV_{i\lambda}^{P} - đV_{i\lambda}^{Q} \right) s^C,$$

$$\frac{\partial đV_{i\lambda}^{F_C}}{\partial x_k^{C'}} = w^{F_C} \tilde{D}_{i\lambda k\,C'}^{\;\;F_C} - \alpha_{\lambda}^{F} w^{F_C} \sum_{\mu} \tilde{D}_{i\mu k\,C'}^{\;\;F_C} \alpha_{\mu}^{F}$$

It is reminded that $w^{F_P} \neq w^{F_Q}$, although F_P and F_Q correspond to the same face. Some of the equations presented above are slightly modified when applied to boundary faces. Their corrected terms, shown with a sidebar, are

$$\left. \frac{\partial đV_{i\lambda}^{F_C}}{\partial x_k^C} \right|_{BC} = \frac{\partial đV_{i\lambda}^{F_C}}{\partial x_k^C} + \frac{\alpha_{\lambda}^{F}}{\Delta^{F_C}} \tilde{\mathbb{D}} L_{ik}^{F_C}$$

$$\left. \frac{\partial \alpha_{\mu}^{F}}{\partial x_k^C} \right|_{BC} = \frac{1}{\Delta^{F_C}} \left(\alpha_{\lambda}^{F} \alpha_{k}^{F} - \delta_{\lambda k} \right)$$

$$\left. \frac{\partial}{\partial x_k^C} \left(\frac{1}{\Delta^{F_C}} \right) \right|_{BC} = \frac{1}{(\Delta^{F_C})^2} \alpha_k^F$$

$$\left. \frac{\partial đV_{i\lambda}^{w}}{\partial x_k^C} \right|_{BC} = \tilde{\mathbb{D}}_{i\lambda k}^{F_C}$$

Finally,

$$\begin{aligned}
\tilde{\mathbb{D}}^{F_C}_{i\lambda k} &= \sum_{C'} \frac{\partial \mathbb{C}_{\lambda C'}^{\,C}}{\partial x_k^{C}} \left(V_i^{C'} - V_i^{C} \right) \\
\tilde{D}_{i\lambda k C'}^{\;F_C} &= \sum_{\hat{C}} \frac{\partial \mathbb{C}_{\lambda \hat{C}}^{\,C}}{\partial x_k^{C'}} \left(V_i^{\hat{C}} - V_i^{C} \right)
\end{aligned} \tag{7.24}$$

Algorithm 14 is also applied for gathering the cell centroids' contributions of the diffusion discretization scheme to the sensitivity derivatives.

The substitution of $\vec{G}$ with $\vec{x}^{F'}$ and of $C(F)$ with F' in eqs. 7.21, 7.22, and 7.23 gives rise to the T_λ^{vis,F_C} term. The differentiation of the orthogonal correction scheme w.r.t. the face centroids gives

$$\begin{aligned}
\frac{\partial \text{đ}V_{i\lambda}^{F_C}}{\partial x_k^{F'}} = \frac{\partial \text{đ}V_{i\lambda}^{w}}{\partial x_k^{F'}} - \frac{\partial \alpha_\lambda^F}{\partial x_k^{F'}} \left(\sum_\mu \text{đ}V_{i\lambda}^{w} \alpha_\mu^F - \frac{V_i^Q - V_i^P}{\Delta^{F_C}} \right) \\
- \alpha_\lambda^F \left[\sum_\mu \left(\frac{\partial \text{đ}V_{i\mu}^{w}}{\partial x_k^{F'}} \alpha_\mu^F + \text{đ}V_{i\mu}^{w} \frac{\partial \alpha_\mu^F}{\partial x_k^{F'}} \right) - \left(V_i^Q - V_i^P \right) \frac{\partial}{\partial x_k^{F'}} \left(\frac{1}{\Delta^{F_C}} \right) \right] \\
+ \left(\frac{\alpha_\lambda^F}{\Delta^{F_C}} \tilde{\mathbb{D}} L_{ik}^{F_C} \right)_{BC}
\end{aligned}$$

The last term appears in case F_C is part of the mesh boundary. The computation of the derivatives that appeared on the r.h.s. differs between inner and boundary faces. In the subsequent expressions, the two cases are denoted as $F\!:\!I$ and $F\!:\!B$,

$$\frac{\partial \alpha_\lambda^F}{\partial x_k^{F'}} = \begin{cases} 0, & F:I \\ -\frac{s^C}{\Delta^{F_C}} \left(\alpha_\lambda^F \alpha_k^F - \delta_{\lambda k} \right) \delta_{F'F}, & F:B \end{cases}$$

$$\frac{\partial}{\partial x_k^{F'}} \left(\frac{1}{\Delta^{F_C}} \right) = \begin{cases} 0, & F:I \\ -\frac{s^C}{(\Delta^{F_C})^2} \alpha_k^F \delta_{F'F}, & F:B \end{cases}$$

$$\frac{\partial \text{đ}V_{i\lambda}^{w}}{\partial x_k^{F'}} = \begin{cases} \frac{\partial w^{F_C}}{\partial x_k^{F'}} \left(\text{đ}V_{i\lambda}^{P} - \text{đ}V_{i\lambda}^{Q} \right) s^C, & F:I,\ F':I \\ w^{F_C} \frac{\partial \text{đ}V_{i\lambda}^{C}}{\partial x_k^{F'}}, & F:I,\ F':B \\ 0, & F:B,\ F':I \\ \frac{\partial \text{đ}V_{i\lambda}^{C}}{\partial x_k^{F'}}, & F:B,\ F':B \end{cases}$$

where

$$
\begin{aligned}
\frac{\partial w^{F_C}}{\partial x_k^{F'}} &= \frac{1}{d^P + d^Q}\left[w^{F_P}\frac{1}{d^P}\left(x_k^P - x_k^F\right) - w^{F_Q}\frac{1}{d^Q}\left(x_k^Q - x_k^F\right)\right] s^C \delta_{F'F} \\
\frac{\partial đV_{i\lambda}^C}{\partial x_k^{F'}} &= \tilde{D}L_{i\lambda k}{}_{F'}^{F_C}
\end{aligned}
$$

Matrix components $\tilde{D}L_{i\lambda k}{}_{F'}^{F_C}$ are computed similarly to eq. 7.24.

Lastly, the diffusion derivative w.r.t. the normal vector component of each wall and cut face is presented. In the case of inner faces, the differentiation is simple and is explained in subsection 7.4.1. On the other hand, term T_λ^{vis,F_C^w} represents the wall faces contribution, and it is computed by substituting $\vec{G}$ with $\vec{n}^{F_C^w}$ and $C(F)$ with F_C^w in eqs. 7.21, 7.22 and 7.23. The emerging derivatives necessary for the T_λ^{vis,F_C^w} computation are

$$
\begin{aligned}
\frac{\partial đV_{i\lambda}^{F_C^w}}{\partial n_k^{F_C^w}} &= -\frac{\alpha_\lambda^{F_w}}{\Delta^{F_w}}\frac{\partial V_i^{F_C^w}}{\partial n_k^{F_C^w}} \\
\frac{\partial V_i^{F_C^w}}{\partial n_k^{F_C^w}} &= \sum_j Q_{ij}^{F_C^w}\tilde{D}L_{jk}{}_{F_C^w}^{F_C^w}
\end{aligned}
$$

where $Q_{ij}^{F_C^w}$ is the derivative of the variables computed by imposing the wall condition w.r.t. the flow variables extrapolated from the cell centroid to the face.

7.4.3 Differentiation of the Unsteady flow Equations

The geometrical complexities caused by the motion of a solid body within a stationary Cartesian mesh perplex the differentiation of the temporal term. Based on its discretization shown in eq. 7.16, only the $Z_{n\bar{C}}^{C}$ coefficients, defined in eq. 7.15, depend directly on geometrical quantities. Their differentiation gives the corresponding contributions to the sensitivity derivatives.

Firstly, it is stated that although the $Z_{n\bar{C}}^{C}$ coefficients are not continuous functions w.r.t. time, they are differentiable w.r.t. the design variables b_q. According to eqs. 3.24 and 3.25, $Z_{n\bar{C}}^{C}$ depend only on the cells' volume and index k_n^C, which stands for the number of cells from time step $n{+}1$ affecting the flow variables of C. Thus, k_n^C remains constant during an infinitesimal change of b_q. Let $T_n^{temp,C}$ be the derivative of the temporal term w.r.t. the volume of C at time step n (Ω_n^C). After the proper mathematical development, the term is computed as

$$T_n^{temp,C} = \frac{\vec{\Psi}_n^C - \vec{\bar{\Psi}}_{n+1}^C}{\Delta t} \cdot \vec{U}_n^C$$

where

$$\vec{\bar{\Psi}}_{n+1}^C = \frac{1}{k_{n+1}^C} \sum_{\hat{C}=1}^{k_{n+1}^C} \vec{\Psi}_{n+1}^{\hat{C}} \tag{7.25}$$

A significant impact of the unsteady residual to the sensitivity derivatives emerges from the use of the geometries' velocity ($\vec{v}^{F_C^v}$) in the no-penetration and/or no-slip conditions imposed along the wall. The flow variables' spatial derivatives and the limiters stored at cut-cells' centroids and the spatial derivatives computed at their faces also depend on $\vec{v}^{F_C^v}$. The extra contributions are called T_λ^{inv,F_C^v} and T_λ^{vis,F_C^v} and represent the convective and diffusive flux derivatives w.r.t. the wall velocity. Starting from the convective flux,

$$T_\lambda^{inv,F_C^v} = \sum_{F_C}\sum_i\sum_j\sum_k \frac{\partial f_{ik}^{inv,F_C}}{\partial \hat{V}_j^{F_C^v}} \left(\Psi_i^P - \Psi_i^Q\right) \tilde{D}L_{j\lambda}{}_{F_C}^{F_C^v}\, n_k^{F_C} \Delta S^{F_C} + p^{F_C}\Psi_5^C\, n_\lambda^{F_C} \Delta S^{F_C}$$

$$\tilde{D}L_{ik}{}_{F_C}^{F_C^v} = \tilde{D}_{ik}{}_{F_C}^{F_C^v} + \left(\frac{\partial V_i^{F_C^v}}{\partial v_k^{F_C^v}} \delta_{F_{\bar{C}} F_C^v} - \tilde{D}_{ik}{}_{\bar{F}_C}^{F_C^v} \right) T_i{}_{\bar{F}}^{F_C}$$

$$\tilde{D}_{ik}{}_{F_C}^{F_C^v} = \phi_i^C \sum_\lambda \mathbb{C}_{\lambda}{}_{F_C^v}^{C} \frac{\partial V_i^{F_C^v}}{\partial v_k^{F_C^v}} \Delta x_k^{F_C}$$

The $\partial V_i^{F_C^v}/\partial v_k^{F_C^v}$ derivative depends on the chosen wall condition. For slip walls,

$$\frac{\partial V_i^{F_C^v}}{\partial v_k^{F_C^v}} = \begin{cases} 0, & i = 1,5 \\ n_{i-1} n_k, & i = 2,3,4 \end{cases}$$

and for no-slip walls,

$$\frac{\partial V_i^{F_C^v}}{\partial v_k^{F_C^v}} = \begin{cases} 0, & i = 1,5 \\ \delta_{i-1,k}, & i = 2,3,4 \end{cases}$$

The expression for computing T_λ^{vis,F_C^v} is based on eqs. 7.21, 7.22 and 7.23 and is derived by substituting $\vec{G}$ with $\vec{v}^{F_w^C}$ and $C(F)$ with F_C^v and adding the term, $\Psi_5^C \sum\limits_k \tau_{\lambda k} n_k^{F_C} \Delta S^{F_C}$. The differentiation of the orthogonal correction scheme w.r.t. $\vec{v}^{F_C^v}$

leads to different expressions for inner or boundary faces,

$$\frac{\partial đV_{i\lambda}^{F_C}}{\partial V_k^{F_C^v}} = \begin{cases} w^{F_C}\mathbb{C}_{\lambda}{}^{C}_{F_C^v}\frac{\partial V_i^{F_C^v}}{\partial v_k^{F_C^v}} - \alpha_\lambda^F w^{F_C}\sum_\mu \mathbb{C}_{\mu}{}^{C}_{F_C^v}\frac{\partial V_i^{F_C^v}}{\partial v_k^{F_C^v}}\alpha_\mu^F, & F_C : I \\ \mathbb{C}_{\lambda}{}^{C}_{F_C^v}\frac{\partial V_i^{F_C^v}}{\partial v_k^{F_C^v}} - \alpha_\lambda^F \sum_\mu \mathbb{C}_{\mu}{}^{C}_{F_C^v}\frac{\partial V_i^{F_C^v}}{\partial v_k^{F_C^v}}\alpha_\mu^F + \frac{\alpha_\lambda^F}{\Delta^{F_C}} Q_{ik}{}^{F_C}_{F_C^v}, & F_C : B \end{cases}$$

where

$$Q_{ik}{}^{F_C}_{F_C^v} = \tilde{D}L_{ik}{}^{F_C^v}_{F_C} + \delta_{F_C^v,F_C}\frac{\partial V_i^{F_C^v}}{\partial v_k^{F_C^v}}$$

7.5 The Incompressible Discrete Adjoint Equation and Sensitivities

The derivation of the incompressible discrete adjoint equations and sensitivities is based on the differentiation of the discretized incompressible flow equations, which is presented in section 3.5 and shares many similarities with the discretization of the compressible equations. The development of the corresponding adjoint equations takes advantage of these similarities embracing most of the analysis presented in subsections 7.3.1 and 7.4.1. Hence, this section mainly deals with the indication of the differences between the two variants.

According to eq. 7.1, the development of the adjoint formulation is based on the differentiation of the incompressible flow equations w.r.t. vector $\vec{V}$ defined in eq 3.26. Regarding the convective terms, all the equations of subsection 7.3.1 are still valid except for eq. 7.9, which is rewritten as

$$\begin{aligned} \frac{\partial \bar{f}_{ik}^{inv,F_P}}{\partial \hat{V}_j^P} n_k^F &= \sum_k \bar{A}_{ijk}^{\Gamma,P} n_k^F + \frac{1}{2}\sum_l\sum_k |\tilde{A}_{ijk}^{\Gamma} n_k^F| - \frac{1}{2} A_{ij}^{d,\Gamma,P} \\ \frac{\partial \bar{f}_{ik}^{inv,F_P}}{\partial \hat{V}_j^Q} n_k^F &= \sum_k \bar{A}_{ijk}^{\Gamma,Q} n_k^F - \frac{1}{2}\sum_l\sum_k |\tilde{A}_{ijk}^{\Gamma} n_k^F| - \frac{1}{2} A_{ij}^{d,\Gamma,Q} \end{aligned} \tag{7.26}$$

where $\bar{A}^\Gamma$ emerges from the differentiation of the preconditioned incompressible flux w.r.t. $\vec{V}$ and $\tilde{A}^\Gamma$ is defined in eq. 3.28. Matrix $A^{d,\Gamma}$ is computed by differentiating the absolute Jacobian matrix w.r.t. $\vec{V}$, which is presented in Appendix P. The artificial compressibility parameter (β) is considered independent of the design variables, and, thus, it remains constant during the optimization process. A main difference between the continuous and discrete adjoint versions is that eqs. 7.26 already incorporates the preconditioned matrix effects, in contrast with the continuous equations at which a new adjoint preconditioned matrix should be introduced during their discretization.

Subsequently, the diffusion differentiation is studied. Eqs. 7.13 and 7.14 should be reintroduced,

$$f_{jk}^{vis,A,F_C} = \sum_{\lambda} \tau_{\lambda jk}^{A}{}_{C}^{F_C} \left(\Psi_{\lambda+1}^{P} - \Psi_{\lambda+1}^{Q} \right)$$

$$B_{j}{}_{C'}^{C} = \sum_{F_C} \sum_{\lambda} \sum_{k} \tau_{\lambda jk}^{A}{}_{C'}^{F_C} \left(\Psi_{\lambda+1}^{P} - \Psi_{\lambda+1}^{Q} \right) n_k^{F_C} \Delta S^{F_C}$$

where

$$\tau_{\lambda jk}^{A}{}_{C(')}^{F_C} = \mu \left(\frac{\partial \tilde{d} V_{\lambda+1,k}^{F_C}}{\partial V_j^{C(')}} + \frac{\partial \tilde{d} V_{k+1,\lambda}^{F_C}}{\partial V_j^{C(')}} \right)$$

The absence of an equation representing the energy conservation in incompressible flows significantly simplifies the above expressions.

The differentiation of the temporal term yields

$$\vec{T}_n^{A,C} = M \frac{\Omega_n^C \vec{\Psi}_n^C - \Omega_{n+\frac{1}{2}}^C \vec{\Psi}_{n+\frac{1}{2}}^C}{\Delta t}$$

where M is defined in eq. 3.26 and $\vec{\Psi}_{n+\frac{1}{2}}^C$ is given by eq. 7.17.

Regarding the sensitivity derivatives, some modifications should also be made to the compressible version presented in section 7.4. The expressions resulted from the convection differentiation remain valid as long as the incompressible flux ($\vec{f}_k^{inv,\Gamma,F_P}$) and variables ($\vec{V}^C$) are used. Also, terms containing the adjoint energy Ψ_5 are neglected. On the contrary, some changes are needed in the diffusive terms. Initially, $\vec{T}^{vis}$ and $B_{\lambda}{}_{C'}^{C}$ become

$$T_{\lambda}^{vis,C(F)} = \sum_{F_C} \sum_{\mu} \sum_{k} \frac{f_{\mu+1,k}^{vis,F_C}}{\partial G_{\lambda}^{C(F)}} \left(\Psi_{\mu+1}^{P} - \Psi_{\mu+1}^{Q} \right) n_k^{F_C} \Delta S^{F_C}$$

where

$$\frac{f_{\mu+1,k}^{vis,F_C}}{\partial G_{\lambda}^{C(F)}} = \sum_{k} \mu \left(\frac{\partial \tilde{d} V_{\mu+1,k}^{F_C}}{\partial G_{\lambda}^{C(F)}} + \frac{\partial \tilde{d} V_{k+1,\mu}^{F_C}}{\partial G_{\lambda}^{C(F)}} \right)$$

and

$$B_{\lambda C'}^{\ C} = \sum_{F_C} \sum_{\mu} \sum_{k} \frac{f_{\mu+1,k}^{vis,F_C}}{\partial x_{\lambda}^{C'}} \left(\Psi_{\mu+1}^{P} - \Psi_{\mu+1}^{Q} \right) n_k^{F_C} \Delta S^{F_C}$$

Finally, the differentiation of the temporal term multiplied by the preconditioner matrix leads to

$$T_n^{temp,C} = \frac{\vec{\Psi}_n^C - \vec{\bar{\Psi}}_{n+1}^C}{\Delta t} \cdot \vec{U}_n^{\Gamma,C}$$

where $\vec{U}_n^{\Gamma,C} = (\beta^2, v_1, v_2, v_3)_n^C$ and $\vec{\bar{\Psi}}_{n+1}^C$ is given by eq. 7.25.

7.6 The Sensitivity Map Post-Processing

The parameterization of the examined geometry is of great importance for the optimization process. The target of a parameterization tool is to control the position of the surface nodes constituting the discretized geometry by handling a much smaller number of design variables, ensuring zero, first, or even second order smoothness in most of the optimized surface parts. A common parameterization technique for 3D industrial applications is based on CAD software, which provide the best counterbalance between complexity and manufacturability. In case the geometry's optimization is driven by a gradient-based method, the parameterization differentiation is required for computing the $\delta \vec{x}^n / \delta b_q$ term included in eq. 7.18. However, the parameterization's differentiation is usually a challenging enough process, especially when a CAD package is used. Moreover, a parameterization tool is not always available. These drawbacks can be surpassed by assuming that every surface node can move independently from the rest and, thus, its coordinates are design variables. Therefore,

$$\frac{\delta x_k^n}{\delta b_q} = \begin{cases} 1, & x_k^n = b_q \\ 0, & x_k^n \neq b_q \end{cases}$$

Then, the objective function's gradient represents a vector at each surface node pointing to the direction in which the node's minimal motion maximizes the objective's improvement. However, any infinitesimal node's movement, tangent to a smooth surface, does not change its shape. Thus, only the gradient's normal component, $(\partial F / \partial \vec{x}^m) \cdot \vec{n}^m$, must be considered. This scalar field plotted on the geometry's surface is called the sensitivity map and highlights the areas characterized by high absolute valued sensitivities, where aerodynamic or hydrodynamic improvement has the most significant potential. Hence, it can be used not only in automated gradient-based optimization methods but also as a tool giving valuable information to the designer. Nevertheless, its computation can be costly since the number of design variables can be equal to 10^4 or even 10^5 for industrial 3D cases illustrating the great advantage of the adjoint-based methods, which can compute the sensitivity map at a cost independent of the surface nodes' number and comparable to the flow simulation's cost.

However, the use of a sensitivity map in an automated optimization process presents some disadvantages. Most significantly, the independent displacement of each node may result in raffled surfaces or even in surfaces with invalid elements causing the optimization to fail. In practice, this phenomenon is amplified by the high-frequency content of the sensitivity field. These signals are caused mainly by the geometry's and computational domain's discretization. A remedy to this problem is smoothing the sensitivity map before using it to deform the shape, although such a process sacrifices the map's accuracy and, potentially, slows down the optimization progress.

The used approach combines an algorithm imitating the diffusive effect of an elliptic equation solution on the deformable surface with the implementation of filters that detect and reduce the sensitivity field's extreme values, diminishing the production of invalid mesh elements after each surface deformation, and thus, facilitating the optimization process completion. Consequently, even though such an approach seems parameterization-free, it actually prohibits the independent motion of the surface nodes forcing them to interact with each other, implying some type of shape parameterization.

The mistraction of the optimization process by the smoothing operation is partly avoided by firstly allowing the optimization algorithm to propose the new position of each node ($\vec{x}^m_{new}$) using the exact derivative computed by the adjoint method. Then, the field $s^m = (\vec{x}^m_{new} - \vec{x}^m_{old}) \cdot \vec{n}^m_{old}$ is smoothed out, where the subscript "*old*" denotes the geometry at the current optimization step. It has also been observed that refining the s instead of the $(\partial F/\partial \vec{x}^m) \cdot \vec{n}^m$ field produces smoother shapes.

The developed method is divided into two steps. Firstly, a filter is applied, which reduces the gradient of s in areas where its value exceeds a user-defined threshold. Then, a smoothing is used, substituting s^m at the m^{th} node with the mean value of its neighbors. The already processed by the filter field allows for an improved spreading of the diffusion caused by the smoothing, preventing the distortion of significant surface areas from existing distinctive spikes. The whole process is described by Algorithm 16.

Algorithm 16: The Sensitivity Map Post-Processing

```
1 foreach surface node $m$ do
2     $\vec{x}_{new} \leftarrow$ optimization method $(m,\ \vec{x}^m_{old})$
3     $s^m = (\vec{x}^m_{new} - \vec{x}^m_{old}) \cdot \vec{n}^m_{old}$
4 end
5 decrease extrema $(s)$
6 smooth field $(s)$
7 foreach surface node $m$ do
8     $\vec{x}^m_{new} \leftarrow \vec{x}^m_{old} + s^m \vec{n}^m_{old}$
9 end
```

Subsequently, the function called "decrease extrema" is described. Firstly, all nodes having s^m greater than their neighbors are detected. In other words, the local maxima or minima are identified. For each node corresponding to a local maximum, the neighboring node with the lowest field's value is found. Let index m_n denote this neighbor for each node m. Then, the gradient

$$g^m = \frac{s^m - s^{m_n}}{|\vec{x}^m - \vec{x}^{m_n}|}$$

is computed. If g^m is greater than a predefined threshold ($\bar{g}$), s^m should be lowered enough to produce a gradient equal to $\bar{g}$. However, this requirement may cause the change in sign of s^m, which may harm the optimization process, and thus, should be prevented, leading to the formula

$$s^m_{new} = \begin{cases} s^{m_n} + \bar{g}|\vec{x}^m - \vec{x}^{m_n}|, & s^m\left(s^{m_n} + \bar{g}|\vec{x}^m - \vec{x}^{m_n}|\right) > 0 \\ 0, & \text{otherwise} \end{cases}$$

for the m^{th} node and

$$s^{m_n}_{new} = \begin{cases} s^{m_n}, & s^m\left(s^{m_n} + \bar{g}|\vec{x}^m - \vec{x}^{m_n}|\right) > 0 \\ -\bar{g}|\vec{x}^m - \vec{x}^{m_n}|, & \text{otherwise} \end{cases}$$

for its neighbor. A similar procedure is used for the modification of the minima.

After the treatment of all extrema, the process is repeated until the elimination of all peculiar spikes is achieved.

The structure of the function “smooth field” is simpler. For each surface node the algorithm computes its neighbors' mean value ($\bar{s}^m$). Then, the corrected field becomes

$$s^m_{new} = \begin{cases} \bar{s}^m, & s^m \bar{s}^m > 0 \\ 0.1 s^m, & \text{otherwise} \end{cases}$$

Values s^m_{new} are firstly computed at all surface nodes and, then, they replace the corresponding s^m values. The process can be repeated multiple times. The larger the number of iterations, the smoother the sensitivity map, and the higher the deviation from its initial proper values. The suitable number of iterations varies depending on the current application.

The above-presented post-processing has successfully been used in several industrial cases with compressible and incompressible flows. Chapter 9 presents applications, including the optimization of 3D ducts, wings, and pumps.

Chapter 8

Adjoint Solver Assessment

In this chapter, the developed adjoint method's ability to compute accurate derivatives is demonstrated in various cases. The descretization of the adjoint equations is based on the governing equations hand-differentiation, presented in chapter 7. The computed sensitivity derivatives are compared with a central Finite Difference (FD) scheme, considered to be the reference values. The adjoint software's assessment refers to compressible or incompressible fluid flows around stationary or moving geometries. After confirming the computed derivatives' accuracy, a shape optimization is carried out using the Conjugate Gradient method (CG). The target is to maximize the lift generated by an isolated airfoil being the optimization's initial shape in all cases presented below. The airfoil is parameterized using two Bézier–Bernstein (or simply Bézier) curves representing the pressure and suction side. The coordinates of their control points are the design variables of the optimization process, excluding the points corresponding to the leading and trailing edge ensuring that the airfoil's chord length remains unchanged during the optimization. Prompted by these cases, an illustration of the adjoint variables' physical meaning is also discussed. It is important to make clear that the chapter's content focuses on examining the derivatives' accuracy and the optimization abilities of the developed software and does not aim to deliver solutions to practical aerodynamic shape optimization problems, which is the subject of chapter 9. Thus, the optimization in the presence of constraints, such as the airfoil's drag coefficient, is beyond the purpose of this chapter.

8.1 Incompressible Adjoint Solver Assessment

The first case examined refers to the lift maximization of an isolated NACA0012 airfoil exposed to an incompressible flow of $Re_\infty = 1000$ and zero angle-of-attack. The objective function is given as

$$L = \int_S (pn_i - \tau_{ij}n_j)r_i dS$$

where the notation of section 3.1 is adopted. Additionally, S is the airfoil's boundary and $\vec{r} = (0, 1)$. The airfoil's geometry and Bézier control polygon are shown in fig. 8.1. The convergence of the flow and adjoint equations is plotted in fig. 8.2.

This application is the perfect example to illustrate the physical meaning of the adjoint variables. Firstly, it is useful to note that the adjoint velocity ($\vec{\Psi}_v$) magnitude is higher close to the airfoil's contour and almost zero at far-field, fig. 8.3, meaning that a small perturbation in the velocity field in the solid boundary's vicinity has a severe impact on the generated lift value. Moreover, the imposed boundary conditions make the adjoint momentum flux to exit from the pressure side and reenter at the suction side. The adjoint velocity direction, depicted by the streamlines shown in fig. 8.3, contains essential information concerning the flow field's effect on lift value configuration. A small force $\delta\vec{f}$ with a direction parallel and opposite to the positive x-axis, implemented to particles close to the pressure side, decelerates the flow, increasing both pressure and lift. The effect of an infinitesimally small force on the lift is described by the adjoint field as well. The analysis of section 7.1 concludes that the lift variation is expressed as $\delta L = -\vec{\Psi}_v \cdot \delta\vec{f}$. The main x-component of the adjoint velocity close to the pressure side is positive, so $\vec{\Psi}_v \cdot \delta\vec{f} < 0$ meaning that $\delta L > 0$, which mathematically expresses the lift's increase due to the force exertion. Similarly, the imposition of $\delta\vec{f}$ close to the suction side causes lift to drop, which is also expressed by the negative x-component sign of the adjoint velocity on that area.

The sensitivity derivatives computed by the proposed adjoint method are compared with FDs, fig. 8.4. Due to the symmetry of the case, derivatives are symmetrical too. Derivatives w.r.t. the x-coordinates of the control points over airfoil's sides are opposite, while derivatives w.r.t. y-coordinates coincide. Overall, both x and y derivatives perfectly match the FDs, verifying the adjoint solver's high accuracy. More specifically, the deviation between the two methods is less than $10^{-4}\%$, meaning that at least five significant digits are correctly computed. It should be kept in mind that the slight inaccuracies of FDs are caused by round-off errors and dependence on the step size choice. Fig. 8.1 presents the gradients of all control points. After 20 optimization cycles, both airfoil sides are cambered, fig. 8.6, causing an increase in lift, as shown in fig. 8.5. Fig. 8.7 compares the pressure field before and after the optimization. The geometry's change causes a reduction in pressure over the airfoil upper side, leading to increased lift.

In the absence of constraints, running the optimization for 20 cycles is enough to display the cut-cell adjoint method's ability to continuously increase airfoil's lift. As expected, fig. 8.5 shows that there is still enough room for improvement. By continuing the optimization for 80 more cycles, fig. 8.8, lift increases by a factor of 8. After around 100 cycles, the flow becomes unsteady, and this is why optimization terminates there. The airfoil's new "abnormal" shape is presented in fig. 8.9. Its intense deformation confirms the mesh ability to handle any resulting shape, verifying the cut-cell method superiority, from this point of view, against other CFD methods used in optimization problems. A clear message is that, the proposed method overcomes difficulties associated with mesh deformation in optimization loops using body-fitted CFD, being a a well-known reason of premature termination.

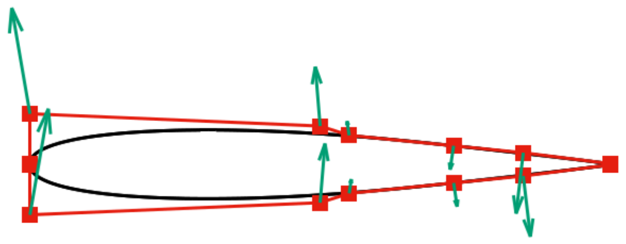


Figure 8.1: NACA0012 airfoil optimization, laminar flow, incompressible fluid: Bézier control polygons (red) separately generating the two sides of the baseline airfoil (black). Green vectors represent the computed gradient at each control point of the initial geometry.

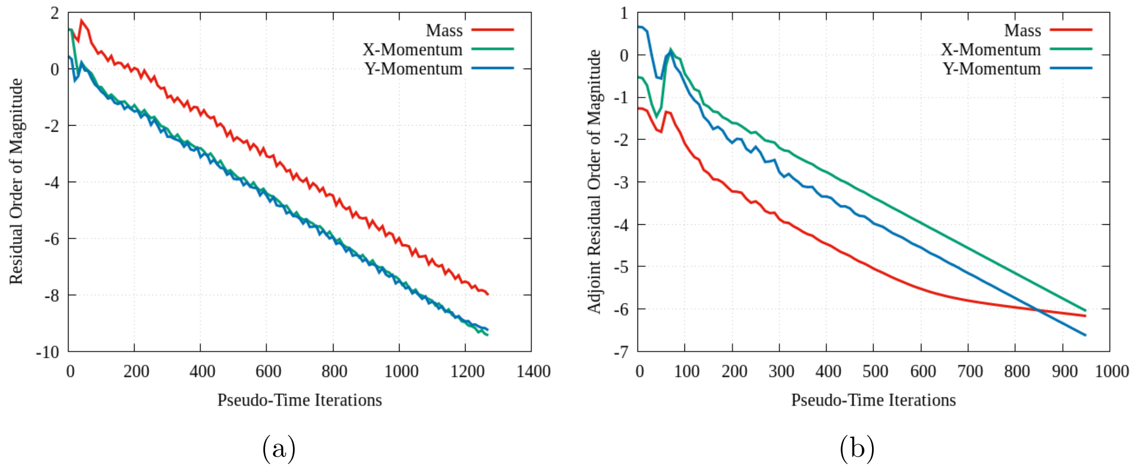


Figure 8.2: NACA0012 airfoil optimization, laminar flow, incompressible fluid: Convergence of the residuals of the (a) flow and (b) adjoint equations.

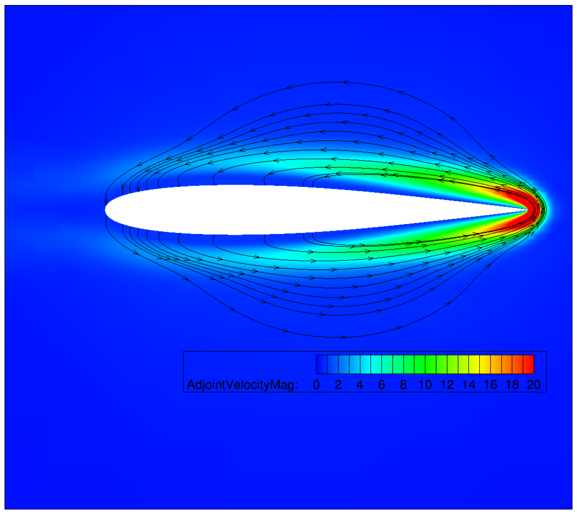


Figure 8.3: NACA0012 airfoil optimization, laminar flow, incompressible fluid: Adjoint iso-velocity contours and adjoint streamlines.

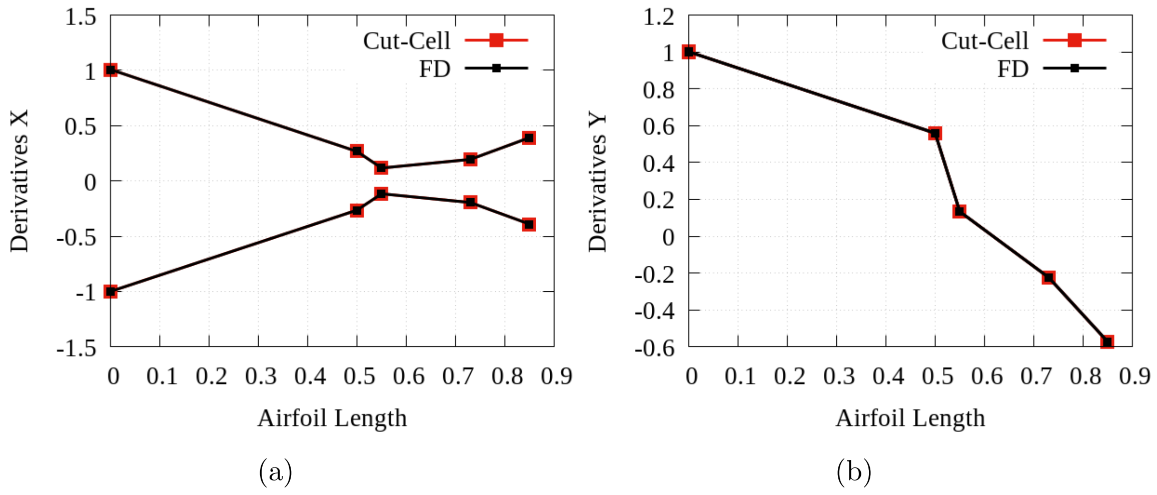


Figure 8.4: NACA0012 airfoil optimization, laminar flow, incompressible fluid: Sensitivity derivatives computed w.r.t. the x and y coordinates of the control points. Comparison between adjoint derivatives (red) and FDs (black).

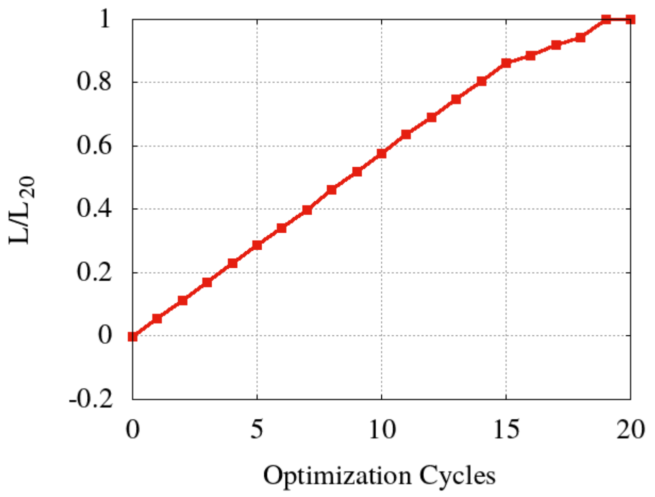


Figure 8.5: NACA0012 airfoil optimization, laminar flow, incompressible fluid: Adjoint-based optimization for lift maximization. To increase the readability of the plot, lift is non-dimensionalized by its value at the 20th cycle.

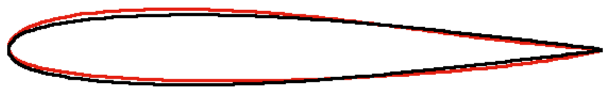


Figure 8.6: NACA0012 airfoil optimization, laminar flow, incompressible fluid: Comparison between the baseline (black) and optimized (red) airfoil after 20 cycles.

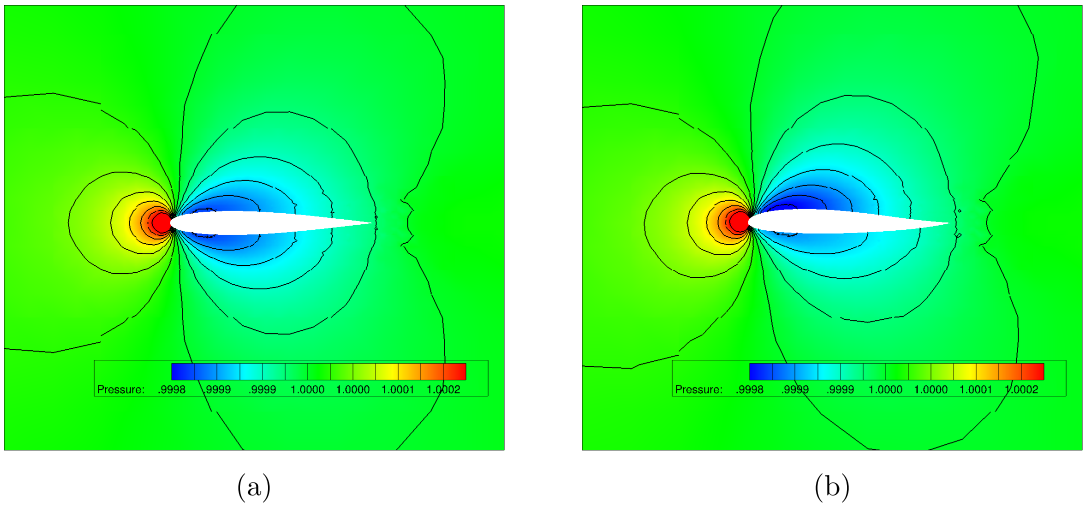


Figure 8.7: NACA0012 airfoil optimization laminar flow, incompressible fluid: Isobar areas of the (a) baseline and (b) optimized airfoil after 20 cycles.

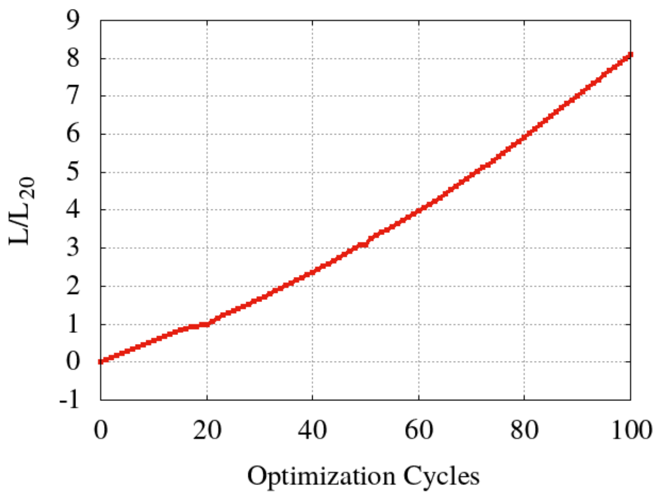


Figure 8.8: NACA0012 airfoil, optimization laminar flow, incompressible fluid: Adjoint-based optimization for lift maximization. Lift is non-dimensionalized by its value at the 20th cycle, which corresponds to lift final value during the optimization presented in fig. 8.5.

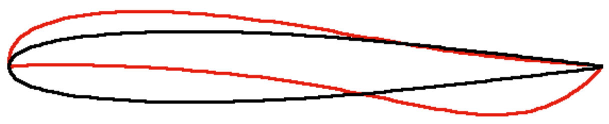


Figure 8.9: NACA0012 airfoil optimization, laminar flow, incompressible fluid: Comparison between the baseline (black) and optimized (red) airfoils after 100 cycles.

8.2 Compressible Adjoint Solver Assessment

The second application is concerned with the optimization of an isolated NACA0012 airfoil exposed to a laminar compressible flow of $M_\infty = 0.293$, $Re_\infty = 1000$, $Pr_\infty = 0.7$ and zero angle-of-attack. The airfoil's parameterization and design variables definition is presented in section 8.1. The flow and adjoint equation's solution terminates when the 3000 pseudo-time steps criterion is met. Their convergence is plotted in fig. 8.11. The adjoint momentum magnitude field and streamlines, shown in fig. 8.12, agree with the analysis presented in section 8.1 regarding the interpretation of the adjoint variables fields. The CG method drives the optimization towards the optimum. The optimization stops prematurely after completing 20 cycles, which was the available computational budget set for the run. Fig. 8.14 presents the corresponding convergence history. The objective function's evaluation remains smooth even though more than 200 cells appear or disappear from the fluid domain due to the airfoil's deformation at each cycle, indicating that the abrupt and non-differentiable mesh topology modification does not hinder the optimization process.

Fig. 8.15 compares the baseline and optimized airfoil's contour. Curvature has increased along the suction side, accelerating the regional flow, while the pressure side has been considerably flattened. The new shape allows for the pressure difference growth between the airfoil's sides, increasing thus lift. Fig. 8.16 illustrates the above statements by comparing the baseline and optimized pressure fields. Sensitivity derivatives, computed by the adjoint method, are plotted in fig. 8.10. A comparison with FDs, presented in fig. 8.13, proves the high accuracy of the computational gradient, with the resulting error being smaller than $10^{-4}\%$.

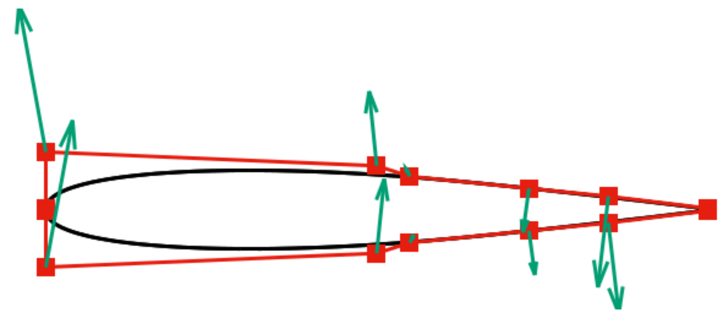
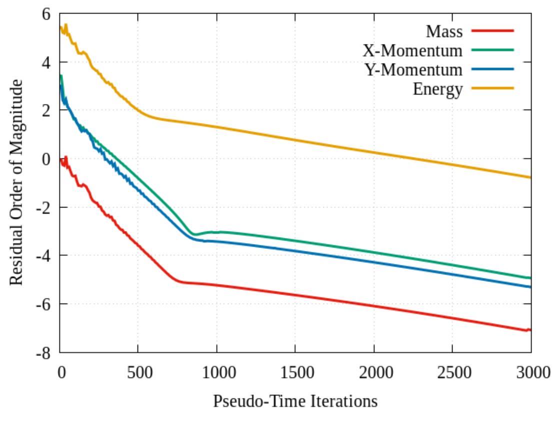
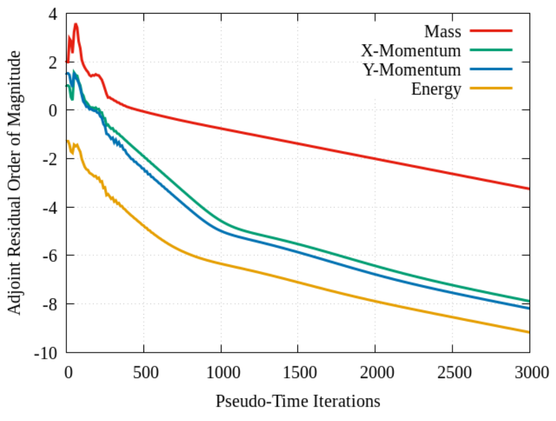


Figure 8.10: NACA0012 airfoil optimization, laminar flow, compressible fluid: Bézier control polygons (red) parameterizing separately the two sides of the baseline airfoil (black). Green vectors represent the computed sensitivity derivative at each control point of the initial airfoil.



(a)



(b)

Figure 8.11: NACA0012 airfoil optimization, laminar flow, compressible fluid: Convergence of the residuals of the (a) flow and (b) adjoint equations.

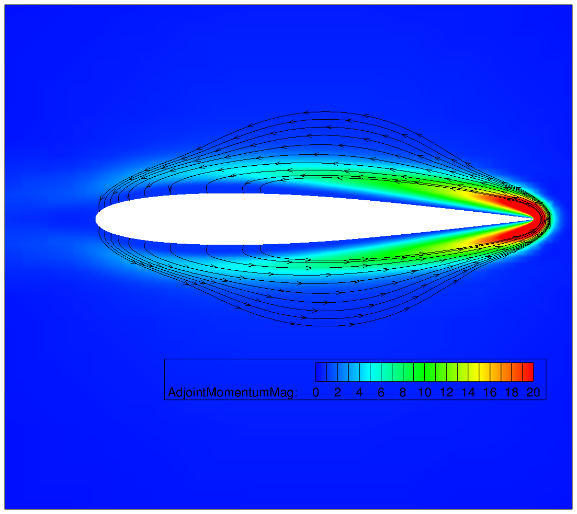


Figure 8.12: NACA0012 airfoil optimization, laminar flow, compressible fluid: Adjoint iso-velocity contours and adjoint streamlines.

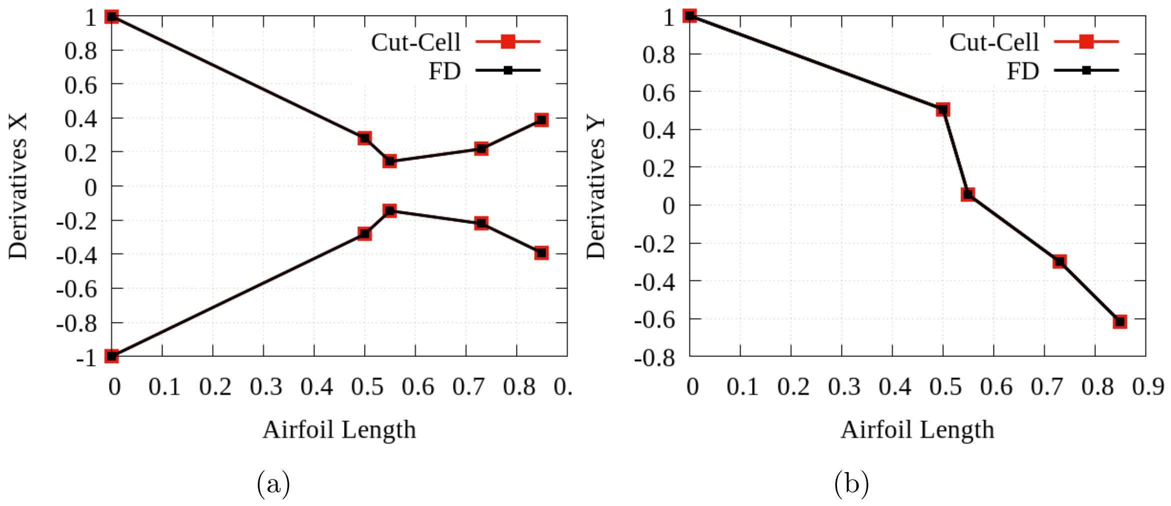


Figure 8.13: NACA0012 airfoil optimization, laminar flow, compressible fluid: Sensitivity derivatives computed w.r.t. the x and y coordinates of the control points. Comparison between adjoint derivatives (red) and FDs (black).

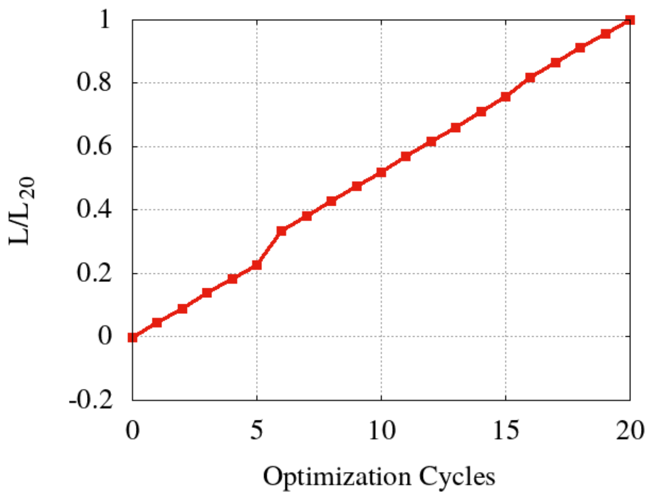


Figure 8.14: NACA0012 airfoil optimization, laminar flow, compressible fluid: Lift evolution during the adjoint-based optimization. To increase readability of the plot, lift is non-dimensionalized by its value at the 20th cycle.

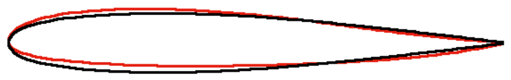


Figure 8.15: NACA0012 airfoil optimization, laminar flow, compressible fluid: Comparison between the baseline (black) and optimized (red) airfoil contours.

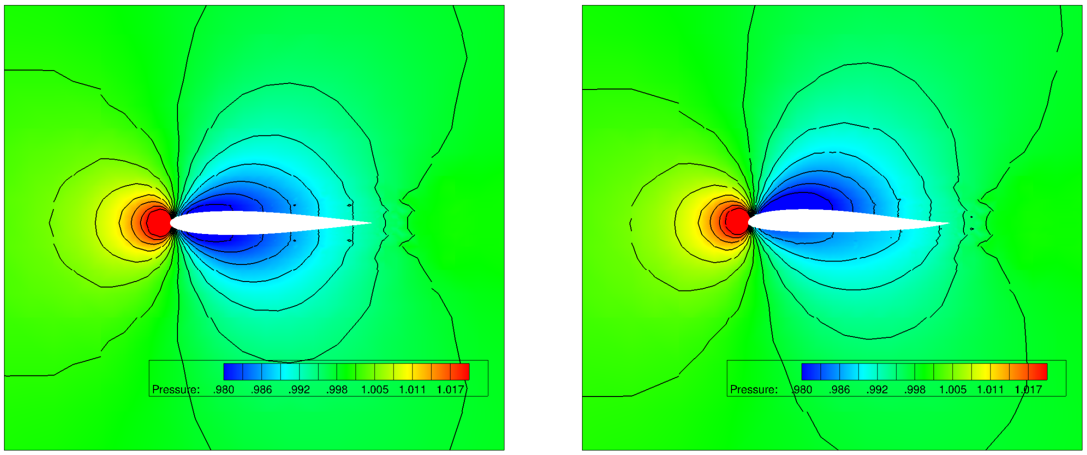


Figure 8.16: NACA0012 airfoil optimization, laminar flow, compressible fluid: Isobar areas of the (a) baseline and (b) optimized airfoil.

8.3 Unsteady Adjoint Solver Assessment

This application concerns the optimization of a pitching NACA0012 airfoil exposed in a subsonic compressible inviscid flow. The objective function to be minimized is the time-averaged lift during one period. The section's main scope confirms the correct differentiation of the algorithm which handles the appearing and disappearing cells due to the airfoil's motion. Moreover, the way the adjoint field is transferred to the mesh of the previous time step, as integration proceeds backwards in time is also verified. The airfoil surface is morphed using two Bézier curves parameterizing the pressure and suction side, as mentioned in section 8.1 and shown in fig. 8.17. The pitching motion around the quarter-chord is prescribed by the sinusoidal function $\alpha(t) = \alpha_\infty + \alpha_0 sin(\omega t)$, where $\alpha(t)$ is the angle between the airfoil's chord and the x-axis. The far-field flow angle and amplitude are $\alpha_\infty = 0.16°$ and $\alpha_0 = 2.51°$, respectively. The period is $T = 0.15s$ and the free-stream Mach number is 0.439. The flow equations are solved for four periods to overcome transient phenomena occurring at the beginning of the simulation. Upon the end of the third period, the integration required to compute the objective function begins and lasts for the entire last period.

The time-averaged objective function's differentiation leads to a reverse time integration method with an initial condition at the end of the unsteady phenomenon.

Thus, the flow problem's solution and mesh are stored at all time steps to be available for the adjoint solver. Data regarding the flow transition from one mesh to the next are also stored. According to the mathematical formulation presented in section 6.4, the adjoint problem must be solved for the entire time window of the four periods, which does not coincide with the time over which the objective function is integrated. Consequently, as the adjoint is solved backwards in time, the adjoint equations' source term which triggers the adjoint field takes on non-zero values only during the fourth period, i.e. the first period for the adjoint solver, and is set to zero afterwards, resulting in a gradual attenuation of the adjoint field.

The adjoint field's and corresponding mesh's behavior bear similarities with the unsteady primal case. Mesh is adapted close to the airfoil's boundary, and cells are transported from the fluid to the solid mesh region or vice-versa at each time step due to the airfoil's motion. The characteristic hysteresis is also evident in figs. 8.18a and 8.18c, where the airfoil passes through its equilibrium position on the upstroke and downstroke, respectively. Comparison between figs. 8.18 and 8.22, which presents the pressure field at the same time steps, shows the airfoil reversed motion during the adjoint simulation.

The objective function's derivatives w.r.t. the design variables computed by the developed adjoint software are compared to FDs in fig. 8.19 showing excellent agreement and yielding to a relative error smaller than $10^{-4}\%$. The FD step size is set to 10^{-6}, after conducting an independence study. Sensitivity derivatives are depicted in fig. 8.17. A shape design based on the CG method is carried out leading to an optimized airfoil contour, validating the software's shape optimization capabilities in unsteady flows. The smooth and monotonous objective function history, shown in fig. 8.20, indicates the correct differentiation of the temporal term, including the adjoint field's projection to the next mesh at each time step and the time-dependent geometrical terms treatment in the sensitivity derivatives' expression. As shown in fig. 8.21, the optimization algorithm cambers the geometry close to the trailing edge affecting the pressure exerted on both airfoil's sides. Additionally, the adjoint field is quite intense close to the trailing edge, as this is the area which is mostly affecting lift. Comparison of figs. 8.22 and 8.23 illustrates that pressure is lower close to the suction side, leading to higher lift.

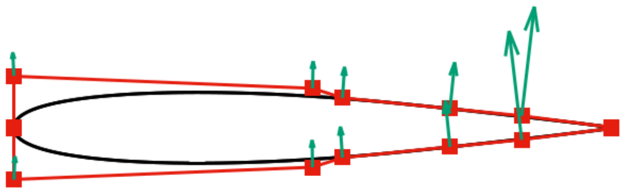


Figure 8.17: Pitching NACA0012 airfoil optimization, inviscid flow, compressible fluid: Bézier control polygons (red) parameterizing separately the two sides of the baseline airfoil (black). Green vectors represent the computed sensitivity derivative at each control point of the initial airfoil.

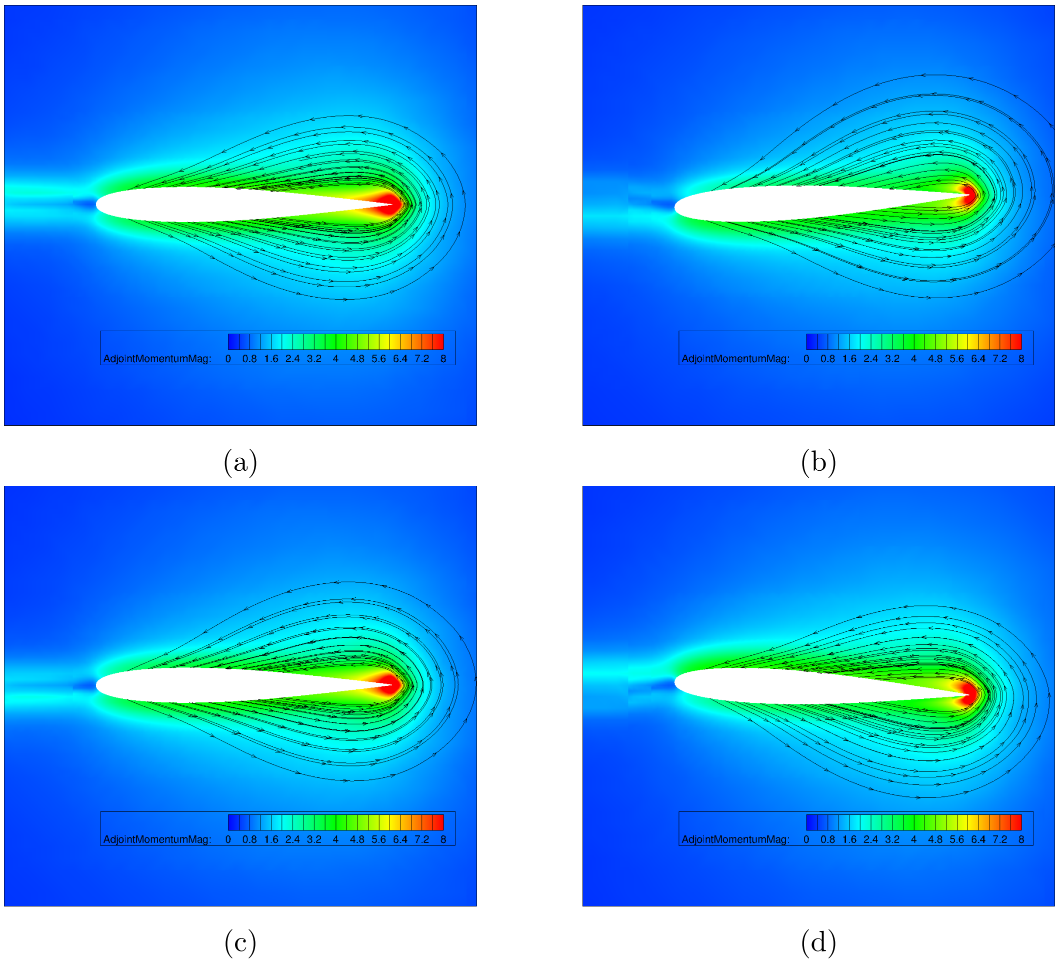
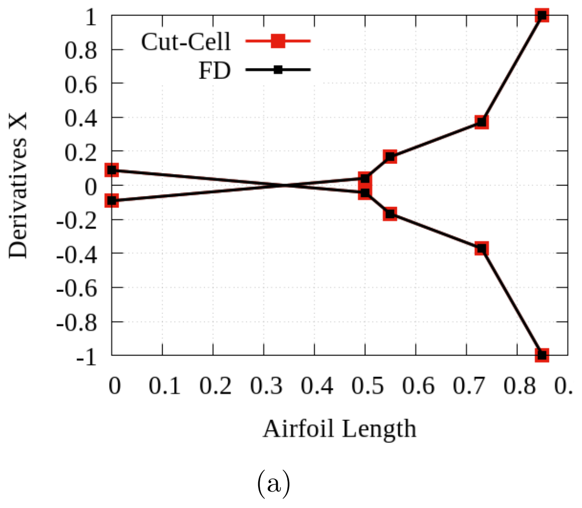


Figure 8.18: Pitching NACA0012 airfoil optimization, inviscid flow, compressible fluid: Adjoint iso-velocity contours and adjoint streamlines at T, 5T/4, 7T/4 and 2T time instants.



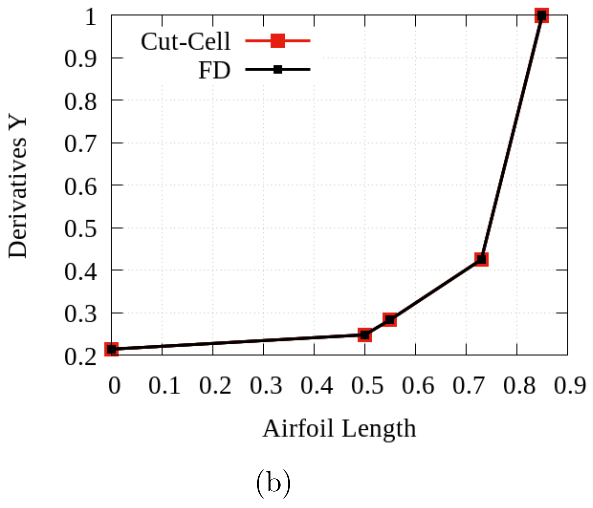


Figure 8.19: Pitching NACA0012 airfoil optimization, inviscid flow, compressible fluid: Sensitivity derivatives computed w.r.t. x and y coordinates of the control points. Comparison between the adjoint derivatives (red) and FDs (black).

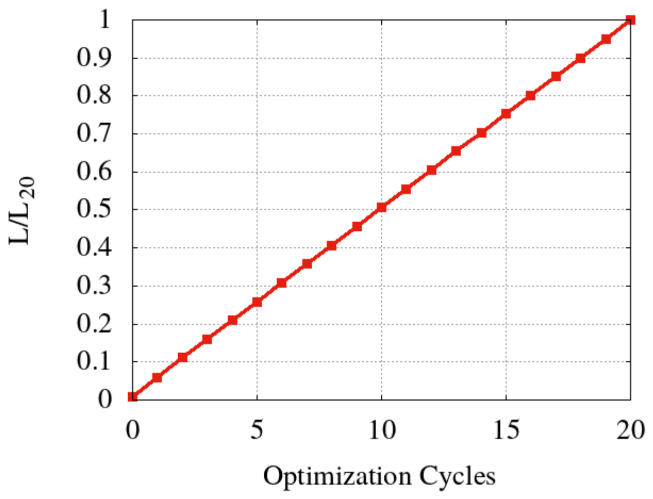


Figure 8.20: Pitching NACA0012 airfoil optimization, inviscid flow, compressible fluid: Time-averaged lift evolution during the adjoint-based optimization.

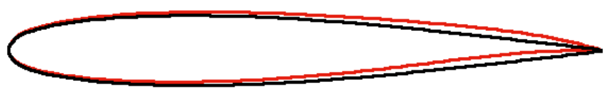
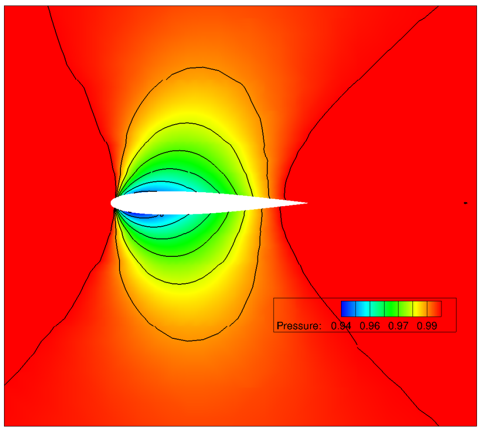
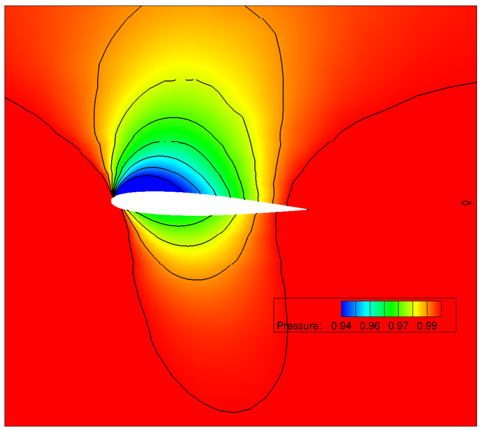


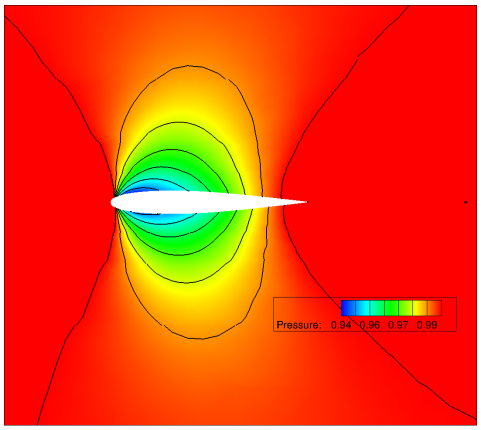
Figure 8.21: Pitching NACA0012 airfoil optimization, inviscid flow, compressible fluid: Comparison between the baseline (black) and optimized (red) airfoil contours.



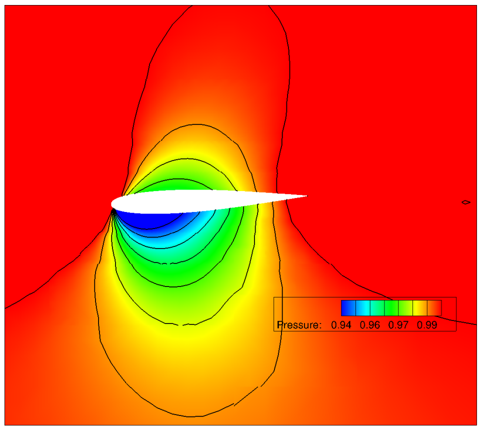
(a)



(b)



(c)



(d)

Figure 8.22: Pitching NACA0012 airfoil optimization, inviscid flow, compressible fluid: Iso-bar areas at T, 5T/4, 7T/4 and 2T time instants of the baseline airfoil.

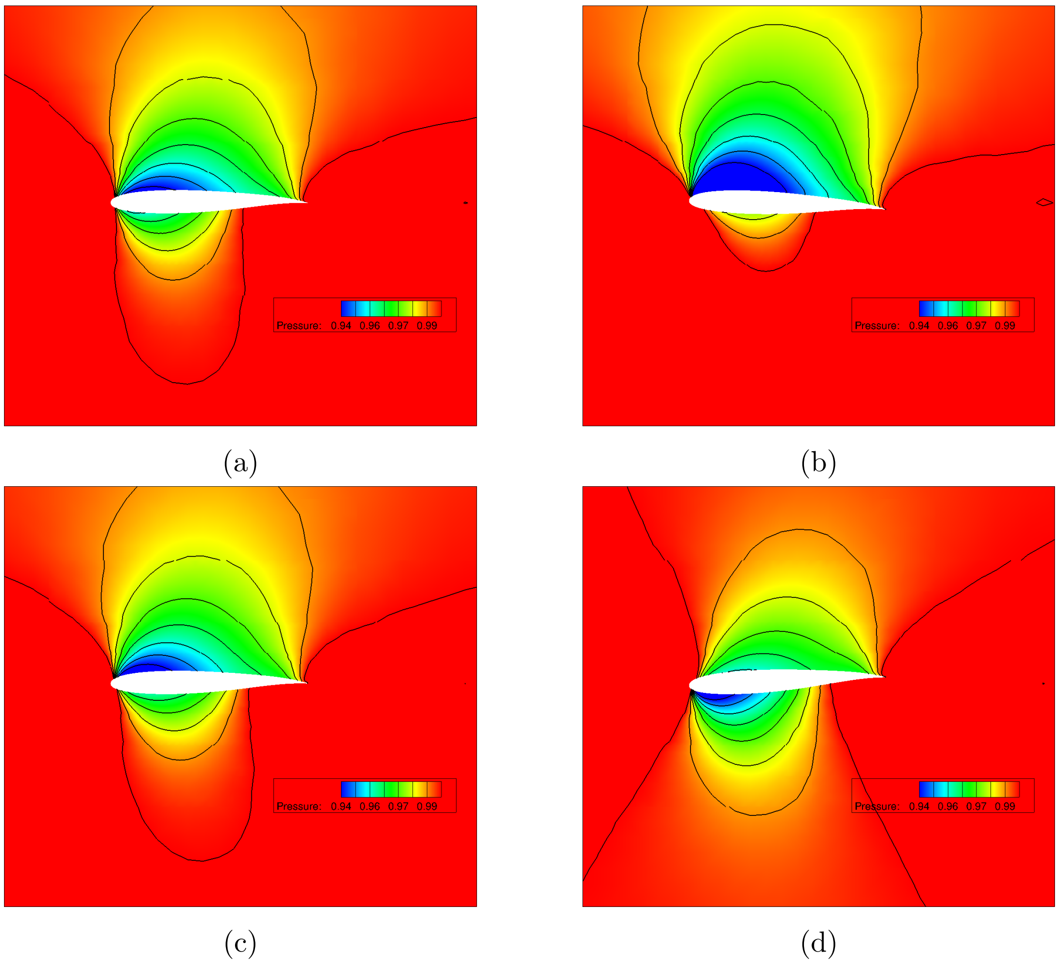


Figure 8.23: Pitching NACA0012 airfoil optimization, inviscid flow, compressible fluid: Iso-bar areas at T, 5T/4, 7T/4 and 2T time instants of the optimized airfoil.

Chapter 9

Optimization of Industrial Applications

This chapter provides the implementation of the gradient-based optimization assisted by the adjoint method in real-world applications. The corresponding adjoint theory is developed in chapter 7. In contrast to the applications presented in chapter 8, no shape parameterization is used. Instead, the design variables are the coordinates of the surface nodes of the optimized shapes. As the number of design variables is high, the adjoint method should be used since the cost of computing the objective function's gradient is independent of the number of design variables. In practice though, the sensitivity field contains high-frequency content, making its use in the optimization very ineffective. An additional processing step is thus needed at each cycle, which guarantees the smoothness of the shape while avoiding the formation of invalid surface elements. The used approach, described in section 7.6, combines an algorithm imitating an elliptic surface equation behavior with filters preventing gradient spikes generation on the surface. Each optimization step requires much more computational time than the benchmark cases studied in chapter 8, which restricts the optimization cycles to 20 for all applications due to computational resource limitations. It is essential to clarify that the premature termination is not related to the software reliability. A study presented in section 8.1 indicates the software's ability to continue the optimization until a predefined criterion is met. The presented optimization cases refer to compressible or incompressible fluid flows with stationary or moving geometries. They are concerned with the total pressure losses minimization in an S-duct, the lift maximization of a wing, the outlet tangential velocity minimization in a submersible pump, and the back-flow minimization along with the volume flow rate maximization (two targets) in a diaphragm micropump. Moreover, an optimization under uncertainties is performed. The cases mentioned before are solved by the cut-cell variant of the adjoint method. However, the adjoint ghost-cell method is also capable of optimizing industrial cases. Such an example is presented in the last section, where a compressor rotor is optimized supported by the SVD method. Finally, work done by the author on the continuous adjoint to the cut-cell method implemented in unsteady problems can be found in [273].

9.1 S-Duct Optimization

In this application, the shape of a 3D S-duct is optimized for minimum total pressure losses. The duct is formed by an S-shaped central part, upwind and downwing extended with straight segments. The baseline geometry's surface and volume mesh consist of around 20K triangles and 100K cells, respectively. The imposed total (1.0022 bar) and static (1 bar) pressure conditions at the duct's inlet and outlet drive the incompressible fluid of density 1 kg/m^3. The Reynolds number based on the duct's inlet diameter and the isentropic velocity which corresponds to the aforementioned total and static pressures is 250. Since the inlet total pressure remains constant during the optimization, only the outlet total pressure contributes to the objective function. The design variables are the coordinates of the duct surface nodes. Nodes placed on the straight segments cannot vary during the optimization. The 20-cycle optimization takes 8 hours on 48 processors. Losses have been decreased by more than 2.5%, as displayed by the convergence history plotted in fig. 9.1, where the vertical axis presents the difference between the losses at each optimization cycle and the baseline geometry. A steep reduction of the objective function is taken place in the first 5 cycles followed by smaller changes while the optimization algorithm reaches the minimum. Initial and optimized geometries are shown in fig. 9.2, and are compared in fig. 9.3. The surface displacement becomes more clear by the comparison of three cross-sections' geometries, fig. 9.4. Displacement is not totally symmetrical with respect to the x-axis due to the non-symmetrical surface triangulation. Iso-velocity contours are presented in fig. 9.5 at the same cross-sections for the baseline and optimized geometry. It seems that the wall deformation increases the low-velocity areas close to the boundary of the optimized geometry causing stress reduction between the solid and the fluid. The objective function reduction is evident in figs. 9.6 and 9.7 where total pressure loss contours are shown along the longitudinal direction and at the duct's exit, respectively. Losses have been mostly decreased in the duct's central axis vicinity, while regions close to the wall remain the losses' primary source. Fig. 9.8 presents the sensitivity map in the duct's initial shape. Red regions indicate areas where the surface should be displaced inwards, blue should do the opposite and green parts should remain still. The optimized surface is smooth, without being affected by high frequency changes in the computed sensitivity derivatives. Its strange shape could have been avoided by using less design variables, for instance by means of a free-form parameterization method [311].

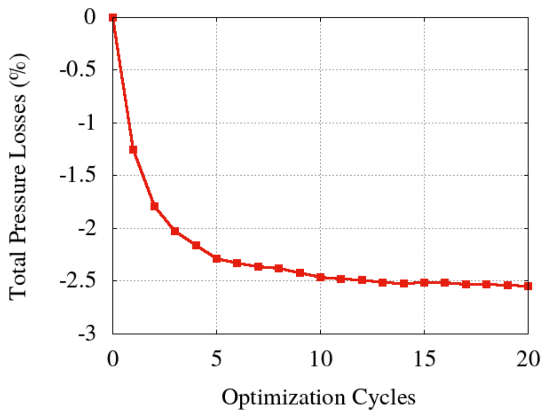


Figure 9.1: S-duct optimization: Reduction in total pressure losses during the optimization loop.

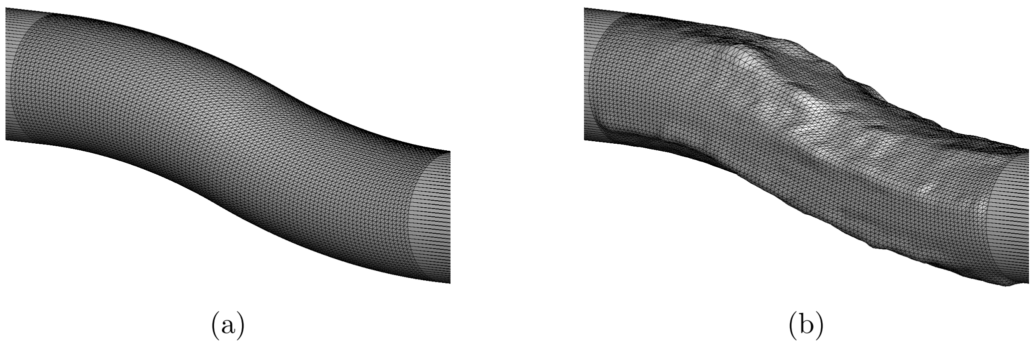


Figure 9.2: S-duct optimization: Close-up views of (a) baseline and (b) optimized geometries, in the middle of the duct.

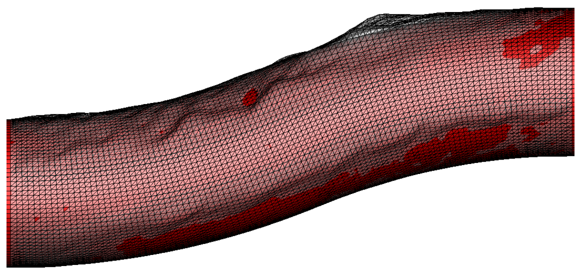


Figure 9.3: S-duct optimization: Baseline (gray-transmissive) and optimized (red) ducts plotted together.

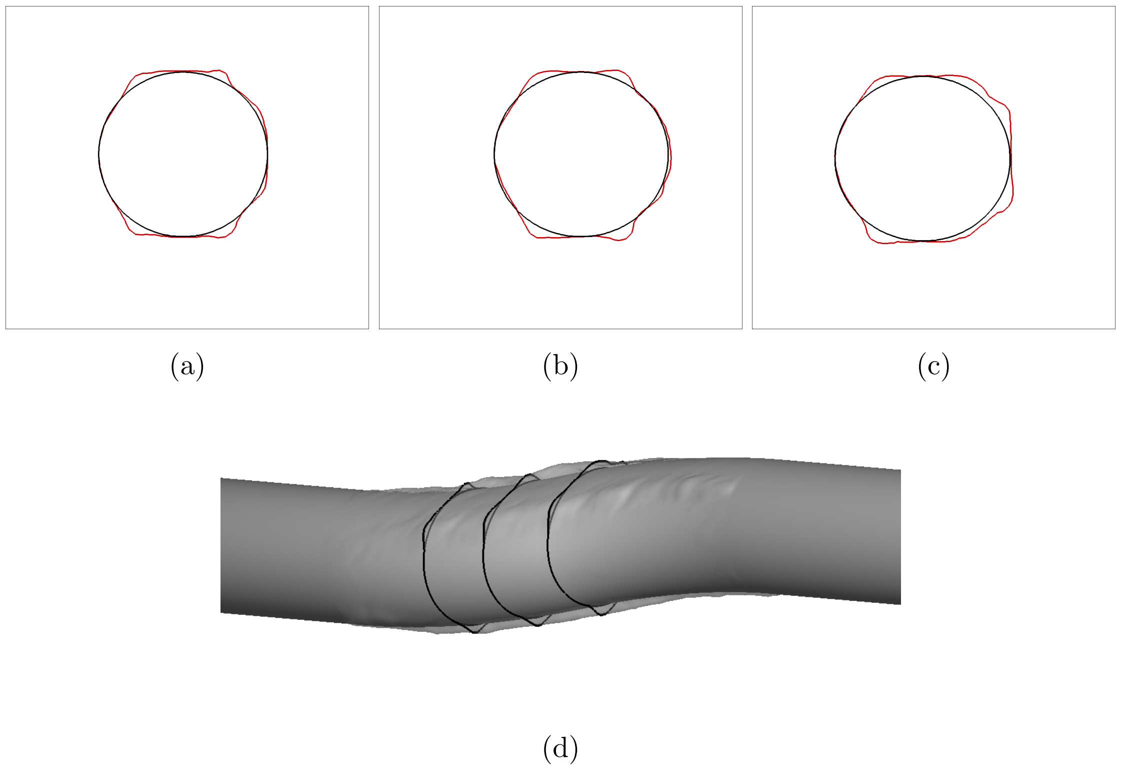


Figure 9.4: S-duct optimization: Comparison of the (a) baseline (black) and (b) optimized geometry (red) at three cross-sections. Their position is shown in (d) which presents the baseline (dark gray) and optimized ducts (light gray).

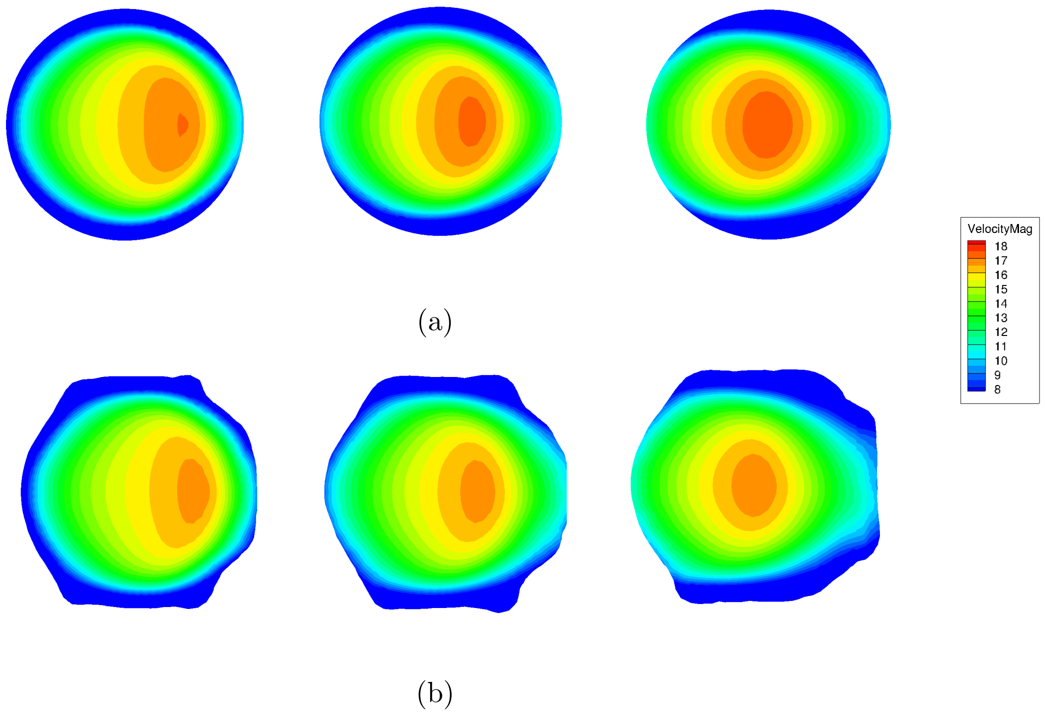


Figure 9.5: S-duct optimization: Velocity magnitude iso-areas at the cross-sections defined in fig. 9.4d for the (a) baseline and (b) optimized geometries.

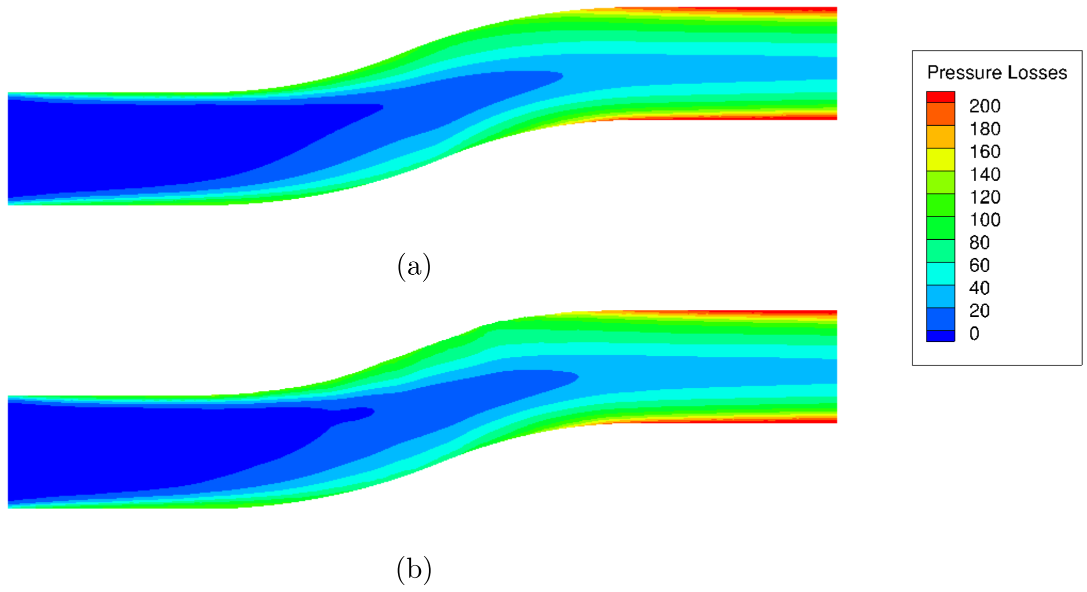


Figure 9.6: S-duct optimization: Total pressure loss contours over a cross-section with a plane along the longitudinal direction in the (a) baseline and (b) optimized geometry.

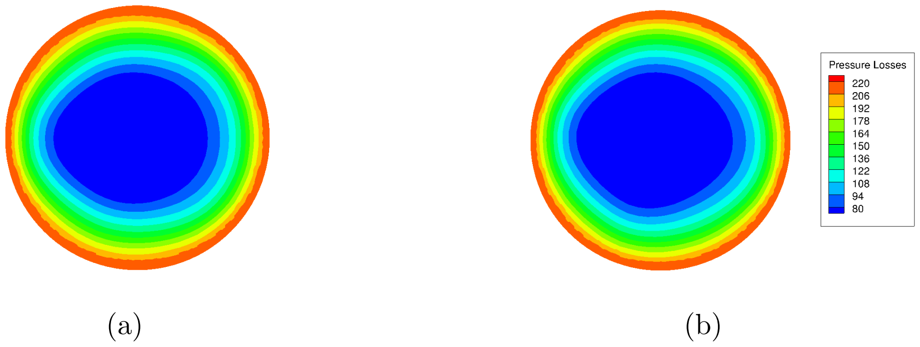


Figure 9.7: S-duct optimization: Total pressure loss contours at the exit of the (a) baseline and (b) optimized duct.

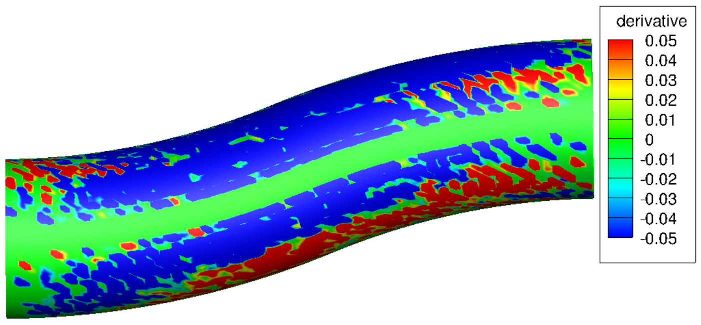


Figure 9.8: S-duct optimization: Sensitivity map on the duct surface computed at the end of the first optimization cycle.

9.2 Wing Optimization

This application is concerned with the optimization of an isolated wing in a subsonic compressible flow for lift maximization. The flow is assumed to be inviscid. The baseline wing is build based on the symmetrical ONERA-D airfoil. The leading and trailing edge sweep angles are 30° and 15.8°, respectively, forming a relevant surface area of around 2.19 m^2. All but the leading and trailing edge surface nodes are allowed to move during the optimization. Nodes placed at the tip are restricted to move only along the vertical direction. The far-field Mach number, angle-of-attack and angle-of-sideslip are 0.5, 0° and 0°, respectively. At these flow conditions, the initial design produces zero lift. The mesh for the baseline geometry consists of 100K cells. After 20 optimization cycles, lift has increased to almost 3 kN as shown in fig. 9.9 leading to a lift coefficient equal to 0.343. The wing geometries before and after the optimization are compared in fig. 9.10 at three cross-sections perpendicular to the lateral direction, fig. 9.10d, . The wing's part close to the leading edge is mainly displaced. Its curvature increases as the cross-section reaches the wing's tip. The new design supports the pressure increase on the lower side while decreasing pressure on the upper side. Fig. 9.11 compares the pressure fields for the baseline and optimized wings. Lift rise is also evident from the surface pressure distributions on the upper, fig. 9.12, and lower sides, fig. 9.13. Finally, the sensitivity map computed at the end of the first optimization cycle is presented in fig. 9.14. The map has been smoothed enough to suppress high frequencies, allowing for a reasonable shape deformation. As expected, derivatives are higher close to the trailing edge. The red color denotes areas that should be pulled outwards. The opposite happens for areas in blue, giving rise to the characteristic cambered shape close to the trailing edge shown in fig. 9.10.

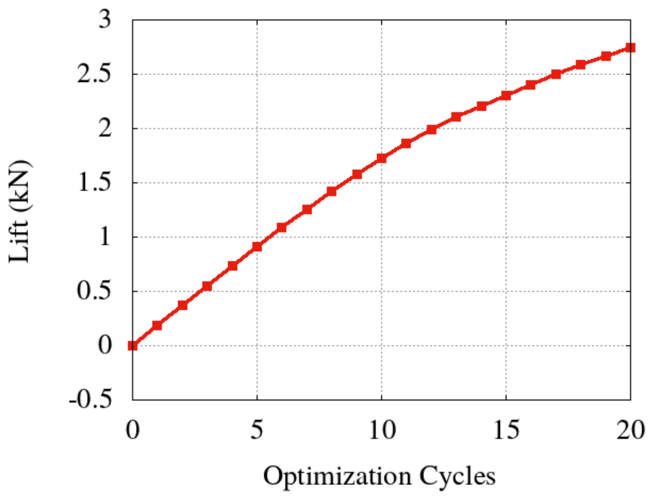
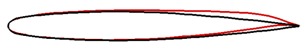
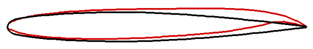


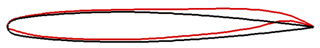
Figure 9.9: Wing optimization: Increase in the objective function during the optimization loop.



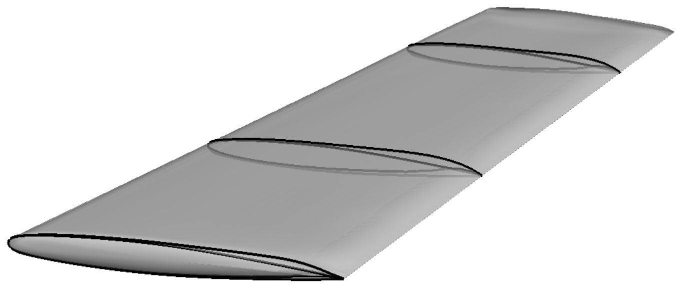
(a)



(b)

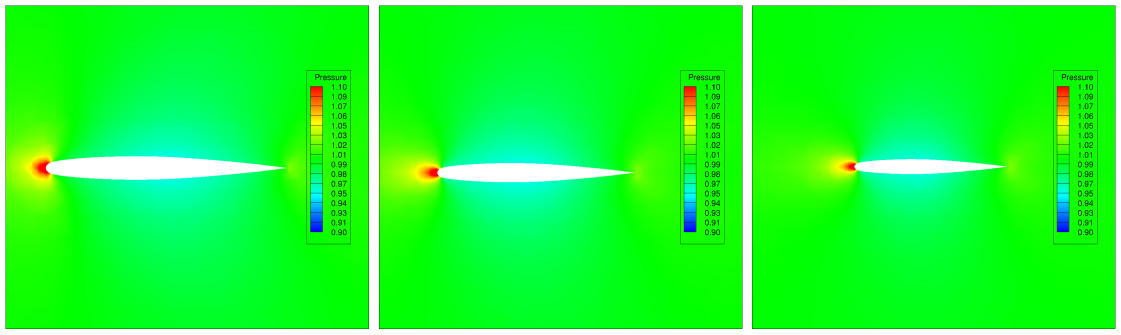


(c)

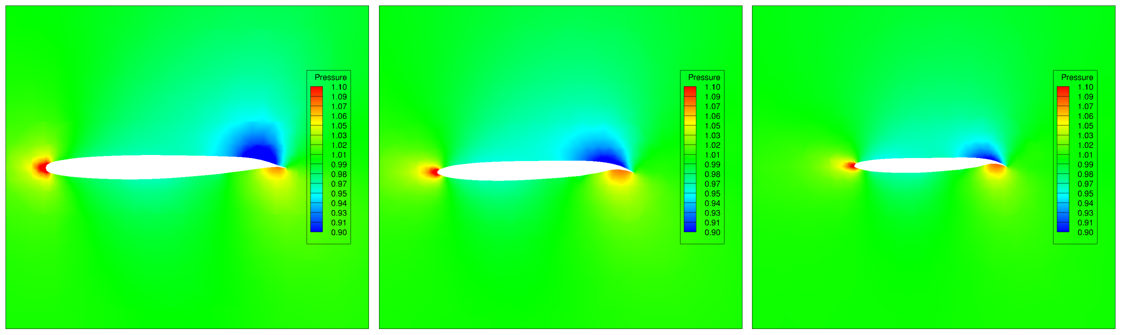


(d)

Figure 9.10: Wing optimization: Comparison between the baseline (black) and optimized (red) wing airfoils. From (a) to (c), wing tip is approached. Cross-sections' position is shown in (d).

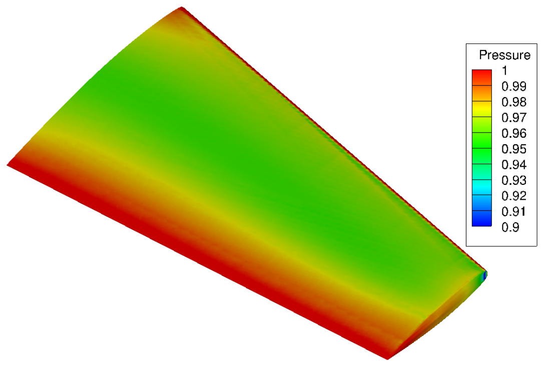


(a)

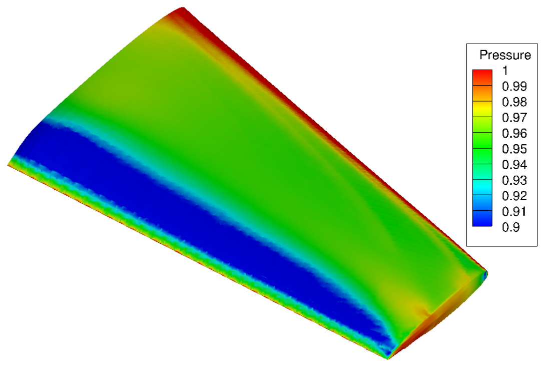


(b)

Figure 9.11: Wing optimization: Iso-bar contours in cross-sections defined in fig. 9.10d for the (a) baseline and (b) optimized wings.

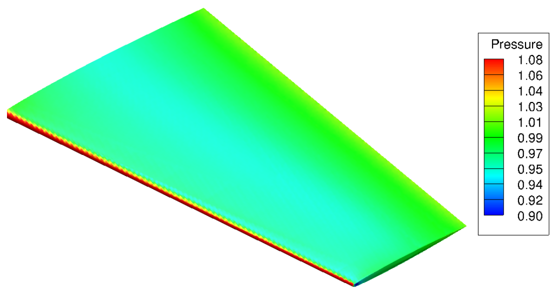


(a)

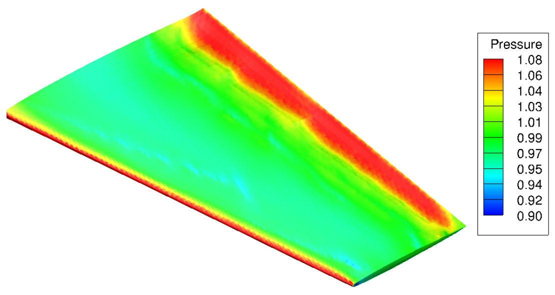


(b)

Figure 9.12: Wing optimization: Pressure distribution on the suction side for the (a) baseline and (b) optimized wings.

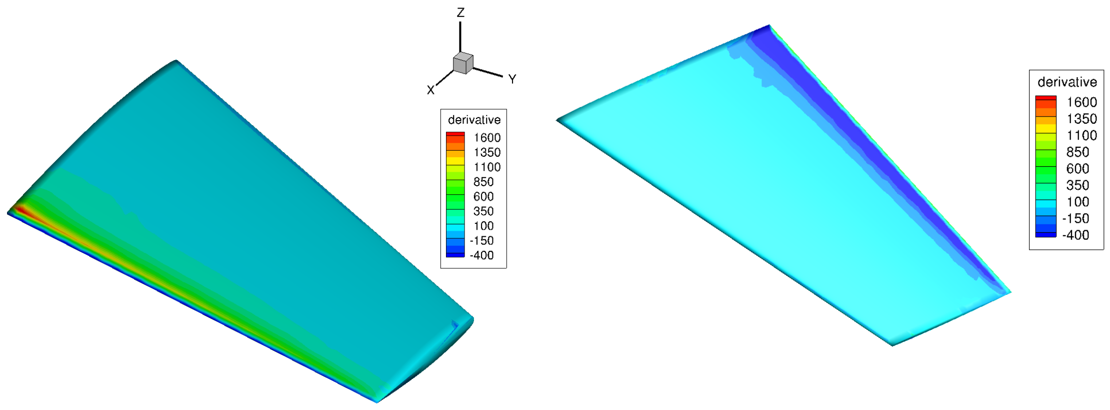


(a)



(b)

Figure 9.13: Wing optimization: Pressure distribution on the pressure side for the (a) baseline and (b) optimized wings.



(a) (b)

Figure 9.14: Wing optimization: Sensitivity map on the (a) suction and (b) pressure side of the baseline wing.

9.3 Submersible Pump Optimization

This application aims at the optimization of a 3D Electrical Submersible Pump (ESP). Publications dedicated to the ESP gradient-based optimization assisted by the adjoint method are limited, although it allows for a low-cost computation while maintaining a high degree of freedom by handling a great number of design variables. Information about the ESP technology and the flow analysis of the baseline geometry can be found in section 5.4. The study is part of a research program funded by Schlumberger Cambridge Research Limited.

The target of the present optimization problem is the minimization of the radial (v_r) and peripheral (v_p) outlet velocity components, which is mathematically expressed as

$$J = \int_{S_{out}} \left(v_r^2 + v_p^2\right) ds$$

where S_{out} is the stage outlet surface. Both impeller and diffuser blades are deformed during the optimization while the casing remains stationary. Nodes located at the hub and shroud are allowed to move only along the peripheral direction. At each optimization cycle, the casing nodes positions are smoothed ensuring the validity of all surface elements. An inviscid flow model for compressible fluids is used.

Fig. 9.15 shows the evolution of objective function's deviation from its value at the end of the first cycle. According to the plot, the outlet tangential velocity has decreased by 35%. Tangential velocity magnitude contours at the exit of the initial and final geometry are compared in fig. 9.16 for four snapshots equally distributed along a single period. Two cross-sections perpendicular to the pump's axis show the impeller and diffuser blades displacement, fig. 9.17. While slight changes in the impeller can be seen, the diffuser has noticeably been changed, meaning that its blades, being closer to the exit, mostly determine the outlet velocity direction. The impeller and diffuser blades deformation is presented in fig. 9.18. Finally, the sensitivity map on the impeller and diffuser baseline blades is shown in fig. 9.19. Areas colored in red should be moved inwards whereas blue areas should be moved outwards. Nodes at which the sensitivity derivative is close to zero are colored in green. Areas with high absolute valued sensitivity derivatives have a more significant potential for optimization. According to the presented sensitivity maps, higher displacements are detected close to their trailing edge which is the area that mainly affects the velocity direction.

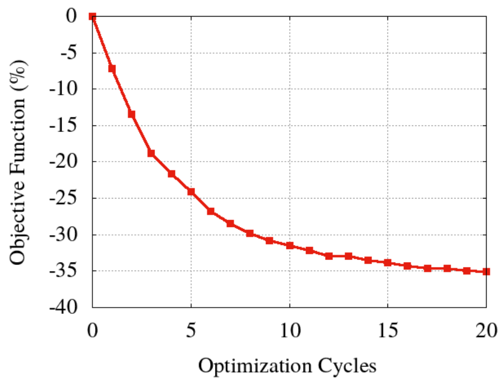


Figure 9.15: Submersible pump stage optimization: Optimization history. A reduction of $\sim$ 35% in the objective function is achieved after 20 optimization cycles.

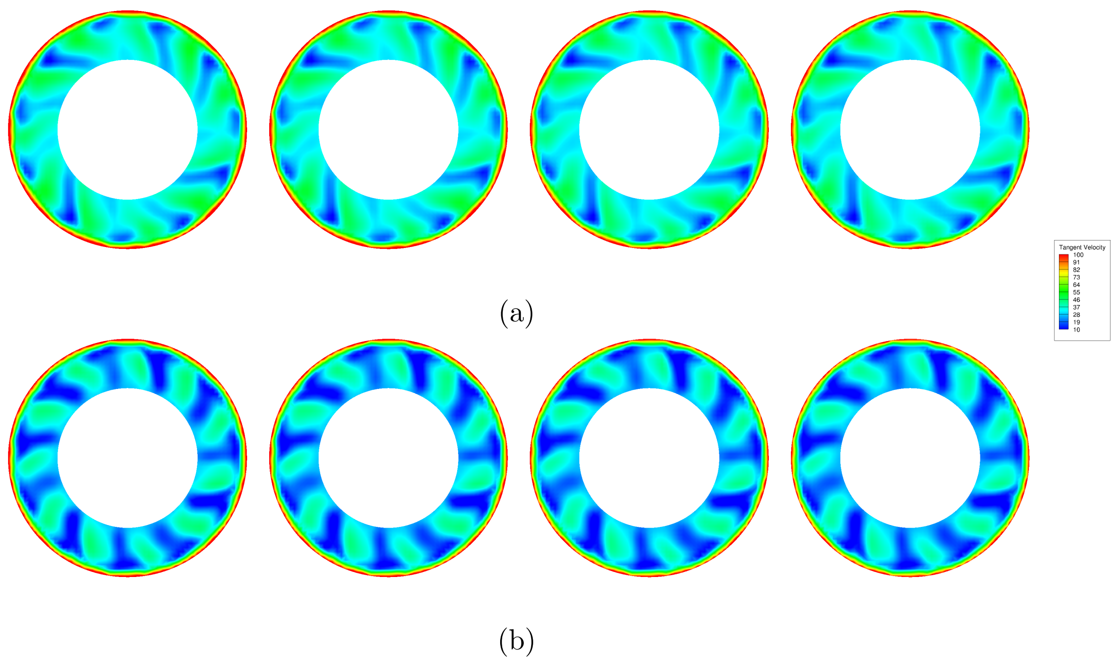


Figure 9.16: Submersible pump stage optimization: Tangential velocity magnitude at the pump outlet at four equally distributed time steps along one period for the (a) baseline and (b) optimized geometry.

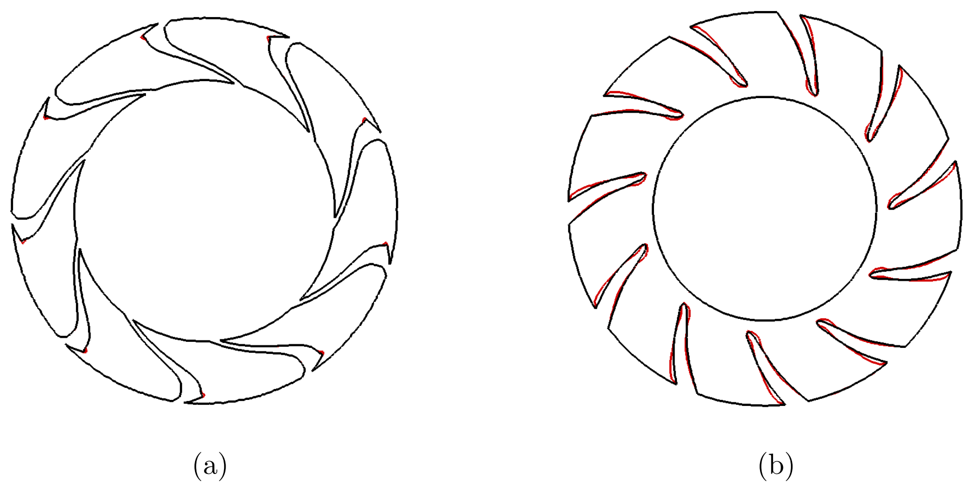
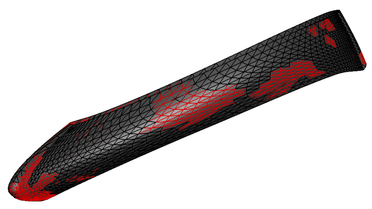
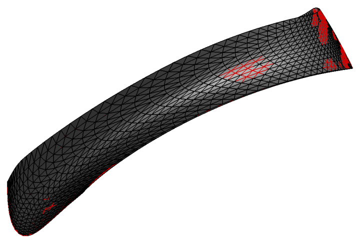


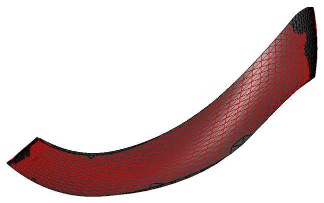
Figure 9.17: Submersible pump stage optimization: Comparison between the baseline (black) and optimized (red) blades presented in cross-sections perpendicular to the pump's axis for (a) the impeller and (b) the diffuser. A close-up view of (a) is shown in (c).



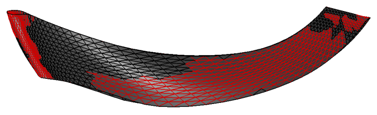
(a)



(b)

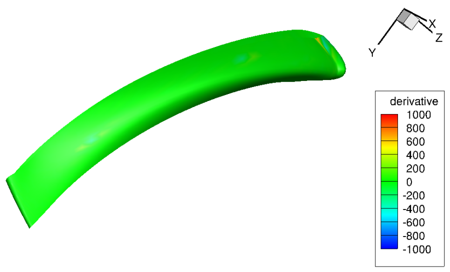


(c)

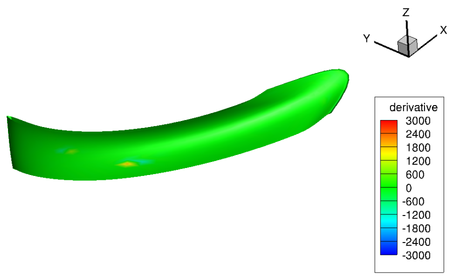


(d)

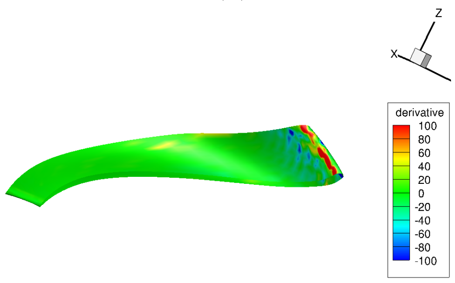
Figure 9.18: Submersible pump stage optimization: Comparison between the baseline (black) and optimized (red) impeller blade for the (a) pressure and (b) suction side and diffuser blade for the (c) pressure and (d) suction side.



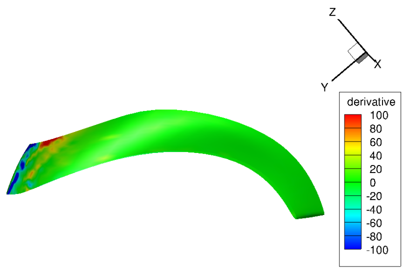
(a)



(b)



(c)



(d)

Figure 9.19: Submersible pump stage optimization: Sensitivity map for the impeller blade on the (a) pressure and (b) suction side and diffuser blade on the (c) pressure and (d) suction side.

9.4 Valveless Diaphragm Pump Optimization under Uncertainties

This section focuses on optimizing a 3D valveless diaphragm micropump in the presence of uncertainties. The diaphragm's motion is parameterized by eight variables being the design and uncertain optimization variables. A detailed description of the parameterization scheme is presented in section 5.3. Its flow analysis is demonstrated in the same section, showing that the absence of valves allows the flow to re-enter from the pump's exits at a significant time window within a period. Backflow is unwanted in most practical cases, such as medical applications, and should be minimized. It is defined as the integral of the negative velocity at the exit.

$$Q_{bf} = \frac{1}{T} \int_T \int_{S_{out}} min(0, \vec{v}.\vec{n}) ds dt$$

where S_{out} is the pump's outlet surface, $\vec{n}$ is its unit normal vector and $\vec{v}$ the flow velocity. Another drawback is the low flow rate micropumps can handle due to their small size [81]. Its mathematical expression is

$$Q_{net} = \frac{1}{T} \int_T \int_{S_{out}} \vec{v}.\vec{n} ds dt$$

These two functions, related to performance of the pump, are the so-called quantities of interest.

Operating and manufacturing inaccuracies, modeled by introducing uncertainties into the design variables, affect the quantities of interest. The two objective functions arise by computing the mean values and standard deviations of the two quantities of interest and forming their weighted sum,

$$\begin{aligned} F_1 =& w_{11}\mu_{Q_{bf}} + w_{12}\sigma_{Q_{bf}} \\ F_2 =& w_{21}\mu_{Q_{net}} + w_{22}\sigma_{Q_{net}} \end{aligned}$$

Here, $w_{11} = +1$, $w_{12} = +1$, $w_{21} = +1$ and $w_{22} = -1$. Their signs depend on whether minimization or maximization is targeted. The non-intrusive Polynomial Chaos Expansion (PCE) [85], [335] is used to compute the required statistical moments. It is assumed that all uncertain variables follow the normal distribution (w), meaning that Hermite Polynomials (He_i) should be selected. The stochastic quantities of interest are given by the following truncated summation,

$$Q_i(b) \simeq \sum_{j=1}^{q} \alpha_{ij} He_j(b)$$

where q is the user-defined chaos order. The PCE coefficients are given by

$$\alpha_{ij} = \int_D Q_i(b) He_j(b) w(b) db$$

where D is the design space. The above integrals are computed by the Gauss quadrature integration method, which determines a set of Gaussian nodes, each of them should be evaluated by the cut-cell CFD solver. The first two statistical moments are computed through Galerkin projections as

$$\mu_{Q_i} = \alpha_{i0}, \quad \sigma^2_{Q_i} = \sum_{j=1}^{q} a^2_{ij}$$

The chaos order is set to one and, thus, 9 expansion coefficients should be computed, requiring the significant number of 256 CFD-based evaluations per candidate solution. The overall computational cost is reduced by using the Smolyak sparse grid theory [287], according to which the same coefficients are approximated at the cost of 17 evaluations on the cut-ell software.

The in-house optimization tool EASY (Evolutional Algorithms SYstem) [1] computes the Pareto front of non-dominant solutions [151]. The optimization cost is reduced by implementing surrogate models or metamodels (Metamodel-Assisted EA or MAEA) [154] and the Principal Component Analysis (PCA) [169] of the population members. Metamodels replicate the objective functions computed by the CFD tool at an almost negligible computational cost. The PCA transforms the design space into a new feature space, in which the evolution operators perform much better. PCA also assists metamodels to be trained with the most significant input variables only, as identified by the PCA. The computational cost is further reduced by combining the PCA-driven MAEA with a Gradient-Based (GB) optimization method, giving rise to a hybrid algorithm, in which the EA undertakes the exploration of the design space and the GB method the refinement of selected promising solutions [285], [152].

The GB method implementation starts by concatenating the two objectives in a single scalar function equal to their weighted sum ($F = w_1F_1 + w_2F_2$). Then, the adjoint cut-cell solver computes its gradient and the selected individuals are improved by performing a single descent step. The computation of $\partial F_i/\partial b_k$ requires the statistical moments' derivatives given by

$$\frac{\partial \mu_{Q_i}}{\partial b_k} = \frac{\partial \alpha_{i0}}{\partial b_k}$$
$$\frac{\partial \sigma_{Q_i}}{\partial b_k} = \frac{1}{\sigma_{Q_i}} \sum_{j=1}^{q} \alpha_{ij} \frac{\partial \alpha_{ij}}{\partial b_k}$$

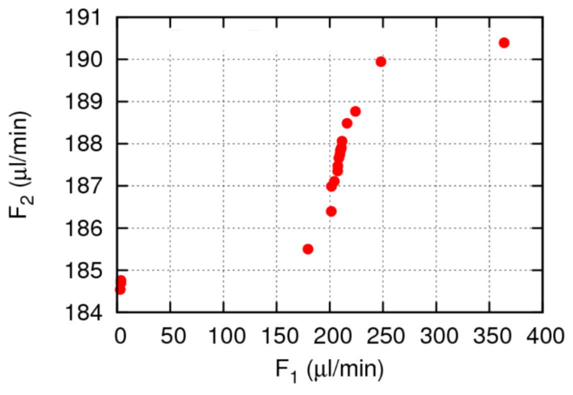
The PCE coefficients' derivatives are

$$\frac{\partial \alpha_{ij}}{\partial b_k} = \int_D \frac{\partial Q_i}{\partial b_k} He_j w db$$

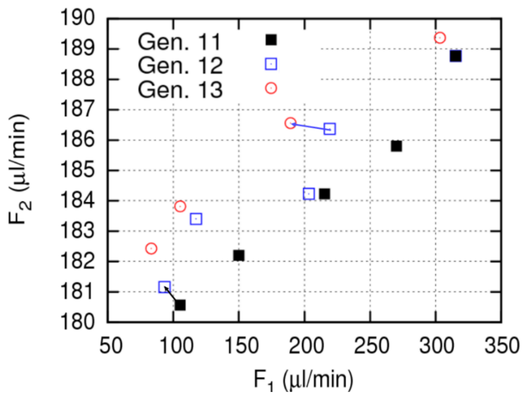
where $\partial Q_i/\partial b_k$ are computed by the cut-cell adjoint software. The GB improvement is applied to just one individual in each generation due to the gradient's extra computational cost. The new individual should simultaneously improve all objective functions, which is possible by correctly choosing the weights' values. Each pair of weights (w_1, w_2) determines a different direction in the objective functions' space. A descent direction leading to a new dominated individual is referred to as the Pareto Advancement Direction (PAD). A proper method for computing the PAD is given in [152]. As mentioned before, according to the non-intrusive PCE assisted by the Smolyak theory, each individual's evaluation costs 17 CFD runs and, if selected for GB refinement, another 34 adjoint runs are needed to compute the gradients.

A (6, 10) PCA-Assisted Hybrid Algorithm is used for the optimization problem. Both metamodels and PCA start being used after the first generation. The computational budget is restricted to 1400 calls to the cut-cell software or its adjoint due to their high computational cost. Fig. 9.20b presents the individuals' update by the GB method between three successive generations. The final front of non-dominated solutions is presented in fig. 9.20a. Moreover, fig. 9.21 shows the backflow and volume flow rate evolution within a period for the two front edges and the reference solution. Black lines indicate the mean values and the blue hatched areas correspond to the three-sigma interval width. In the maximum F_2 solution, the volume flow rate remains negative for almost half of the period, although its positive part overweights the negative one. The solution appears to be quite erratic, as it is seriously affected by the design variables' uncertainties. Regarding the minimum F_1 solution, backflow appears during a single step only. In the reference solution, the volume flow rate is positive only during half of the period, yielding a lower volume flow rate and higher backflow than both of the above-mentioned optimized solutions. From this point of view, any of the Pareto solutions dominate the reference pump. Finally, fig. 9.22 shows the instantaneous flow field of the two extreme points on the front and the reference solutions when the reference pump has the greatest backflow.

Research presented in this section has been funded by the Business Plan "Development of Human Resources, Education and Lifelong Learning" entitled "Support Researchers with Emphasis on Young Researchers" with the co-financing of Greece and the European Union. The project title is "*Design-Optimization of Diaphragm Pumps under Operational/Manufacturing Uncertainties using the cut-cell Method and Polynomial Chaos Expansion*".



(a)



(b)

Figure 9.20: Diaphragm pump optimization: (a) Computed Pareto front of non-dominated solutions. (b) Progress made in the front computed by the PCA-Assisted Hybrid Algorithm in three consecutive generations. Arrows show the PAD used by the GB method to upgrade a single individual per generation. Only part of the front is shown.

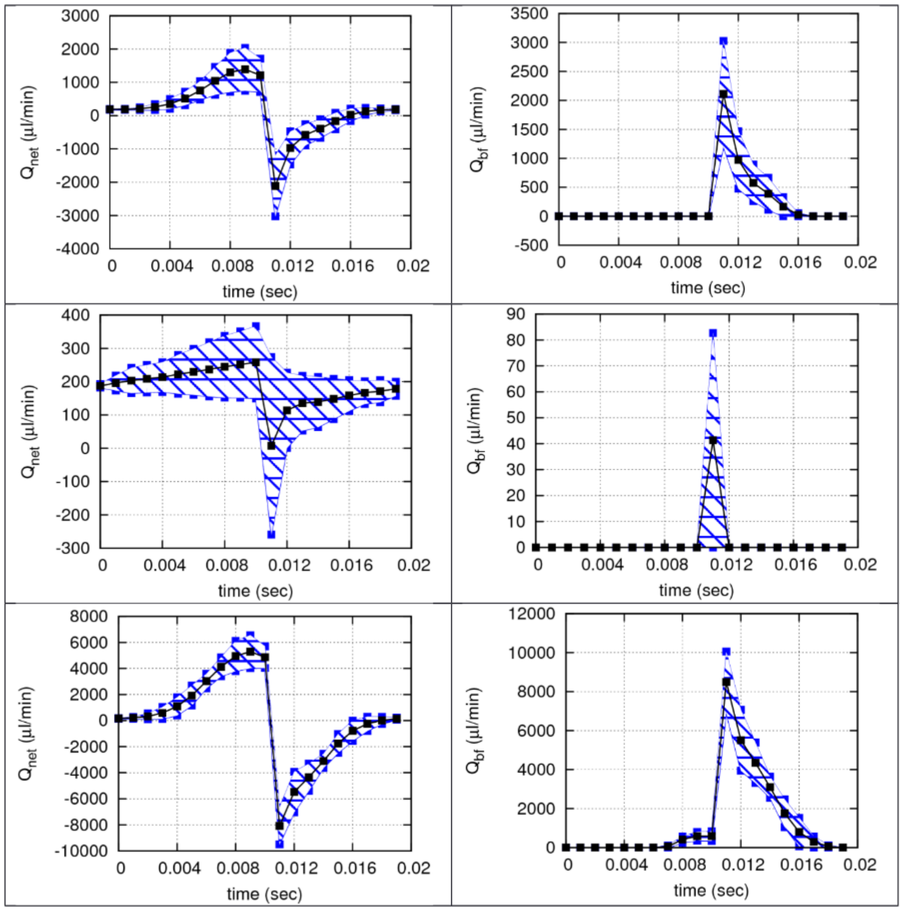


Figure 9.21: Diaphragm pump optimization: Q_{net} and Q_{bf} times series for the maximum F_2 (top), minimum F_1 (middle) and reference (bottom) solutions. Black line corresponds to the mean values. The hatched area signifies the $\pm 3\sigma$ zone.

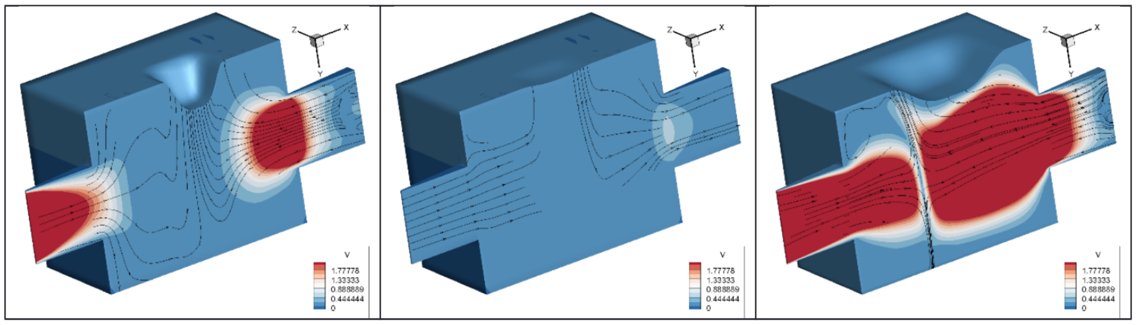


Figure 9.22: Diaphragm pump optimization: Instantaneous velocity fields of the maximum F_2 (left), minimum F_1 (middle) and reference solutions (right). Axes not in scale.

9.5 Optimization of a Compressor Rotor

In this subsection, the optimization of a compressor rotor is carried out, minimizing the swirl at its exit. The nodes coordinates constituting the rotor blades stand for the corresponding design variables. Its experimental investigation was carried out by Inoue in a low-speed rotating cascade facility [136]. Details of the geometry and the blading are given in [135].

Inlet boundary conditions are total pressure (1 bar) and total temperature (290 K). Inlet flow is aligned with the axial direction. At the outlet, the static pressure is 0.995 bar. The rotating speed is set to 680 rad/s. A fixed uniform mesh of $122 \times 122 \times 120$ cells is used for all time steps and the flow equations discretization is based on the ghost-cell method explained in section 3.8. Flow simulation terminated after periodicity is established. Fig. 9.23 shows the compressor's blades and iso-Mach contours at two cross-sections along the axial and radial directions, respectively. The pressure field has been projected on the blades surface and is presented in fig. 9.24. The resulting contours are quite noisy due to the wall boundary conditions indirect imposition and errors introduced during the projection process.

The continuous adjoint to the ghost-cell method is implemented to compute the sensitivity map on the blades' surface. Its backward in time integration along with the constant number of cells at each time step enables the iSVD method, presented in Appendix M, to compress the flow field time series. However, the combination of the iSVD or SVD with an explicit solver should be avoided due to its inability to represent the produced flow field accurately. The complication is detected on the choice of a very small time step for the flow simulation due to stability issues of the explicit ghost-cell solver. Let M be the matrix in which the field time series is stored. Although the matrix is never constructed as a whole, the iSVD aims to decompose it computing its singular values and vectors. However, due to the small time step choice, minor changes are expected between two successive flow fields inducing similarities among the columns of M. Therefore, the matrix is ill-conditioned, causing instability issues to the SVD or iSVD algorithm, significantly increasing the roundoff errors in the singular values and vectors computation.

The problem mentioned above is surpassed by introducing a slightly different procedure for flow field compression. According to that, the iSVD algorithm takes into account the instantaneous flow field every 100 time steps, avoiding the column similarities in M. Thus, the iSVD can provide the corresponding flow field to the adjoint solver only at every 100 steps. The rest necessary fields are approximated by linear interpolation. The method's computational cost is comparable with the cost of the unsteady flow simulation and is less than the cost of the check-pointing algorithm.

Six optimization cycles were carried out, reducing the objective function by 7%, fig. 9.25. The sensitivity map computed on the pressure and suction side of the blade is presented in fig. 9.26, where the blue color indicates the regions pushed inwards, while the red-colored regions are deformed along the opposite direction. The derivatives' magnitude is higher on the blade's parts closer to the exit, signifying their major importance for the objective's minimization. Finally, fig. 9.27 compares the initial and optimized blades.

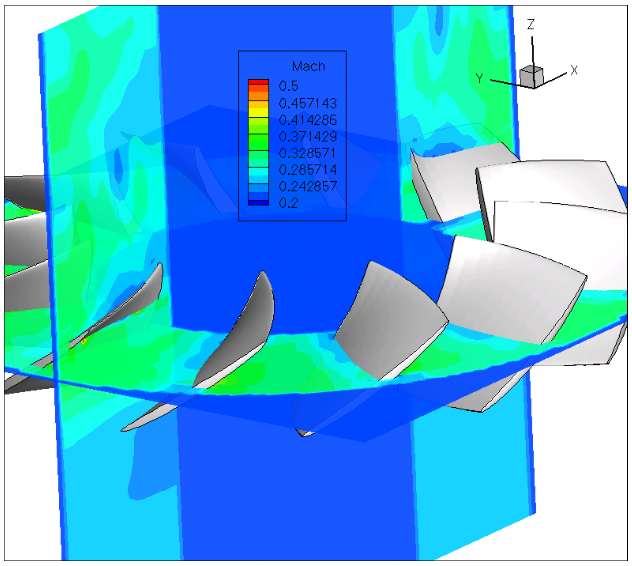
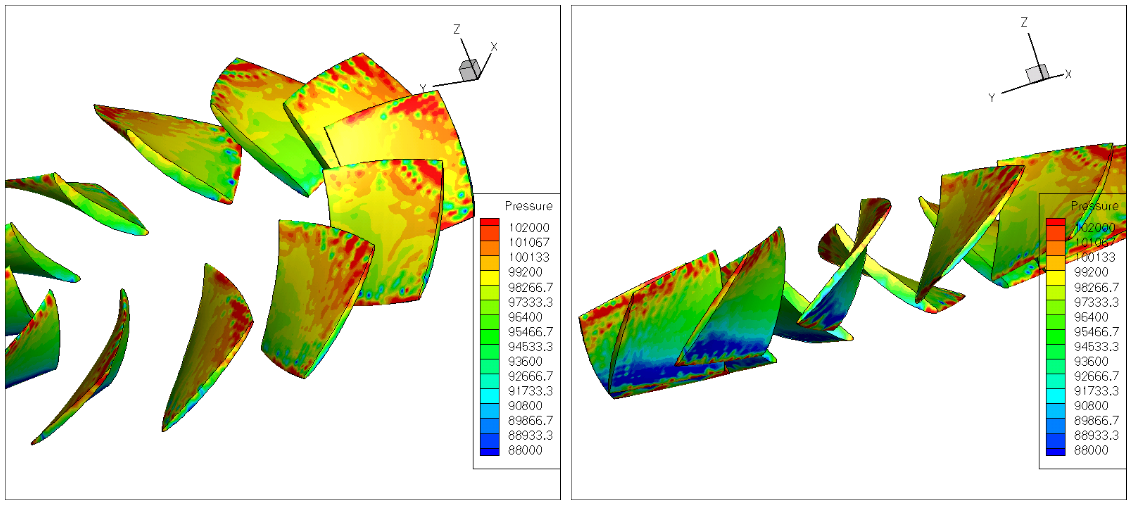


Figure 9.23: Compressor Rotor Optimization: Iso-Mach contours around rotor blades at the mid-step of its operation cycle. Dark blue regions are excluded from the flow simulation.



(a) (b)

Figure 9.24: Compressor Rotor Optimization: Pressure contours on the (a) pressure and (b) suction side of Inoue's rotor blades at the mid-step of its operation cycle.

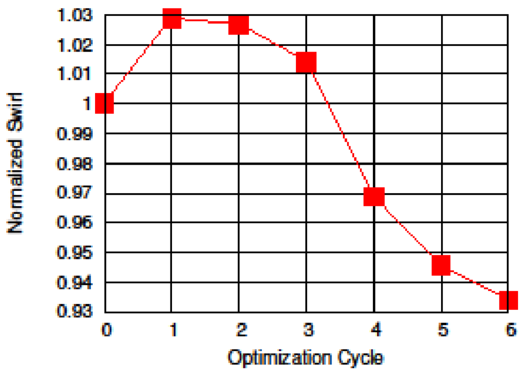
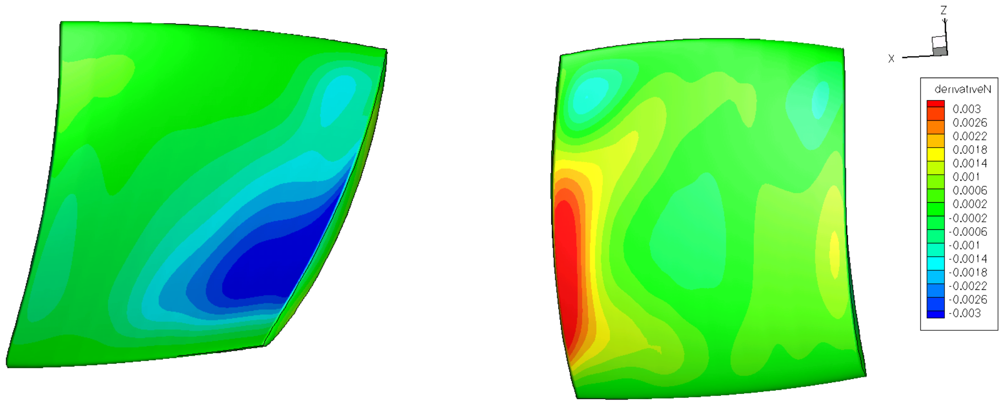


Figure 9.25: Compressor Rotor Optimization: Adjoint based optimization convergence supported by the iPGD method for swirl minimization at the exit. The objective function has been decreased by 7% after 6 optimization cycles.



(a) (b)

Figure 9.26: Compressor Rotor Optimization: Sensitivity map on the (a) pressure and (b) suction side of the blade. Blue and red colors indicate inward and outward displacements, respectively.

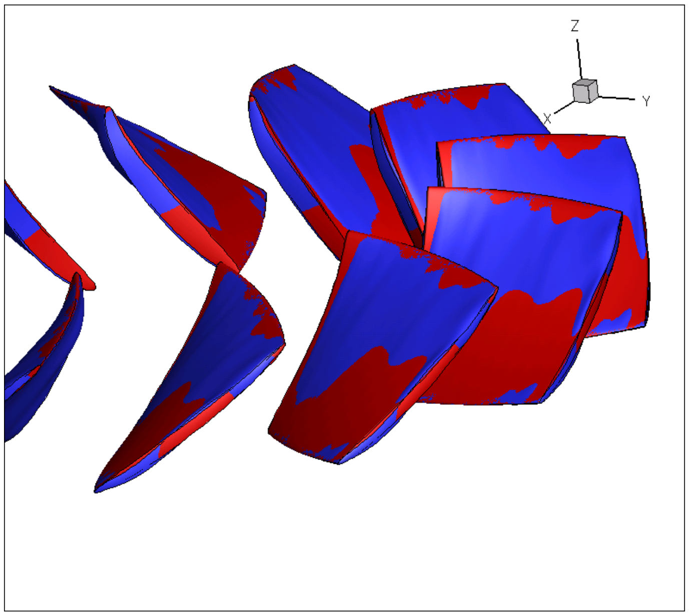


Figure 9.27: Compressor Rotor Optimization: Comparison between the background (red) and optimized (blue) geometry after 6 optimization cycles The blade is mostly displaced downwards minimizing the swirl at the outlet.

Chapter 10

Closure

The scope of this dissertation was the development of an integrated software for the CFD-based analysis and optimization in real-world applications concerning steady flow phenomena and flows around moving geometries. To this end, the cut-cell method was proposed as a robust and efficient alternative, which allows for an automated mesh generation around complex geometries, minimizing user intervention. In addition, the development of the continuous and discrete adjoint methods in Cartesian meshes offered a versatile computational tool for optimizing the shape of industrial products, which removes the limitations introduced by conventional body-conforming approaches. Section 10.1 summarizes the research presented throughout this Ph.D. thesis and section 10.2 describes its main concluding remarks. Sections 10.3 and 10.4 outline the novel contributions of this thesis in the scientific field of IBMs and adjoint methods. Finally, suggestions for future work are proposed in section 10.5.

10.1 Summary

The core of this Ph.D. thesis and the basis for any additionally developed tool is the mesh generation which is appropriate for the cut-cell method. The proposed algorithm is based on an octree data structure and starts by defining the tree's root cell, which coincides with the computational domain. Then, the root cell is isotropically subdivided into 4 (in 2D) or 8 (in 3D) offspring cells. The algorithm proceeds by repetitively splitting cells, guided by the presence of the immersed geometry, giving rise to new generations of cells. During the mesh generation, the maximum refinement level difference between neighboring cells is limited to one. Therefore, the division of intersected cells quickly propagates through the mesh, affecting cells far from the wall. The process continues until predefined criteria related to the cells' size are met. Thereafter, the mesh quality is increased by smoothing its resolution from dense regions, in the vicinity of the solid boundary, to coarser areas in the far-field, increasing the accuracy of the flow simulations.

A structured mesh indexing was introduced to identify each cell efficiently. This information was used to detect the parent and offspring of each cell in the tree-like hierarchy and compute its dimensions and centroid. These data are part of the proposed data structure, which exploits the flexibility of a face-based data structure, commonly used in unstructured meshes, and takes advantage of the unique nature of Cartesian meshes leading to a highly compact data set. Thus, the time-consuming, repetitive traversal of a considerable part of the tree was avoided. Algorithms for fast neighbor detection, mesh connectivity computation, and solid cells identification were also programmed and incorporated into the software.

Moreover, a robust algorithm for computing the intersection between the Cartesian mesh and an arbitrary geometry was developed. The proposed method was based on the Sutherland-Hodgman clipping algorithm, but it was considerably extended to accommodate the needs of the flow-solver and post-processor. Subsequently, the developed method's capabilities were demonstrated on various cases, considered demanding by most mesh generators due to their high complexity. Examples of Cartesian cells intersected in multiple regions or split into more than one finite volumes were given.

In addition, this thesis dealt with the so-called "small cell problem" by following a cell merging technique, which alleviates numerical instabilities during the flow solution caused by the considerable difference in size between adjacent cut-cells. According to this approach, one or more small cut-cells were geometrically merged with a larger neighbor creating a hyper-cell, which was treated as a regular cell, significantly simplifying the structure of the flow solver. Furthermore, a mesh partitioning technique was delivered based on the Hilbert space-filling curve to solve the governing equations in a multi-processor system. The presented algorithm was a 3D extension of 2D approaches found in the literature.

Moreover, the discretization of the compressible and incompressible flow equations in a Cartesian mesh was explained in detail. The presented numerical scheme benefited from the Cartesian mesh structure and adjusted to the complex geometry of cut-cells. It is based on a cell-centered, second-order finite volume approach employing the MUSCL scheme. The Roe's approximate Riemann solver was used to compute inviscid fluxes, and second-order accuracy was attained by extrapolating the flow variables at mesh faces conforming to the Taylor series expansion. The required gradients of the flow variables were computed through the least squares technique. The viscous flux between cells of different refinement levels was modified by employing orthogonal correction. Finally, the artificial compressibility method was applied to solve the incompressible flow equations.

This thesis also presented methods for predicting unsteady flows around impermeable moving boundaries within fixed Cartesian meshes. The time integration was based on an Arbitrary Lagrangian-Eulerian approach facilitated by a dual time stepping method which allowed for relatively large time steps, reducing the wall clock time of the simulation. At each time step, the mesh was re-adapted to the displaced immersed boundary, keeping track of its motion. Thus, regions close to the geometry's previous position were coarsened, and cells in the vicinity of the displaced wall were split anew, increasing the flow simulation's accuracy. At each time step, the necessary extrapolation of the flow solution to the subsequent mesh was taking advantage of the developed tree data structure, and thus, the flow solver's efficiency was not affected.

However, the displaced fluid-solid interface usually covers Cartesian cells changing their nature from fluid to solid and vice-versa. These transitions act like spurious sources or sinks, generating artificial oscillations traveling throughout the flow field and deteriorating the flow solution. Thus, this thesis proposed a novel cell linking method that mimics and improves the merging technique. In particular, each solidified cell was transferring its flow variables into a neighboring cell, which continued to exist at the next time step. The opposite process was followed for newly appeared cells in the fluid part of the mesh, equipping them with the appropriate time history that ensures smooth time integration. Therefore, strict flow conservation was maintained even for large boundary displacements, retaining the flow solver's efficiency.

Subsequently, this thesis developed a ghost-cell method for steady and unsteady flows, which proved more robust and simpler to implement but not as accurate compared to the cut-cell method. More specifically, the intersection between the immersed geometry and the mesh was not explicitly detected, allowing for the straightforward treatment of complex moving geometries. Instead, the presence of the solid wall within the flow field was expressed by the Signed Distance Function. Thus, the flow boundary conditions along the wall were indirectly imposed by solving an additional PDE for each primitive variable field in a thin layer of solid cells close to the fluid-solid interface. An investigation of the method's ability to satisfy conservation was presented, verifying the superiority of the cut-cell method in terms of accuracy.

Concerning shape optimization, the conjugate gradient method was used to explore the design space. Both the continuous and discrete adjoint formulations were employed to compute the derivatives of the objective function w.r.t. the design variables. In particular, the continuous adjoint method was introduced to the cut-cell and the ghost-cell methods for the first time in the literature. The adjoint ghost-cell software was developed in a GPU-based environment. The differentiation of the compressible and incompressible governing PDEs was described in detail resulting in the adjoint PDEs and the accompanied boundary conditions and sensitivity derivatives. Moreover, the adjoint Riemann problem was defined, and different solution approximations were adopted to build equal in number discretization schemes. The proposed adjoint schemes were equivalent to the FVS, HLLC, and Roe's Riemann solvers. Subsequently, the unsteady variant of the adjoint method was studied, and its backward in time integration was discussed. Data compression techniques were proposed to alleviate the increased demand for memory resources.

Moreover, the discrete adjoint formulation to the cut-cell method was also developed. Hand-differentiation was applied to both the compressible and incompressible solvers for steady and unsteady viscous flows. During the differentiation process, no simplifications were introduced, resulting in the exact discrete adjoint expressions and the accurate computation of the objective's gradient. In addition, a new mathematical notation was introduced, which allowed for a compact presentation of the discrete adjoint expressions. Furthermore, attention was paid to techniques for developing the corresponding adjoint software in a parallel processing environment.

This thesis also focused on properly treating the discrete adjoint time integration, especially in cases including moving geometries. That study proved that the adjoint field extrapolation to the mesh of the next time step should not follow the rules applied to the flow problem. Instead, alternative schemes were developed capable of computing the exact sensitivity derivatives. Additionally, the proper differentiation of algorithms treating the appearance and disappearance of cells from the fluid domain was demonstrated. Finally, filtering was implemented to eliminate the high-frequency signals from the derived sensitivity map, resulting in smooth-shaped optimized geometries.

Subsequently, the cut-cell mesh generator was differentiated w.r.t. the design variables resulting in the geometric sensitivities required for the objective's gradient computation. More specifically, the mathematical development concerned the differentiation of every geometric quantity included in the flow equations' discrete form. In addition, this approach incorporated the differentiation of the cut-cells' construction, highlighting the difference between the present method and the rest of conventional body-fitted approaches.

Finally, the developed computational tools for flow analysis and gradient-based optimization were assessed in detail. Experimental and numerical data were used for the flow solvers' validation/verification, and FDs were employed to ascertain the accuracy of the adjoint software. Moreover, a series of applications confirmed the ability of the aforementioned tools to deal with challenging industrial problems.

10.2 Concluding Remarks

This dissertation contributed to the need for fully automated and reliable computational tools employed for the analysis and optimization of practical applications. Its main concluding remarks are shortly presented. Initiating from the mesh generation, the proposed method minimizes the user intervention, automatically adapts the mesh in the vicinity of stationary or moving walls, and supports the flow solver providing solution-based refined meshes upon complex domains. Furthermore, detailed comparisons with experimental data assessed its ability to ensure the validity of conservation laws, retaining the flow solution's accuracy. Additional verification in unsteady applications proved its ability to successfully deal with covered and uncovered by the solid geometry cells, even for large boundary displacements.

The implementation of the artificial compressibility method was combined with the compressible flow solver resulting in a versatile software appropriate for all flow regimes. Its exhaustive assessment in internal and external, inviscid and laminar flows indicated the ability of the cut-cell approach to deliver realistic flow solutions, the accuracy of which is comparable to those obtained by body-conforming meshes. In the case of laminar flows, this study demonstrated that although simplicity of mesh generation comes at the cost of a non-aligned mesh on the wall, cut-cells' irregularities do not harm the accurate representation of the developed boundary layer. The presented results showed good agreement with experimental measurements and data provided by conventional CFD approaches.

Subsequently, the advantageous performance of the cut-cell method was exploited in a series of applications involving complex moving geometries of industrial interest. In particular, the flow simulation inside a valved duct indicated the method's superiority since it successfully handled the large displacement of the butterfly valve from the fully open to its closed position, avoiding mesh morphing techniques. A more challenging application was the flow simulation inside a scroll expander, and especially within the tight gap between the stationary and moving spirals of the machine. The flow solver's ability to maintain strict conservation was successfully tested in the case of the valveless diaphragm micropump, where the diaphragm was intensely deformed, covering and uncovering a significant number of cells at each time step. Finally, the flow simulation within an electrical submersible pump stage introduced the cut-cell method as an effective alternative to the Multiple Reference Frame, the Sliding Mesh, or other approaches dealing with the rotor-stator interaction problem.

The implementation of the adjoint theory to the cut-cell framework created an effective optimization tool overcoming mesh generation barriers. Contrary to the body-conforming approaches, the mesh deformation triggered by the geometry's shape modification was avoided, preventing the premature breakdown of the optimization loop. The Cartesian mesh restricted the mesh perturbation to the cut-cell zone, reducing the sensitivity derivatives computational cost and accelerating the optimization process. The adjoint compressible and incompressible flow solver was assessed by comparing the computed objective's gradient with FDs, resulting in almost zero deviations in cases involving stationary or moving solid bodies. Finally, the proposed post-processing of the sensitivity map successfully eliminated its high-frequency signals providing smooth-shaped optimized geometries in all the presented applications.

The high performance of the adjoint cut-cell software was confirmed through its implementation in several industrial applications, including the total pressure losses minimization of an S-shaped duct, the lift maximization of a wing, and the outlet tangential velocity minimization in a submersible pump. In all cases, the developed software proposed optimized solutions of adequately improved performance. Finally, the multi-objective optimization under uncertainties of a valveless diaphragm micropump incorporated additional strategies to the optimization loop. More specifically, the Polynomial Chaos Expansion implementation effectively reduced the computational cost of each evaluation, and the introduction of the proposed gradient-based method in the EASY platform [1] drove the optimization faster to the minimum.

10.3 Novel Contributions

This Ph.D. thesis has contributed to the scientific fields of IBMs and adjoint methods, developing original computational tools and implementing them in real-world applications. The main novelties of this research are summarized below.

- A new mesh data structure was proposed, which combines the beneficial features of CSAMR [144] and face-based structures. The new data structure contributes to the flow solver's low memory footprint by storing a compact data set. Moreover, it reduces the computational cost of the simulation by offering direct cell-to-cell and face-to-cell mappings.
- An integrated method for the accurate cut-cell construction was proposed. The corresponding algorithm was based on previously published works [6], but it was significantly extended to compute more topological data, facilitating the flow solver and the post-processor. The final algorithm is robust enough to handle all possible mesh-geometry intersections, including the Cartesian cell separation into more than one finite volumes.
- An alternative algorithm for generating cut-cells was suggested, which simplifies their construction, and thus, it is very efficient, easy to develop, and allows for a compact data structure.
- A new cell-merging method was presented, which combines clusters of cells to create hyper-volumes in 3D meshes. Although various cell-merging techniques

have already been proposed [340], the newly presented method takes additional mesh quality criteria under consideration, increasing the stability of the flow solver.

- The partitioner for 2D Cartesian meshes presented in [147] was extended to 3D cases, preserving its effectiveness by exploiting the unique features of the Hilbert space-filling curve.

- A new cell-linking algorithm was proposed, which deals with covered and uncovered cells due to the motion of solid bodies upon a stationary Cartesian mesh. The method was validated and used in challenging applications where large boundary displacements occur.

- During this thesis, the cut-cell method was introduced to applications, for which the CFD-based analysis is pretty rare due to their high complexity. In particular, it was used for the flow simulation within a scroll expander and an Electrical Submersible Pump stage.

- The continuous adjoint to the cut-cell and ghost-cell methods were presented for the first time in the literature and applied for 3D unsteady problems of inviscid flows.

- A theoretical investigation was presented about the continuous adjoint counterparts of the FVS, HLLC, and Roe's discretization schemes.

- The discrete adjoint to the cut-cell method for unsteady and viscous flow phenomena was first presented in this thesis. Although other relevant works have been published, they are restricted to the study of inviscid steady flows.

- The increased demand for computational memory during the unsteady adjoint PDEs' solution was treated using memory reduction based on the incremental SVD and PGD methods. In particular, the present thesis adopted these methods from relevant works [323], [236], and combined them with the cut-cell and ghost-cell adjoint solvers generating novel and cost-effective optimization tools.

10.4 List of Publications

Journal Articles:

- Y.-P. Vrionis, K. Samouchos, K. Giannakoglou. Topology Optimization in Fluid Mechanics Using Continuous Adjoint and the Cut-cell Method. Computers and Mathematics with Applications, 97:286-297, 2021.
- Y.-P. Vrionis, K. Samouchos, K. Giannakoglou. The Continuous Adjoint Cut-Cell Method for Shape Optimization in Cavitating Flows. Computers & Fluids, 224:104974, 2021.
- D. Kapsoulis, K. Samouchos, X. Trompoukis, K. Giannakoglou. Hybrid Optimization of a Valveless Diaphragm Micropump Using the Cut-Cell Method. Journal of Mechanics Engineering and Automation, 9:120-127, 2019.
- D. Kapsoulis, K. Samouchos, X. Trompoukis, K. Giannakoglou. Optimization under uncertainties of a valveless diaphragm pump Using the cut-cell method. The International Journal of Engineering and Science, 8(8):7-14, 2019.

Peer-Reviewed Conference Papers:

- K. Samouchos, D. Kapsoulis, X. Trompoukis, K. Giannakoglou. Shape Optimization of 3D Diaphragm Pumps Using the Continuous Adjoint Approach to the Cut-Cell Method. 10th International Conference on Computational Methods (ICCM2019), Singapore, July 9-13, 2019.
- Y.-P. Vrionis, K. Samouchos, K. Giannakoglou. Implementation of a Conservative Cut-Cell Method for the Simulation of Two-Phase Cavitating Flows. 10th International Conference on Computational Methods (ICCM 2019), Singapore, July 9-13, 2019.
- D. Kapsoulis, K. Samouchos, X. Trompoukis, K. Giannakoglou. Design-Optimization of a Valveless Diaphragm Micropump under Uncertainties Using Evolutionary Algorithms. International Conference on Adaptive Modeling and Simulation (ADMOS), El Campello, Spain, May 27-29, 2019.
- K. Samouchos, D. Kapsoulis, X. Trompoukis, K. Giannakoglou. Design of a Diaphragm Pump under Uncertainties Using the Continuous Adjoint to the

Cut-Cell Method. 6th European Conference on Computational Mechanics (ECCM 6) - 7th European Conference on Computational Fluid Dynamics (ECFD 7), Glasgow, UK, June 11-15, 2018.

- V. Papageorgiou, K. Samouchos, K. Giannakoglou. The Unsteady Continuous Adjoint Method Assisted by the Proper Generalized Method. EUROGEN 2017, International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems, Madrid, Spain, September 13-15, 2017.

- K. Samouchos, S. Katsanoulis, K. Giannakoglou. Unsteady Adjoint to the Cut-Cell Method Using Mesh Adaptation on GPU's. ECCOMAS Congress 2016, VII European Congress on Computational Methods in Applied Sciences and Engineering, Crete island, Greece, June 5-10, 2016.

Invited Lectures:

- K. Giannakoglou , E. Papoutsis-Kiachagias, K. Gkaragkounis, K. Samouchos, C. Vezyris, J. Koch. The Continuous Adjoint Method in Aerodynamic Optimization. von Karman Institute Lectures Series on Introduction to Optimization and Multidisciplinary Design, September 23-27, 2018.

10.5 Future Work Recommendations

This dissertation indicated the potential of the cut-cell method for flow analysis and optimization in challenging applications. Its novel contribution to these fields smoothed the path for further developments. Various recommendations for future research are exposed below.

A reasonable extension of the current work is its application to problems concerning turbulent flows. In such cases, a much larger mesh is usually needed to predict the boundary layer correctly, which considerably increases the simulation's computational cost. A remedy to this problem is the anisotropic cell subdivision during the mesh generation offering improved flexibility especially close to the solid boundary, where flattened cells are preferred. Other approaches change the discretization scheme close to the wall achieving reliable flow simulations in coarser meshes. Such a method places small line segments perpendicular to the wall and uses them to accurately compute normal derivatives on the wall [36]. Therefore, it mitigates local mesh irregularities, delivering smoother skin friction distributions along the solid boundaries.

The accuracy of unsteady flow simulations can be increased by combining Cartesian and structured meshes. According to that method, a body-conforming structured mesh is embodied around the moving geometry, following its motion upon a stationary background Cartesian mesh. Then, cells covered or uncovered by the structured mesh can be treated as explained in section 2.8. This method can further be improved by taking the flow field from the structured mesh into account to properly define the time history for these irregular cut-cells. Alternative approaches to this issue can be developed based on the space-time integration of the flow equations in 4D cells [219].

The continuous adjoint to the cut-cell method may be extended to turbulent flows by solving the RANS equations. Differentiating the turbulence model is suggested leading to the formulation of its adjoint counterpart [327]. This study is in progress in a Ph.D. thesis [325] carried out in PCOpt/LTT. The same Ph.D. thesis investigates the extension of the continuous adjoint to multi-phase flows in a cut-cell framework, where the homogeneous mixture model is employed to predict cavitating flows [326].

A valuable study on the discrete adjoint method concerns the development of strategies to reduce its relatively high memory requirements. Although a plethora of research has already been performed in approaches based on body-conforming meshes, none relevant work has been published yet about the effect of Cartesian meshes on the discrete equations and, consequently, to the computational memory usage. This can be achieved by assessing the impact of terms comprising the adjoint equations and the sensitivity derivatives expression on the accuracy of the objective function gradient. Hence, the less critical terms can be discarded in similar cases balancing accuracy and memory shavings.

Finally, an interesting field of research concerns reducing the significant storage requirements of unsteady adjoint due to its backward integration in time. Among various existing methods, this thesis adopted the SVD and PGD algorithms to compress data to be stored. An extension to this study would be the combination of these methods with other approaches such as the check-pointing technique [329]. Relevant research [192] conducted in PCOpt/LTT proposes the hybridization of

PGD and the ZFP compression library [182]. However, a decisive step forward would be defining an alternative adjoint problem governed by PDEs that could be solved forward in time and simultaneously with the flow equations, significantly reducing the memory and computational cost of the current approaches.

Appendix A

Identification of Cells in an Octree Data Structure

The mesh generator developed in this thesis is based on an octree data structure. According to subsection 2.2.2, the position of a cell in the tree is uniquely defined by the integer coordinates (i, j, k) or the index ID given by eq. 2.6. This Appendix proves this relation and ensures the unique bidirectional match of each cell to a single index ID for any developed octree.

Each level L of an octree, defined by eq. 2.3, represents a uniform 3D Cartesian mesh constituted by 2^L cells in each direction. Depending on the application, a different subset of these cells participates in the mesh generation process. However, the cell indexing by ID considers that all cells are used and need identification. Its computation is assisted by a local enumerator $\widetilde{ID}$ ranging at each level from 0 to $2^L \times 2^L \times 2^L - 1$. Then,

$$ID = \widetilde{ID} + C(L) \tag{A.1}$$

where $C(L)$ is the total number of cells constituting levels from 0 to $L - 1$.

Similarly, a local structured grid indexing $\left(\tilde{i}, \tilde{j}, \tilde{k}\right)$ is introduced as

$$\begin{aligned} \tilde{i} &= i - 2^L \\ \tilde{j} &= j - 2^L \\ \tilde{k} &= k - 2^L \end{aligned}$$

where 2^L is the minimum value of i, j, and k at each level. Then,

$$\begin{aligned}\widetilde{ID} &= 2^L 2^L \tilde{k} + 2^L \tilde{j} + \tilde{i} \\ &= 2^L 2^L \left(k - 2^L\right) + 2^L \left(j - 2^L\right) + \left(i - 2^L\right)\end{aligned} \tag{A.2}$$

implying that $\widetilde{ID} \in \left[0, 2^{3L} - 1\right]$. Moreover, ID continuously increases from each level to the next one, avoiding jumps in the cells' numbering. Therefore,

$$\begin{aligned}&ID_{max}(L) = ID_{min}(L+1) - 1 \Leftrightarrow \\ &\widetilde{ID}_{max}(L) + C(L) = \widetilde{ID}_{min}(L+1) + C(L+1) - 1 \Leftrightarrow \\ &C(L+1) = C(L) + 2^{3L}\end{aligned} \tag{A.3}$$

By definition, ID is zero for the root cell, and thus,

$$ID_{min}(0) = 0 \Leftrightarrow \widetilde{ID}_{min}(0) + C(0) = 0 \Leftrightarrow C(0) = 0 \tag{A.4}$$

Eqs. A.3 and A.4 imply that

$$C(L) = \sum_{l=1}^{L} 2^{3(l-1)} = \frac{8^L - 1}{8 - 1} \tag{A.5}$$

Finally, by substituting eqs. A.2 and A.5 to eq. A.1, the following relation arises,

$$ID = 4^L (k-1) + 2^L (j-1) + (i-1) - \frac{6}{7}\left(8^L - 1\right)$$

Appendix B

Fast Cut-Cell Construction

The treatment of the Cartesian mesh intersection by the geometry's surface is the most challenging task of the mesh generation process. Section 2.4 presents a method to accurately detect the fluid-solid interface and construct the corresponding cut-cells discarding their solid part. However, such methods are pretty complicated, and thus, are avoided by a significant number of researchers [57], [123], [110], [218]. Instead, simplified methods are preferred, which locally change the geometry's shape to facilitate the creation of cut-cells. This appendix proposes an approach in its 2D version, which allows for straightforward software development with low memory requirements.

Fig. B.1 shows a Cartesian cell in black, which is intersected by the solid blue boundary. The resulting cut-cell, computed by the method suggested in section 2.4, is gray shaded. According to a common simplification found in the literature, the red line segment replaces the actual solid boundary. This line connects the two intersection points, shown in the exact figure, and represents the only edge adjacent to the wall, converting the cut-cell's shape to a triangle. This modification allows exclusively triangular, quadrilateral or pentagonal cut-cells to appear, significantly simplifying the mesh data structure.

The proposed method handles all these shapes consistently by introducing a set of four variables ϕ_i for each cut-cell, where i indicates the vertices of the intersected quadrilateral. It is defined as the signed distance of each vertex from the newly defined solid segment. This information determines the shape of the cut-cell and is used to compute all geometrical data required by the flow solver. The method's implementation is straightforward and significantly reduces the memory usage by storing just the ϕ_i variables and computing the rest geometric quantities on the fly during the flow simulation. Moreover, it can upgrade other IBMs, like the ghost cell method, that already use the Signed Distance Function (SDF) in their implementation. In particular, a better representation of the solid boundary can easily be achieved by approximating the ϕ_i variables by the SDF and then applying the proposed method to introduce the cut-cells effect to the flow solution, increasing its accuracy.

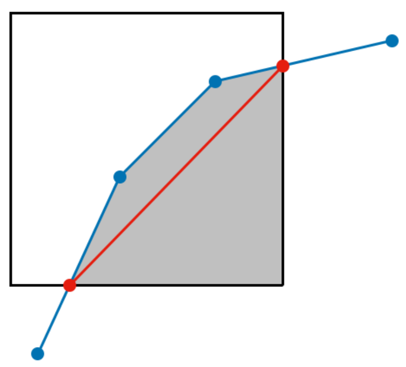


Figure B.1: A squared Cartesian cell in black is intersected by the solid blue boundary. The gray shaded area depicts the cut-cell resulting from the method followed in this thesis and presented in section 2.4. The red segment represents the geometry's reformation after applying the simplification suggested in this Appendix.

Hereafter, the mathematical formulation of the developed method is presented. Firstly, a local numbering of the cell's vertices and edges is introduced in fig. B.2a. The cell's length at each dimension is Δx and Δy. Furthermore, ϕ_i is considered positive for vertices located in the solid region of the mesh and negative for the rest. In the example of fig. B.2b, the red line demonstrates the location of the fluid-solid interface, and its distance from vertices v_0 and v_1 is $\phi_0 = -|\vec{b} - \vec{v}_0|$ and $\phi_1 = |\vec{d} - \vec{v}_1|$, respectively. Moreover, point a indicates its intersection with the square's edge, and $\vec{n}_s$ is the unit vector, normal to the solid edge.

Additionally, a new variable r_i is introduced at each edge i, defined as the ratio of the edge's fluid part to its total length. It is computed by using the similarity relation between triangles $(\widehat{v_0ab})$ and $(\widehat{v_0v_1c})$, which reads

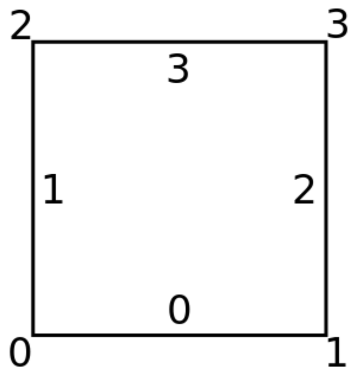
$$\frac{r_0\Delta x}{\Delta x} = \frac{-\phi_0}{\phi_1 - \phi_0} \Leftrightarrow r_0 = \frac{|\phi_0|}{|\phi_1| + |\phi_0|}$$

In an arbitrary edge defined by vertices v_s and v_e, ratio r_{se} is computed as

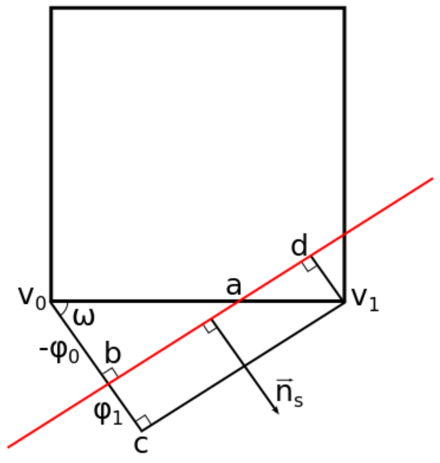
$$r_{se} = \frac{|min\{\phi_s, 0\}| + |min\{\phi_e, 0\}|}{|\phi_s| + |\phi_e|} \tag{B.1}$$

Furthermore, the triangle $(\widehat{v_0v_1c})$ of fig. B.2b is used to compute the first component of $\vec{n}_s$ as $n_{s_1} = cos(\omega) = (\phi_1 - \phi_0)/\Delta x$. Similarly, in an arbitrarily intersected cell, it can be proved that

$$\vec{n}_s = \left(\frac{\phi_1 - \phi_0}{\Delta x}, \frac{\phi_2 - \phi_0}{\Delta y}\right) \tag{B.2}$$



(a)



(b)

Figure B.2: (a) Local enumeration of the cell's nodes and edges. (b) A Cartesian cell is cut by the boundary of a solid body depicted by a straight red line. The geometric construction in the cell's bottom is used to compute the ratio r_i of each edge and the unit vector $\vec{n}_s$.

The length Δs of the solid edge is computed by using the relation

$$\int_c \vec{n}ds = \vec{0}$$

where c is the cell's boundary curve and $\vec{n}$ stands for the unit normal vector along the curve. The relation gives two equivalent ways for its computation,

$$\Delta s = \begin{cases} \dfrac{|r_2 - r_1|}{\phi_1 - \phi_0}\Delta A, & |\phi_1 - \phi_0| > \epsilon \\ \\ \dfrac{|r_3 - r_0|}{\phi_2 - \phi_0}\Delta A, & |\phi_2 - \phi_0| > \epsilon \end{cases} \tag{B.3}$$

where $\Delta A{=}\Delta x \Delta y$ and ϵ is a small user-defined number.

The cut-cell's area is computed by firstly defining vector $\vec{w}$ as the centroid of the quadrilateral of the background mesh. Its wall distance is

$$\phi_s = \frac{1}{4}\sum_{i=0}^{3}\phi_i$$

Then, the cut-cell is divided into triangles defined by their common vertex $\vec{w}$ and each of the cut-cell's edges. The area A_s of the triangle whose base coincides with the solid face is $0.5\Delta s \phi_s$. For the rest triangles, the following relation holds,

$$A_i = \frac{1}{4}r_i \Delta A, \quad i = 1, \cdots, 4$$

Their sum gives the total cut-cell area,

$$A = \frac{1}{4}\Delta A \sum_{i=0}^{3} r_i - \frac{1}{8}\Delta s \sum_{i=0}^{3}\phi_i \tag{B.4}$$

Similarly, the area's projection to each dimension is

$$\begin{aligned} A_x &= max\{r_0, r_3\}\Delta x \\ A_y &= max\{r_1, r_2\}\Delta y \end{aligned} \tag{B.5}$$

The midpoint $\vec{x}_i^e$ of each edge i is computed as the arithmetic mean of its boundary vertices and is expressed w.r.t. a Cartesian coordinate system, the origin of which is located at point $\vec{\omega}$,

$$\begin{aligned} \vec{x}_0^e &= \tfrac{1}{2}(s_0 r_0 \Delta x \quad , \quad -\Delta y) \\ \vec{x}_1^e &= \tfrac{1}{2}(\quad -\Delta x \quad , s_0 r_1 \Delta y) \\ \vec{x}_2^e &= \tfrac{1}{2}(\quad \Delta x \quad , s_1 r_2 \Delta y) \\ \vec{x}_3^e &= \tfrac{1}{2}(s_2 r_3 \Delta x \quad , \quad \Delta y) \end{aligned} \tag{B.6}$$

where s_i is the sign of ϕ_i. The midpoint coordinates of the solid edge are computed by the formulas

$$A = \int_c (x,0) \cdot \vec{n} ds = \int_c (0,y) \cdot \vec{n} ds$$

which indicate that

$$\begin{aligned} x_s^e &= \begin{cases} \dfrac{1}{n_{s_1} \Delta s} \left(A - \dfrac{r_1 + r_2}{2} \Delta A \right), & |n_{s_1}| > \epsilon \\ 0, & \text{otherwise} \end{cases} \\ y_s^e &= \begin{cases} \dfrac{1}{n_{s_2} \Delta s} \left(A - \dfrac{r_0 + r_3}{2} \Delta A \right), & |n_{s_2}| > \epsilon \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{B.7}$$

Finally, the aforementioned triangulation is used to compute the cut-cell's centroid $\vec{x}^c$, which equals the weighted average of the triangles' centroids,

$$A \vec{x}^c = \sum_{i=0}^{3} \vec{\bar{x}}_i A_i + \vec{x}^s A_s$$

where $\vec{\bar{x}}_i$ are the centroids of the triangles adjacent to the Cartesian edge i and $\vec{x}^s$ is the centroid of the triangle adjacent to the solid face. Therefore,

$$\vec{x}^c = \frac{1}{A} \left[\frac{1}{6} \Delta A \sum_{i=0}^{3} (r_i \vec{x}_i^e) - \frac{1}{3} \Delta s \vec{x}_s^e \sum_{i=0}^{3} \phi_i \right] \tag{B.8}$$

Consequently, the set of equations from B.1 to B.8 comprise a valuable tool to compute all the necessary geometric variables of a cut-cell by only storing the four ϕ_i variables. Moreover, they are valid for any cut-cell constructed under the discussed simplification, avoiding the time-consuming process of testing each intersection case separately.

However, these benefits come at the cost of an inaccurate boundary representation, questioning the proper imposition of the no-penetration and/or no-slip flow conditions. Indeed, the newly defined solid edge may not effectively reproduce the wall's impact on the flow, particularly close to high curvature boundaries. In this case, local mesh refinement is required to assure the simulation's accuracy. Therefore, the method presented in section 2.4 should be used whenever higher priority is given to the accuracy of the flow simulation than to the complexity of the developed software.

Appendix C

Optimal Value of the Artificial Compressibility Parameter

This Appendix presents the mathematical analysis which leads to the expression giving the optimal artificial compressibility parameter β in terms of numerical stability. This parameter is a positive real number and is used in the artificial compressibility approach for the incompressible flow equations discretization shown in section 3.5. According to this method the steady inviscid incompressible PDEs are

$$\frac{\partial V_i}{\partial \tau} + \frac{\partial f_{ik}}{\partial x_k} = 0, \quad i = 1, \cdots, 4,\ k = 1, \cdots, 3$$

where

$$\vec{V} = \begin{bmatrix} p \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}, \quad \vec{f_k} = \begin{bmatrix} \beta^2 v_k \\ v_1 v_k + \delta_{1k} p \\ v_2 v_k + \delta_{2k} p \\ v_3 v_k + \delta_{3k} p \end{bmatrix}$$

For more details the reader is referred to sections 3.4 and 3.5. Jacobian matrices over each direction are defined

$$A_{ijk} = \frac{\partial f_{ik}}{\partial V_j}$$

The Jacobian characterizing the 1D Riemann problem along the $\vec{n}$ direction is

$$A_k n_k = \begin{bmatrix} 0 & \beta^2 n_1 & \beta^2 n_2 & \beta^2 n_3 \\ n_1 & v_1 n_1 + v_n & v_1 n_2 & v_1 n_3 \\ n_2 & v_2 n_1 & v_2 n_2 + v_n & v_2 n_3 \\ n_3 & v_3 n_1 & v_3 n_2 & v_3 n_3 + v_n \end{bmatrix}$$

where $v_n = v_k n_k$. Its eigenvalues are u_n, u_n, $u_n + c$ and $u_n - c$ with $c = \sqrt{v_n^2 + \beta^2}$. Three distinctive variables are defined

$$\lambda_1 = u_n + c = u_n + |u_n|\xi$$
$$\lambda_2 = u_n$$
$$\lambda_3 = u_n - c = u_n - |u_n|\xi$$

with

$$\xi = \sqrt{1 + \left(\frac{\beta}{u_n}\right)^2}$$

where u_n is considered different from zero. Apparently, $\xi > 1$. The optimal β choice is the one that minimizes the largest ratio of wave speeds. Thus, the function to be minimized is

$$F = max\left|\frac{\lambda_i}{\lambda_j}\right| = \frac{max|\lambda_i|}{min|\lambda_j|}$$

Hence, the maximum and minimum absolute eigenvalues should be found for every u_n. Three cases are considered depending on the normal velocity's sign.

Case 1: $u_n > 0$

The three eigenvalues are $\lambda_1 = u_n(1+\xi)$, $\lambda_2 = u_n$ and $\lambda_3 = u_n(1-\xi)$. Their minimum and maximum absolute values should be found. It will be shown that $|\lambda_1|$ is greater than $|\lambda_2|$ and $|\lambda_3|$,

$$|\lambda_1| > |\lambda_2| \Leftrightarrow u_n(1+\xi) > u_n \Leftrightarrow \xi > 0$$

which is true. Also,

$$|\lambda_1| > |\lambda_3| \Leftrightarrow u_n(1+\xi) > -u_n(1-\xi) \Leftrightarrow 1 > -1$$

Therefore, $max|\lambda_i| = \lambda_1$. Subsequently, the relation between $|\lambda_2|$ and $|\lambda_3|$ is tested,

$$|\lambda_2| > |\lambda_3| \Leftrightarrow u_n > -u_n(1-\xi) \Leftrightarrow \xi < 2$$

Hence,

$$min|\lambda_i| = \begin{cases} \lambda_2 & \xi \geqslant 2 \\ -\lambda_3 & \xi < 2 \end{cases}$$

Consequently, the function to be minimized is

$$F = \begin{cases} \xi + 1 & \xi \geqslant 2 \\ \frac{\xi+1}{\xi-1} & \xi < 2 \end{cases}$$

F is continuous, decreasing for every $\xi \in (1,2]$ and increasing for every $\xi \in (2,+\infty]$. Therefore, it achieves its minimum at $\xi = 2$. The corresponding optimal β value is computed from ξ definition

$$\sqrt{1+\left(\frac{\beta}{u_n}\right)^2} = 2 \Leftrightarrow \beta^2 = 3u_n^2 \tag{C.1}$$

<u>*Case 2:* $u_n < 0$</u>

The three eigenvalues are $\lambda_1 = u_n(1-\xi)$, $\lambda_2 = u_n$ and $\lambda_3 = u_n(1+\xi)$. By following the same process as in the previous case, $max|\lambda_i| = \lambda_3$ and

$$min|\lambda_i| = \begin{cases} \lambda_2 & \xi \geqslant 2 \\ -\lambda_1 & \xi < 2 \end{cases}$$

The function to be minimized is again

$$F = \begin{cases} \xi + 1 & \xi \geqslant 2 \\ \frac{\xi+1}{\xi-1} & \xi < 2 \end{cases}$$

which achieves its minimum at $\xi = 2$. Thus, the optimal β value is

$$\beta^2 = 3u_n^2 \tag{C.2}$$

Case 3: $u_n = 0$

In this case ξ is not defined and the three eigenvalues are

$$\lambda_1 = u_n + c = \beta$$
$$\lambda_2 = 0$$
$$\lambda_3 = u_n - c = -\beta$$

Thus, $max|\lambda_i| = \lambda_1$ and $min|\lambda_i| = \lambda_2$ meaning that F is not defined. In order to reduce the spread of wave speeds created from discontinuities, the pseudo-compressibility parameter should be equal to a very small positive number, namely

$$\beta = \epsilon \tag{C.3}$$

Subsequently, by combining eqs. C.1, C.2 and C.3 one concludes that

$$\beta = max(\sqrt{3}|u_n|,\ \epsilon) \tag{C.4}$$

which is in agreement with eq. 3.32.

Appendix D

The Compressible and Incompressible Jacobian Matrices

According to chapter 3, the Jacobian matrix, defined by eq. 3.6, plays an essential role in discretizing the flow and adjoint equations. In particular, this matrix and its diagonalized form accompanies Roe's approximate Riemann solver and its adjoint counterpart, implemented in this thesis. Therefore, this Appendix presents the mathematical formulas for the compressible and incompressible Jacobian matrices, as well as the matrices comprising the corresponding eigenvalues and eigenvectors. Furthermore, the notation used below is explained in section 3.1.

Initially, the Jacobian matrix A_k at each Cartesian direction k is

$$A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -u^2+\hat{\gamma}|\vec{v}|^2/2 & (3-\gamma)u & -\hat{\gamma}v & -\hat{\gamma}w & \hat{\gamma} \\ -uv & v & u & 0 & 0 \\ -uw & w & 0 & u & 0 \\ -u(\gamma E-\hat{\gamma}|\vec{v}|^2) & \gamma E-\hat{\gamma}(|\vec{v}|^2/2+u^2) & -\hat{\gamma}vu & -\hat{\gamma}wu & \gamma u \end{bmatrix},$$

$$A_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ -vu & v & u & 0 & 0 \\ -v^2+\hat{\gamma}|\vec{v}|^2/2 & -\hat{\gamma}u & (3-\gamma)v & -\hat{\gamma}w & \hat{\gamma} \\ -vw & 0 & w & v & 0 \\ -v(\gamma E-\hat{\gamma}|\vec{v}|^2) & -\hat{\gamma}uv & \gamma E-\hat{\gamma}(|\vec{v}|^2/2+v^2) & -\hat{\gamma}wv & \gamma v \end{bmatrix},$$

$$A_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ -wu & w & 0 & u & 0 \\ -wv & 0 & w & v & 0 \\ -w^2 + \hat{\gamma}|\vec{v}|^2/2 & -\hat{\gamma}u & -\hat{\gamma}v & (3-\gamma)w & \hat{\gamma} \\ -w(\gamma E - \hat{\gamma}|\vec{v}|^2) & -\hat{\gamma}uw & -\hat{\gamma}vw & \gamma E - \hat{\gamma}(|\vec{v}|^2/2 + w^2) & \gamma w \end{bmatrix}$$

where $\vec{v} = (u, v, w)$ is the velocity vector and $\hat{\gamma} = \gamma - 1$.

Matrix $A_n = A_k n_k$ stands for the Jacobian matrix along an arbitrary direction defined by the unit normal vector $\vec{n}$. Its diagonalized form is $A_n = P\Lambda P^{-1}$ where

$$\Lambda = \begin{bmatrix} v_n \\ v_n \\ v_n \\ v_n + c \\ v_n - c \end{bmatrix},$$

$$P = \begin{bmatrix} n_1 & n_2 & n_3 & \hat{c} & \hat{c} \\ un_1 & un_2 - \rho n_3 & un_3 + \rho n_2 & \hat{c}u + \hat{\rho}n_1 & \hat{c}u - \hat{\rho}n_1 \\ vn_1 + \rho n_3 & vn_2 & vn_3 - \rho n_1 & \hat{c}v + \hat{\rho}n_2 & \hat{c}v - \hat{\rho}n_2 \\ wn_1 - \rho n_2 & wn_2 + \rho n_1 & wn_3 & \hat{c}w + \hat{\rho}n_3 & \hat{c}w - \hat{\rho}n_3 \\ a_1 & a_2 & a_3 & \hat{c}h_t + \hat{\rho}v_n & \hat{c}h_t - \hat{\rho}v_n \end{bmatrix},$$

and

$$P^{-1} = \begin{bmatrix} b_1 & ur_2n_1 & vr_2n_1 + r_1n_3 & wr_2n_1 - r_1n_2 & -r_2n_1 \\ b_2 & ur_2n_2 - r_1n_3 & vr_2n_2 & wr_2n_2 + r_1n_1 & -r_2n_2 \\ b_3 & ur_2n_3 + r_1n_3 & vr_2n_3 - r_1n_1 & wr_2n_3 & -r_2n_3 \\ r_3|\vec{v}|^2/2 - r_1v_n & r_1n_1 - ur_3 & r_1n_2 - vr_3 & r_1n_3 - wr_3 & r_3 \\ r_3|\vec{v}|^2/2 + r_1v_n & -r_1n_1 - ur_3 & -r_1n_2 - vr_3 & -r_1n_3 - wr_3 & r_3 \end{bmatrix}$$

Additionally, $v_n = \vec{v} \cdot \vec{n}$, $\hat{\rho} = \rho/2$, $\hat{c} = \hat{\rho}/c$, $r_1 = 1/\rho$, $r_2 = \hat{\gamma}/c^2$, $r_3 = \hat{\gamma}/(\rho c)$, and

$$\vec{a} = |\vec{v}|^2\vec{n} + \rho\,(\vec{v} \times \vec{n}),$$
$$\vec{b} = \left(1 - \frac{\gamma - 1}{2}M^2\right)\vec{n} - \frac{1}{\gamma}\,(\vec{v} \times \vec{n})$$

Similarly, the preconditioned Jacobian matrix A_k^Γ for incompressible flows, defined by eq. 3.29, is

$$A_1^\Gamma = \begin{bmatrix} 0 & \beta^2 & 0 & 0 \\ 1 & 2u & 0 & 0 \\ 0 & v & u & 0 \\ 0 & w & 0 & u \end{bmatrix}, \quad A_2^\Gamma = \begin{bmatrix} 0 & 0 & \beta^2 & 0 \\ 0 & v & u & 0 \\ 1 & 0 & 2v & 0 \\ 0 & 0 & w & v \end{bmatrix}, \quad A_3^\Gamma = \begin{bmatrix} 0 & 0 & 0 & \beta^2 \\ 0 & w & 0 & u \\ 0 & 0 & w & v \\ 1 & 0 & 0 & 2w \end{bmatrix}$$

The diagonalization of $A_n^\Gamma = A_k^\Gamma n_k$ implies $A_n^\Gamma = P^\Gamma \Lambda^\Gamma \left(P^\Gamma\right)^{-1}$, where

$$\Lambda^\Gamma = \begin{bmatrix} v_n & v_n & v_n + \tilde{c} & v_n - \tilde{c} \end{bmatrix}^T,$$

$$P^\Gamma = \begin{bmatrix} 0 & 0 & \tilde{c} & -\tilde{c} \\ t_1^1 & t_1^2 & n_1 + us_1 & n_1 + us_2 \\ t_2^1 & t_2^2 & n_2 + vs_1 & n_2 + vs_2 \\ t_3^1 & t_3^2 & n_3 + ws_1 & n_3 + ws_2 \end{bmatrix},$$

and

$$\left(P^\Gamma\right)^{-1} = \begin{bmatrix} (\vec{t}^2 \times \vec{v}) \cdot \vec{n}/\tilde{c}^2 & \tilde{t}_1^1 & \tilde{t}_2^1 & \tilde{t}_3^1 \\ (\vec{v} \times \vec{t}^1) \cdot \vec{n}/\tilde{c}^2 & \tilde{t}_1^2 & \tilde{t}_2^2 & \tilde{t}_3^2 \\ (\tilde{c} - v_n)/(2\tilde{c}^2) & \tilde{n}_1/2 & \tilde{n}_2/2 & \tilde{n}_3/2 \\ -(\tilde{c} + v_n)/(2\tilde{c}^2) & \tilde{n}_1/2 & \tilde{n}_2/2 & \tilde{n}_3/2 \end{bmatrix}$$

Vectors $\vec{t}^1$ and $\vec{t}^2$ are defined such that the set $\{\vec{t}^1, \vec{t}^2, \vec{n}\}$ forms an orthonormal basis of $\mathbb{R}^3$. Its orientation is dictated by the relation $\vec{t}^1 \times \vec{t}^2 = \vec{n}$. Moreover, the artificial sound speed $\tilde{c}$ is defined by eq. 3.33. Finally, the temporary variables used in the above expressions are

$$s_1 = \frac{v_n + \tilde{c}}{\beta^2}, \quad s_2 = \frac{v_n - \tilde{c}}{\beta^2}$$

and

$$\begin{aligned} \vec{\tilde{n}} &= \lambda^2 \vec{n}, \\ \vec{\tilde{t}}^1 &= \lambda^2 \vec{t}^1 + \frac{v_n}{\tilde{c}^2} \left(\vec{t}^2 \times \vec{v}\right), \\ \vec{\tilde{t}}^2 &= \lambda^2 \vec{t}^2 + \frac{v_n}{\tilde{c}^2} \left(\vec{v} \times \vec{t}^1\right) \end{aligned}$$

where $\lambda = \beta/\tilde{c}$.

Appendix E

Approximate Riemann Solver of Roe

The purpose of this Appendix is to present the mathematical development leading to the final expression of the Roe scheme, which is one of the best-known approximate Riemann solvers used by the Godunov discretization method. The proof is partly based on [306]. Riemann's problem mathematical expression for the unsteady 3D Euler equations is presented below. The equations are

$$\frac{\partial U_i}{\partial t} + \frac{\partial f_i}{\partial x} = 0 \quad i = 1, \cdots, N \tag{E.1}$$

$$U_i(x, t = 0) = \begin{cases} U_i^L & x \leqslant 0 \\ U_i^R & x > 0 \end{cases}$$

where the system of $N \times N$ PDEs is conservative, hyperbolic and non-linear. $\vec{U}$ is the vector of conservative variables and $\vec{f}$ is the flux. The Jacobian matrix $A(\vec{U})$ is introduced as

$$A_{ij} = \frac{\partial f_i}{\partial U_j}$$

and is used to rewrite the system of PDEs in a non-conservative form as

$$\frac{\partial U_i}{\partial t} + \frac{\partial f_i}{\partial x} = 0 \Leftrightarrow \frac{\partial U_i}{\partial t} + \frac{\partial f_i}{\partial U_j}\frac{\partial U_j}{\partial x} = 0 \Leftrightarrow \frac{\partial U_i}{\partial t} + A_{ij}\frac{\partial U_j}{\partial x} = 0$$

where repetitive indices imply summation. On the other hand, the summation symbol will be used only in case the summation lower and upper limits should be explicitly mentioned. Although the Riemann problem analytical solution is available, it is computationally costly. On the contrary, the Riemann problem solution of a linear PDEs' system is relatively simpler and numerically faster. Therefore, Roe's approach intends to exactly solve an approximate linear Riemann problem. It arises by replacing Jacobian matrix $A(\vec{U})$ with the new constant matrix $\tilde{A}(\vec{U}^L, \vec{U}^R)$,

$$\begin{aligned} \frac{\partial \hat{U}_i}{\partial t} + \tilde{A}_{ij}\frac{\partial \hat{U}_j}{\partial x} =& 0 \quad i, j = 1, \cdots, N \\ \hat{U}_i(x, t=0) =& \begin{cases} U_i^L & x \leqslant 0 \\ U_i^R & x > 0 \end{cases} \end{aligned} \tag{E.2}$$

where $\vec{\hat{U}}$ stands for the exact solution of the approximate Riemann problem. According to Roe's approach, the new Jacobian matrix satisfies three requirements. Firstly, the new system of equations should retain hyperbolicity, meaning that $\tilde{A}$ should have N real eigenvalues and a corresponding set of linearly independent right eigenvectors, meaning that $\tilde{A}$ is diagonalizable and, therefore, can be written as

$$\tilde{A} = \tilde{P}\tilde{\Lambda}\tilde{P}^{-1}$$

where $\tilde{\Lambda} = diag(\tilde{\lambda}_1, \cdots, \tilde{\lambda}_N)$ with λ_i being the Jacobian's eigenvalues, and columns of $\tilde{P}$ are the right eigenvectors. Values in $\tilde{\Lambda}$ diagonal are positioned in ascending order ($\tilde{\lambda}_i \leqslant \tilde{\lambda}_{i+1}$). Secondly, $\tilde{A}$ should ensure consistency by requiring

$$\tilde{A}(\vec{U}^L = \vec{U}, \vec{U}^R = \vec{U}) = A(\vec{U})$$

Finally, the new PDEs should remain conservative across discontinuities, which is expressed as

$$\vec{f}(\vec{U}^R) - \vec{f}(\vec{U}^L) = \tilde{A}(\vec{U}^R - \vec{U}^L)$$

This identity also ensures an exact wave recognition in case a single, isolated discontinuity separates the left and right initial conditions. The construction of a matrix that satisfies all three criteria is based on the definition of the Roe averaged variables $\vec{\tilde{U}}(\vec{U}^L, \vec{U}^R)$ according to which

$$\tilde{A} = A(\vec{\tilde{U}})$$

Their exact expressions are given by eq. 3.8.

Next step is the approximate Riemann problem solution. To do so, the characteristic

Riemann variables are introduced

$$\vec{W} = \tilde{P}^{-1}\vec{\hat{U}}$$

Eq. E.2 becomes

$$\begin{aligned}
&\frac{\partial \hat{U}_i}{\partial t} + \tilde{A}_{ij}\frac{\partial \hat{U}_j}{\partial x} = 0 \Leftrightarrow \frac{\partial \hat{U}_i}{\partial t} + \tilde{P}_{im}\tilde{\Lambda}_{mn}\tilde{P}^{-1}_{nj}\frac{\partial \hat{U}_j}{\partial x} = 0 \Leftrightarrow \\
&\tilde{P}^{-1}_{im}\frac{\partial \hat{U}_m}{\partial t} + \tilde{\Lambda}_{im}\tilde{P}^{-1}_{mj}\frac{\partial \hat{U}_j}{\partial x} = 0 \Leftrightarrow \frac{\partial (\tilde{P}^{-1}_{im}\hat{U}_m)}{\partial t} + \tilde{\Lambda}_{im}\frac{\partial (\tilde{P}^{-1}_{mj}\hat{U}_j)}{\partial x} = 0 \Leftrightarrow \\
&\frac{\partial W_i}{\partial t} + \tilde{\Lambda}_{im}\frac{\partial W_m}{\partial x} = 0
\end{aligned}$$

The previous procedure decouples the system of PDEs. By defining the characteristic curves $dx/dt = \tilde{\lambda}_i$ in the space-time plane, the governing equations become

$$\frac{\partial W_i}{\partial t} + \tilde{\Lambda}_{im}\frac{\partial W_m}{\partial x} = 0 \Leftrightarrow \frac{\partial W_i}{\partial t} + \frac{\partial W_i}{\partial x}\frac{dx}{dt} = 0 \Leftrightarrow \frac{DW_i\left(x(t), t\right)}{Dt} = 0$$

which means that W_i remains constant along the i^{th} characteristic curve. DW_i/Dt stands for the total derivative of W_i. Considering that $\tilde{\lambda}_i$ is constant, the curves are the straight lines $x = \tilde{\lambda}_i t + x_0$, which implies that $\vec{W}_i(x,t)$ is equal to $\vec{W}_i(x_0, 0)$, where x_0 is the intersection point between the x-axis and the line with $\tilde{\lambda}_i$ slope passing from (x,t). In other words

$$W_i(x,t) = W_i(x_0) = W_i(x - \tilde{\lambda}_i t) = \begin{cases} W^L_{0i} & x_0 \leqslant 0 \\ W^R_{0i} & x_0 > 0 \end{cases}$$

where $\vec{W}^L_0 = \tilde{P}^{-1}\vec{U}^L$ and $\vec{W}^R_0 = \tilde{P}^{-1}\vec{U}^R$. Finally, by using the invariants definition, the approximate Riemann problem solution is

$$\begin{aligned}
&W_i(x,t) = \sum_{m=1}^{N}\tilde{P}^{-1}_{im}\hat{U}_m(x,t) \Leftrightarrow \hat{U}_i(x,t) = \sum_{m=1}^{N}\tilde{P}_{im}W_m(x,t) \Leftrightarrow \\
&\hat{U}_i(x,t) = \sum_{m=1}^{\hat{m}}\tilde{P}_{im}W_m(x - \tilde{\lambda}_m t) + \sum_{m=\hat{m}+1}^{N}\tilde{P}_{im}W_m(x - \tilde{\lambda}_m t) \Leftrightarrow \\
&\hat{U}_i(x,t) = \sum_{m=1}^{\hat{m}}\tilde{P}_{im}W^R_{0m} + \sum_{m=\hat{m}+1}^{N}\tilde{P}_{im}W^L_{0m}
\end{aligned}$$

where $\hat{m}$ is the maximum value for which $x - \tilde{\lambda}_{\hat{m}} t > 0$. The last equation shows that

$\vec{\tilde{U}}(x,t)$ is affected only by $\hat{m}$, which depends on the inequality $x/t > \tilde{\lambda}$ satisfaction. Thus, $\vec{U}$ is a 1D function of x/t. Fig. E.1 represents the exact solution graphically. Three lines represent the discontinuity propagation starting from the axis' origin and separating the space-time plane in regions of constant $\vec{\tilde{U}}(x,t)$. Another set of dashed gray lines represents the characteristic lines in which $\vec{W}$ is constant. Their confluence specifies the value of $\vec{\tilde{U}}(x,t)$ field.

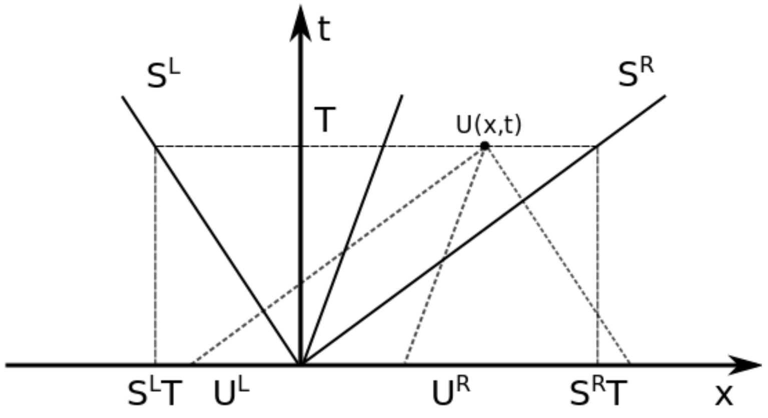


Figure E.1: Structure of the Riemann problem solution of a linear hyperbolic system.

According to Godunov's method the mean flux along $x\!=\!0$ line should be found, or equivalently $\vec{f}(\vec{U}(0))$. Firstly, $\vec{U}(0)$ is computed. According to the previous analysis

$$\hat{U}_i(0) = \sum_{m=1}^{\hat{m}} \tilde{P}_{im} W_{0m}^R + \sum_{m=\hat{m}+1}^{N} \tilde{P}_{im} W_{0m}^L \tag{E.3}$$

where $\lambda_{\hat{m}} < 0$ and $\lambda_{\hat{m}+1} > 0$. Moreover, it's true that

$$U_i^L = \sum_{m=1}^{N} \tilde{P}_{im} W_{0m}^L \tag{E.4}$$

and

$$U_i^R = \sum_{m=1}^{N} \tilde{P}_{im} W_{0m}^R \tag{E.5}$$

By subtracting eq. E.3 from eqs. E.4 and E.5 one gets

$$\hat{U}_i(0) = U_i^L + \sum_{m=1}^{\hat{m}} \tilde{P}_{im} (W_{0m}^R - W_{0m}^L) \tag{E.6}$$

and

$$\hat{U}_i(0) = U_i^R - \sum_{m=\hat{m}+1}^{N} \tilde{P}_{im}(W_{0m}^R - W_{0m}^L) \tag{E.7}$$

Subsequently, the $\vec{f}^0 = \vec{f}(\vec{U}(0))$ computation follows. Firstly, the exact flow eqs. E.1 are integrated in the $[S^LT, 0] \times [0, T]$ control volume, where S^L is the smallest wave velocity, T is a chosen time window and S^LT is the length the wave has traveled within time T, fig. E.1,

$$\begin{aligned}
&\int_0^T \int_{S^LT}^0 \frac{\partial U_i}{\partial t} dxdt + \int_0^T \int_{S^LT}^0 \frac{\partial f_i}{\partial x} dxdt = 0 \Leftrightarrow \\
&\int_{S^LT}^0 \left(U_i(x,T) - U_i(x,0)\right) dx + \int_0^T \big(f_i(0,t) - f_i(S^LT,t)\big)\, dt = 0 \Leftrightarrow \\
&\int_{S^LT}^0 U_i(x,T) dx + U_i^L S^L T + f_i^0 T - f_i^L T = 0
\end{aligned}$$

where $\vec{f}^L = \vec{f}(\vec{U}^L)$. Thus, the exact expression for the flux numerical computation is obtained from

$$f_i^0 = f_i^L - U_i^L S^L - \frac{1}{T} \int_{S^LT}^0 U_i(x,T) dx \tag{E.8}$$

The unknown integral of eq. E.8 is approximated by the solution obtained by the linearized Riemann problem of eq. E.2. By integrating the PDEs in the same control volume one gets

$$\begin{aligned}
&\int_0^T \int_{S^LT}^0 \frac{\partial \hat{U}_i}{\partial t} dxdt + \int_0^T \int_{S^LT}^0 \tilde{A}_{ij} \frac{\partial \hat{U}_j}{\partial x} dxdt = 0 \Leftrightarrow \\
&\int_0^T \int_{S^LT}^0 \frac{\partial \hat{U}_i}{\partial t} dxdt + \int_0^T \int_{S^LT}^0 \frac{\partial (\tilde{A}_{ij} \hat{U}_j)}{\partial x} dxdt = 0 \Leftrightarrow \\
&\int_{S^LT}^0 \hat{U}_i(x,T) dx + \hat{U}_i^L S^L T + \tilde{A}_{ij} \hat{U}_j(0) T - \tilde{A}_{ij} U_j^L T = 0 \Leftrightarrow \\
&\int_{S^LT}^0 \hat{U}_i(x,T) dx = \tilde{A}_{ij} U_j^L T - \tilde{A}_{ij} \hat{U}_j(0) T - U_i^L S^L T
\end{aligned} \tag{E.9}$$

where $\hat{U}_j(0)$ stands for the constant flow solution along the t-axis. Substitution of eq. E.9 into eq. E.8 gives

$$f_i^0 = f_i^L + \tilde{A}_{ij}(\hat{U}_j(0) - U_j^L)$$

and by using eq. E.6

$$f_i^0 = f_i^L + \sum_{j=1}^{N}\left[\tilde{A}_{ij}\sum_{m=1}^{\hat{m}}\tilde{P}_{jm}(W_{0m}^R - W_{0m}^L)\right] \Leftrightarrow$$

$$f_i^0 = f_i^L + \sum_{m=1}^{\hat{m}}\tilde{P}_{im}\tilde{\lambda}_m(W_{0m}^R - W_{0m}^L) \tag{E.10}$$

Similarly, by integrating eqs. E.1 and E.2 in the $[0, S^R T] \times [0, T]$ control volume, where S^R is the largest wave velocity, one gets

$$f_i^0 = f_i^R - \sum_{m=\hat{m}+1}^{N}\tilde{P}_{im}\tilde{\lambda}_m(W_{0m}^R - W_{0m}^L) \tag{E.11}$$

where $\vec{f}^R = \vec{f}(\vec{U}^R)$. The eqs. E.10 and E.11 summation results in

$$2f_i^0 = f_i^L + f_i^R + \sum_{m=1}^{\hat{m}}\tilde{P}_{im}\tilde{\lambda}_m(W_{0m}^R - W_{0m}^L) - \sum_{m=\hat{m}+1}^{N}\tilde{P}_{im}\tilde{\lambda}_m(W_{0m}^R - W_{0m}^L) \Leftrightarrow$$

$$f_i^0 = \frac{1}{2}(f_i^L + f_i^R) - \frac{1}{2}\sum_{m=1}^{N}\tilde{P}_{im}|\tilde{\lambda}_m|(W_{0m}^R - W_{0m}^L)$$

The last equation is valid because

$$\lambda_m = \begin{cases} -|\lambda_m|, & m \leqslant \hat{m} \\ \ \ |\lambda_m|, & m > \hat{m} \end{cases}$$

Substitution of $\vec{W}_0^L$ and $\vec{W}_0^R$ definition into the above equation gives

$$f_i^0 = \frac{1}{2}(f_i^L + f_i^R) - \frac{1}{2}\sum_{m=1}^{N}\tilde{P}_{im}|\tilde{\lambda}_m|\tilde{P}_{mj}^{-1}(U_j^R - U_j^L)$$

The Roe scheme final expression is usually written as

$$f_i^0 = \frac{1}{2}(f_i^L + f_i^R) - \frac{1}{2}|\tilde{A}_{ij}|(U_j^R - U_j^L)$$

where absolute Jacobian matrix is defined as $|\tilde{A}_{ij}| = \sum_{m=1}^{N}\tilde{P}_{im}|\tilde{\lambda}_m|\tilde{P}_{mj}^{-1}$.

Appendix F

Approximate Riemann Solver of Roe for Preconditioned Conservative Laws

This appendix describes a prolongation of the approximate Riemann Solver of Roe presented in Appendix E. It focuses on conservative, non-linear PDEs which require a preconditioned method to alter their mathematical behavior and easily be handled by the current discretization methods. Such PDEs are the incompressible equations discussed in section 3.5. The general case of the following 1D preconditioned PDE

$$\Gamma_{ij}^{-1}\frac{\partial U_j}{\partial t}+\frac{\partial f_i}{\partial x}=0$$

is investigated, where $\vec{U}$ and $\vec{f}$ are the conservative variables and flux vectors and Γ stands for the preconditioning matrix. A set of variables $\vec{W}$ is defined such that

$$\frac{\partial W_i}{\partial U_j}=\Gamma_{ij}^{-1}$$

Therefore, the above PDE becomes

$$\frac{\partial W_i}{\partial U_j}\frac{\partial U_j}{\partial t}+\frac{\partial f_i}{\partial x}=0\Leftrightarrow\frac{\partial W_i}{\partial t}+\frac{\partial f_i}{\partial W_j}\frac{\partial W_j}{\partial x}=0\Leftrightarrow\frac{\partial W_i}{\partial t}+A_{ij}^{\Gamma}\frac{\partial W_j}{\partial x}=0$$

The final PDE is hyperbolic and A^Γ is diagonalizable. The Jacobian and the preconditioned Jacobian matrices are defined as

$$A_{ij} = \frac{\partial f_i}{\partial U_j}$$
$$A^\Gamma_{ij} = \frac{\partial f_i}{\partial W_j}$$

They are related through the preconditioning matrix,

$$A^\Gamma_{ij} = \frac{\partial f_i}{\partial W_j} = \frac{\partial f_i}{\partial U_m}\frac{\partial U_m}{\partial W_j} = A_{im}\Gamma_{mj} \tag{F.1}$$

The Roe approach accurately solves the Riemann problem for the approximate linearized PDE

$$\frac{\partial W_i}{\partial t} + \tilde{A}^\Gamma \frac{\partial W_j}{\partial x} = 0$$

where $\tilde{A}^\Gamma$ is a constant matrix satisfying the Roe criteria. The initial values $(\vec{W}^L, \vec{W}^R)$ of the corresponding Riemann problem are chosen to satisfy the following condition

$$W_i^R - W_i^L = \tilde{\Gamma}^{-1}_{ij}(U_j^R - U_j^L)$$

According to Appendix E, Roe's scheme is

$$f_i^0 = \frac{1}{2}(f_i^L + f_i^R) - \frac{1}{2}|\tilde{A}^\Gamma_{ij}|(W_j^R - W_j^L)$$

which is rewritten as

$$f_i^0 = \frac{1}{2}(f_i^L + f_i^R) - \frac{1}{2}|\tilde{A}_{im}\tilde{\Gamma}_{mn}|\tilde{\Gamma}^{-1}_{nj}(U_j^R - U_j^L)$$

where eq. F.1 has been used. The preconditioned Jacobian eigenvectors $(\vec{p})$ and eigenvalues (λ_p) satisfy the expression

$$(A\Gamma)\vec{p} = \lambda_p \vec{p}$$

which is properly modified as

$$\Gamma(A\Gamma)\vec{p} = \lambda_p \Gamma\vec{p} \Leftrightarrow (\Gamma A)(\Gamma\vec{p}) = \lambda_p(\Gamma\vec{p}) \Leftrightarrow (\Gamma A)\vec{q} = \lambda_p \vec{q}$$

Therefore, ΓA is diagonalizable having independent eigenvectors $\vec{q} = \Gamma\vec{p}$ and real eigenvalues $\lambda_q = \lambda_p$.

The two matrices are expressed in diagonalizable form,

$$\begin{aligned}(A\Gamma) =& P\Lambda_P P^{-1}\\ (\Gamma A) =& Q\Lambda_Q Q^{-1}\end{aligned}$$

where $\Lambda_P = \Lambda_Q$ and $Q = \Gamma P$. Based on the previous analysis, matrices in Roe's scheme can be expressed as

$$|A\Gamma|\Gamma^{-1} = P|\Lambda_P|P^{-1}\Gamma^{-1} = \Gamma^{-1}\Gamma P|\Lambda_P|P^{-1}\Gamma^{-1} =$$
$$\Gamma^{-1}(\Gamma P)|\Lambda_P|(\Gamma P)^{-1} = \Gamma^{-1}Q|\Lambda_Q|Q^{-1} = \Gamma^{-1}|\Gamma A|$$

Hence, the Roe scheme is transformed to a more convenient form,

$$f_i^0 = \frac{1}{2}(f_i^L + f_i^R) - \frac{1}{2}\tilde{\Gamma}_{im}^{-1}|\tilde{\Gamma}_{mn}\tilde{A}_{nj}|(U_j^R - U_j^L)$$

If the initial PDE is multiplied by Γ, it becomes

$$\frac{\partial U_i}{\partial t} + \Gamma_{ij}\frac{\partial f_j}{\partial x} = 0$$

The discretization scheme should also be multiplied by Γ. Therefore,

$$f_i^{\Gamma,0} = \Gamma f_i^0 = \frac{1}{2}(f_i^{\Gamma,L} + f_i^{\Gamma,R}) - \frac{1}{2}|\tilde{\Gamma}_{im}\tilde{A}_{mj}|(U_j^R - U_j^L) \tag{F.2}$$

where $\vec{f}^{\,\Gamma,L} = \Gamma\vec{f}^{\,L}$ and $\vec{f}^{\,\Gamma,R} = \Gamma\vec{f}^{\,R}$. Last expression agrees with eq. 3.28 presented in section 3.5.

Appendix G

Monotone and TVD schemes relation

This Appendix is concerned with the study of monotone schemes and their relation to the set of Total Variation Diminishing (TVD) schemes applied to the discretization of a 1D, non-linear, scalar PDE. More specifically, it will be proved that monotone schemes are a subclass of TVD schemes. The proof is mostly based on [121], [122]. The following single conservation law is considered

$$\frac{\partial u}{\partial t}+\frac{\partial f(u)}{\partial x}=0$$

which is discretized in the following way

$$v_i^{n+1}=H\left(v_{i-k}^n,v_{i-k+1}^n,\cdots,v_i^n,v_{i+1}^n,\cdots,v_{i+k}^n\right)$$

with k being a non-negative integer indicating the range of nodes the discretization scheme takes into account. Variables v_i^n stand for the numerical approximation of $u(x,t)$ field on node i and at time step n. The scheme H is said to be monotone if

$$\frac{\partial H}{\partial v_i^n}\geqslant 0,\quad \forall i$$

The above discretization scheme can be rewritten in the following conservative form

$$v_i^{n+1}=u_i^n-c\left[h\left(v_{i-k+1}^n,v_{i-k+2}^n,\cdots,v_{i+k}^n\right)-h\left(v_{i-k}^n,v_{i-k+1}^n,\cdots,v_{i+k-1}^n\right)\right]$$

where $c = \Delta t/\Delta x$ with Δx and Δt being the space and time discretization steps. Functions H and h are related to $2k+1$ and $2k$ variables, respectively. For convenience, these variables will be named as x_i starting from $i = 0$. Therefore, the conservative form is rewritten as

$$H(x_0, x_1, \cdots, x_{2k}) = x_k - c\left[h(x_1, x_2, \cdots, x_{2k}) - h(x_0, x_1, \cdots, x_{2k-1})\right]$$

Before continuing with the TVD schemes analysis and their relation with monotone schemes, a useful identity of H is presented. Differentiation of the above equation w.r.t. x_0 results in

$$\frac{\partial H}{\partial x_0} = c\frac{\partial h}{\partial x_0}$$

Similarly, by differentiation w.r.t. x_1 leads to

$$\frac{\partial H}{\partial x_1} = -c\left[\frac{\partial h(x_1, x_2, \cdots, x_{2k})}{\partial x_1} - \frac{\partial h(x_0, x_1, \cdots, x_{2k-1})}{\partial x_1}\right]$$

The first term on the r.h.s. corresponds to the partial differentiation of h w.r.t. its first variable and is, therefore, equivalent to $\partial h/\partial x_0$. Thus

$$\frac{\partial H}{\partial x_1} = -c\left(\frac{\partial h}{\partial x_0} - \frac{\partial h}{\partial x_1}\right)$$

where both h functions on the r.h.s. are considered as $h = h(x_0, x_1, \cdots, x_{2k-1})$. By repeating the same procedure for $i = 2, \cdots, 2k$ one gets

$$\left\{\begin{array}{lcl}
\frac{\partial H}{\partial x_0} & = & c\frac{\partial h}{\partial x_0} \\
\frac{\partial H}{\partial x_1} & = & -c\frac{\partial h}{\partial x_0} + c\frac{\partial h}{\partial x_1} \\
\frac{\partial H}{\partial x_2} & = & -c\frac{\partial h}{\partial x_1} + c\frac{\partial h}{\partial x_2} \\
 & \vdots & \\
\frac{\partial H}{\partial x_k} & = & 1 - c\frac{\partial h}{\partial x_{k-1}} + c\frac{\partial h}{\partial x_k} \\
 & \vdots & \\
\frac{\partial H}{\partial x_{2k-1}} & = & -c\frac{\partial h}{\partial x_{2k-2}} + c\frac{\partial h}{\partial x_{2k-1}} \\
\frac{\partial H}{\partial x_{2k}} & = & -c\frac{\partial h}{\partial x_{2k-1}}
\end{array}\right.$$

The summation of the above system of equations leads to the important identity

$$\sum_{i=0}^{2k} \frac{\partial H}{\partial x_k} = 1 \tag{G.1}$$

Subsequently, a study on the TVD schemes is presented. A scheme is called TVD when the solution's total variation is not increased in time. The total variation at time step n is defined as

$$TV(v^n) = \sum_{i=-\infty}^{+\infty} \left| v_{i+1}^n - v_i^n \right|$$

For the above summation to be infinite, one assumes that an integer I exists, such that v_i^n remains constant for every $i \geqslant I$. Discretization scheme H is said to be TVD if

$$TV(v^{n+1}) \leqslant TV(v^n), \quad \forall n$$

Thereafter, it will be proved that every monotone scheme is also a TVD scheme. The proof initially assumes that H is monotone. Then, by using TV definition

$$TV(v^{n+1}) = \sum_i \left| v_{i+1}^{n+1} - v_i^{n+1} \right| = \sum_i \left| H\left(v_{i+1-k}^n, \cdots, v_{i+1+k}^n\right) - H\left(v_{i-k}^n, \cdots, v_{i+k}^n\right) \right|$$

where the summation limits have been neglected. A new set of functions is defined as

$$\xi_{i+1}^n(\theta) = \theta v_{i+1}^n + (1-\theta) v_i^n$$

so as $\xi_{i+1}^n(0) = u_i^n$ and $\xi_{i+1}^n(1) = u_{i+1}^n$. Therefore TV is rewritten as

$$\begin{aligned} TV(v^{n+1}) = & \sum_i \left| \left[H\left(\xi_{i+1-k}^n, \cdots, \xi_{i+1+k}^n\right) \right]_0^1 \right| = \\ & \sum_i \left| \int_0^1 \frac{\partial H\left(\xi_{i+1-k}^n, \cdots, \xi_{i+1+k}^n\right)}{\partial \theta} d\theta \right| = \\ & \sum_i \left| \int_0^1 \left(\frac{\partial H}{\partial \xi_{i+1-k}^n} \frac{\partial \xi_{i+1-k}^n}{\partial \theta} + \cdots + \frac{\partial H}{\partial \xi_{i+1+k}^n} \frac{\partial \xi_{i+1+k}^n}{\partial \theta} \right) d\theta \right| \end{aligned}$$

Term $\frac{\partial H}{\partial \xi_{i+1-k}^n}$ represents the partial derivative of H w.r.t. its first variable and for simplicity, considering that H can be expressed as $H(x_0, \cdots, x_{2k})$, it is represented by $\frac{\partial H}{\partial x_0}$. Moreover, considering that $\frac{\partial \xi_i^n}{\partial \theta} = v_i^n - v_{i-1}^n$, TV becomes

$$TV(v^{n+1}) = \sum_i \left| \int_0^1 \left[\frac{\partial H}{\partial x_0}(v_{i+1-k}^n - v_{i+1-k-1}^n) + \cdots + \frac{\partial H}{\partial x_{2k}}(v_{i+1+k}^n - v_{i+1+k-1}^n) \right] d\theta \right| =$$
$$\sum_i \left| \int_0^1 \sum_{l=0}^{2k} \frac{\partial H}{\partial x_l}(v_{i-k+l+1}^n - v_{i-k+l}^n) d\theta \right| \leqslant$$
$$\sum_i \sum_{l=0}^{2k} \int_0^1 \left| \frac{\partial H}{\partial x_l} \right| \left| v_{i-k+l+1}^n - v_{i-k+l}^n \right| d\theta$$

Since H is monotone, it is true that $\frac{\partial H}{\partial x_l} \geqslant 0$, which simplifies the above inequality. Moreover, for the sake of clarity, a vertical bar is added on the left of H derivative, which indicates the index of the function's middle (k^{th}) variable,

$$TV(v^{n+1}) \leqslant \sum_i \sum_{l=0}^{2k} \int_0^1 \left. \frac{\partial H}{\partial x_l} \right|_i \left| v_{i-k+l+1}^n - v_{i-k+l}^n \right| d\theta$$

Terms of the above double summation are rearranged by setting $m = i - k + l$,

$$TV(v^{n+1}) \leqslant \sum_m \sum_{l=0}^{2k} \int_0^1 \left. \frac{\partial H}{\partial x_l} \right|_{m+k-l} \left| v_{m+1}^n - v_m^n \right| d\theta =$$
$$\sum_m \left| v_{m+1}^n - v_m^n \right| \int_0^1 \sum_{l=0}^{2k} \left. \frac{\partial H}{\partial x_l} \right|_{m+k-l} d\theta$$

By using the already proved eq. G.1 the inequality becomes

$$TV(v^{n+1}) \leqslant \sum_m \left| v_{m+1}^n - v_m^n \right| \int_0^1 d\theta =$$
$$\sum_m \left| v_{m+1}^n - v_m^n \right| = TV(v^n)$$

Consequently, $TV(v^{n+1}) \leqslant TV(v^n)$, which signifies that every monotone scheme is also a TVD scheme.

Appendix H

The Barth-Jespersen Limiter

The MUSCL approach, used for the convection term discretization, allows for second-order accurate simulations. However, according to Godunov's theorem, high-order methods produce spurious oscillations, especially across discontinuities. Therefore, a slope limiter is used to suppress these oscillations while keeping second-order accurate reconstruction in smooth regions of flow fields. The commonly used Barth-Jespersen limiter modifies the piecewise linear distribution at each control volume removing local extrema and ensuring stability. This Appendix aims to prove that the value extrapolated from the cell center to one of its faces, computed by using the Barth-Jespersen limiter, does not exceed the maximum or minimum neighboring flow variables. The proof starts by considering that the extrapolation does not create a new local extremum and concludes with the Barth-Jespersen limiter formulation.

The case of an arbitrary neighborhood of an unstructured 3D mesh is considered. The cell in which the extrapolation takes place is called P, and Q_n indicates its n^{th} neighbor. A second-order extrapolation is achieved as follows,

$$\hat{u} = u^P + \phi \left.\frac{\partial u}{\partial x_i}\right|_P \Delta x_i, \quad i = 1, 3$$

where u^P is any flow variable stored at the cell's barycenter, $\hat{u}$ stands for the extrapolated value, and $\Delta\vec{x}_i$ is the distance vector between the cell's and face's barycenters. The limiter function is represented by ϕ, the codomain of which is strictly set $[0, 1]$.

The extrapolated value computed without the limiter use is defined as

$$\tilde{u} = u^P + \left.\frac{\partial u}{\partial x_i}\right|_P \Delta x_i, \quad i = 1,3$$

By combining the $\hat{u}$ and $\tilde{u}$ definitions, one concludes to the very useful relation

$$\hat{u} = u^P + \phi\left(\tilde{u} - u^P\right)$$

The maximum and minimum flow variables in the neighborhood are

$$\begin{aligned} u_{min} &= min\left(u^P, u^{Q_1}, u^{Q_2}, \cdots, u^{Q_N}\right) \\ u_{max} &= max\left(u^P, u^{Q_1}, u^{Q_2}, \cdots, u^{Q_N}\right) \end{aligned}$$

where N is the number of neighbors. As already explained, $\hat{u}$ should not exceed u_{max} and u_{min}. Thus,

$$\begin{aligned} & u_{min} \leqslant \hat{u} \leqslant u_{min} \Leftrightarrow u_{min} \leqslant u^P + \phi\left(\tilde{u} - u^P\right) \leqslant u_{max} \Leftrightarrow \\ & u_{min} - u^P \leqslant \phi\left(\tilde{u} - u^P\right) \leqslant u_{max} - u^P \end{aligned} \tag{H.1}$$

Three distinctive cases are considered below.

Case 1: $\tilde{u}$=u^P

This case is possible only if $\left.\frac{\partial u}{\partial x_i}\right|_P \Delta x_i = 0$ implying that $\hat{u} = u^P$. Therefore, the limiter's use is trivial and its value is defined as

$$\phi = 1 \tag{H.2}$$

Case 2: $\tilde{u} > u^P$

By dividing ineq. H.1 with $\left(\tilde{u} - u^P\right)$ one gets

$$\frac{u_{min} - u^P}{\tilde{u} - u^P} \leqslant \phi \leqslant \frac{u_{max} - u^P}{\tilde{u} - u^P} \tag{H.3}$$

The first inequality is always satisfied because

$$u_{min} \leqslant u^P \Leftrightarrow \frac{u_{min} - u^P}{\tilde{u} - u^P} \leqslant 0 \leqslant \phi \tag{H.4}$$

The last inequality of ineq. H.4 is true due to the requirement $\phi \in [0, 1]$. The second inequality in ineq. H.3 is satisfied by setting

$$\phi = min(1, \frac{u_{max} - u^P}{\tilde{u} - u^P}) \tag{H.5}$$

which evidently satisfies the $\phi \leqslant 1$ condition. Then, in case $\phi \neq 1$, requirement $\phi \geqslant 0$ is automatically satisfied because

$$u_{max} \geqslant u^P \Leftrightarrow \frac{u_{max} - u^P}{\tilde{u} - u^P} \geqslant 0 \Leftrightarrow \phi \geqslant 0$$

Case 3: $\tilde{u} < u^P$

By dividing ineq. H.1 with $\left(\tilde{u} - u^P\right)$, one gets

$$\frac{u_{min} - u^P}{\tilde{u} - u^P} \geqslant \phi \geqslant \frac{u_{max} - u^P}{\tilde{u} - u^P} \tag{H.6}$$

The second inequality is always satisfied because

$$u_{max} \geqslant u^P \Leftrightarrow \frac{u_{max} - u^P}{\tilde{u} - u^P} \leqslant 0 \leqslant \phi$$

The first inequality in ineq. H.6 is satisfied by setting

$$\phi = min(1, \frac{u_{min} - u^P}{\tilde{u} - u^P}) \tag{H.7}$$

which evidently satisfies the $\phi \leqslant 1$ requirement. From the above definition, in case $\phi \neq 1$, inequality $\phi \geqslant 0$ is automatically satisfied because

$$u_{min} \leqslant u^P \Leftrightarrow \frac{u_{min} - u^P}{\tilde{u} - u^P} \geqslant 0 \Leftrightarrow \phi \geqslant 0$$

Consequently, definitions H.2, H.5 and H.7 presented in the three above cases are combined in the following expression, which is equivalent to the Barth-Jespersen limiter presented in eq. 3.10,

$$\phi = \begin{cases} 1, & \tilde{u} = u^P \\ min(1, \frac{u_{max} - u^P}{\tilde{u} - u^P}), & \tilde{u} > u^P \\ min(1, \frac{u_{min} - u^P}{\tilde{u} - u^P}), & \tilde{u} < u^P \end{cases} \tag{H.8}$$

Appendix I

Orthogonal Correction Expression

The purpose of this Appendix is to give more details on the mathematical formulation of the orthogonal correction used for the computation of $\left.\frac{\partial \Phi}{\partial x_i}\right|_f$ for any variable Φ. Fig. I.1 shows the general case of two cell centroids P and Q connected with a blue line. The red line depicts the face separating the two cells. Vector $\vec{n}$ is perpendicular to the face, $\vec{t}$ is parallel to the face and $\vec{\alpha}$ is parallel to (PQ) line. All three vectors are unitary. The method's goal is to express the derivative on the face as a function of the already known Φ and its derivatives in both P and Q.

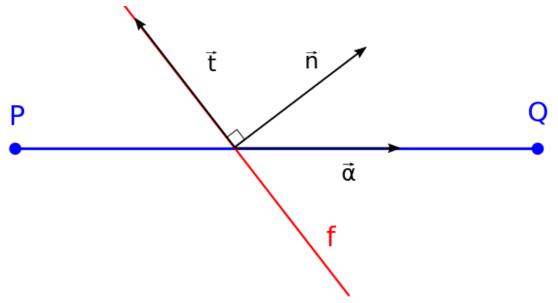


Figure I.1: Finite volumes P and Q are separated by their common edge, shown as a red line. Their centroids are connected with the blue line.

The gradient computation is based on its expression in the $(\vec{\alpha}, \vec{t})$ coordinate system,

$$\left.\frac{\partial \Phi}{\partial x_i}\right|_f = \left.\frac{\partial \Phi}{\partial \alpha}\right|_f \alpha_i + \left.\frac{\partial \Phi}{\partial t}\right|_f t_i \tag{I.1}$$

Its tangent component is computed by linear interpolation,

$$\left.\frac{\partial \Phi}{\partial t}\right|_f t_i = \left[\left.\frac{\partial \Phi}{\partial t}\right|_P w + \left.\frac{\partial \Phi}{\partial t}\right|_Q (1-w)\right] t_i$$

where $w = (fQ)/(PQ)$. Segment (fQ) is the distance between the face centroid and Q, while (PQ) is the distance between and P and Q. Similarly, it is true that

$$\left.\frac{\partial \Phi}{\partial t}\right|_{P,Q} t_i = \left.\frac{\partial \Phi}{\partial x_i}\right|_{P,Q} - \left.\frac{\partial \Phi}{\partial \alpha}\right|_{P,Q} \alpha_i$$

By combining the two above-mentioned equations one gets

$$\left.\frac{\partial \Phi}{\partial t}\right|_f t_i = \left(\left.\frac{\partial \Phi}{\partial x_i}\right|_P - \left.\frac{\partial \Phi}{\partial \alpha}\right|_P \alpha_i\right) w + \left(\left.\frac{\partial \Phi}{\partial x_i}\right|_Q - \left.\frac{\partial \Phi}{\partial \alpha}\right|_Q \alpha_i\right)(1-w)$$

By using the identity

$$\left.\frac{\partial \Phi}{\partial \alpha}\right|_{P,Q} = \left.\frac{\partial \Phi}{\partial x_i}\right|_{P,Q} \alpha_i$$

and rearranging the terms, the tangential derivative becomes

$$\left.\frac{\partial \Phi}{\partial t}\right|_f t_i = \overline{\frac{\partial \Phi}{\partial x_i}} - \overline{\frac{\partial \Phi}{\partial x_j}} \alpha_j \alpha_i \tag{I.2}$$

where

$$\overline{\frac{\partial \Phi}{\partial x_i}} = \left.\frac{\partial \Phi}{\partial x_i}\right|_P w + \left.\frac{\partial \Phi}{\partial x_i}\right|_Q (1-w) \tag{I.3}$$

The gradient component parallel to (PQ) is approximated by central finite differences,

$$\left.\frac{\partial \Phi}{\partial \alpha}\right|_f = \frac{\Phi^Q - \Phi^P}{(PQ)} \tag{I.4}$$

Substitution of eq. I.2 into eq. I.1 gives

$$\left.\frac{\partial \Phi}{\partial x_i}\right|_f = \overline{\frac{\partial \Phi}{\partial x_i}} - \left(\overline{\frac{\partial \Phi}{\partial x_j}} \alpha_j - \left.\frac{\partial \Phi}{\partial \alpha}\right|_f\right) \alpha_i \tag{I.5}$$

Consequently eq. I.5 expresses the gradient on the face as a function of already computed quantities from eqs. I.3 and I.4.

Appendix J

Boundary Conditions Differentiation

The adjoint boundary conditions for compressible and incompressible flows, given in eqs. 6.13 6.27, require the computation of the Jacobian matrices $\frac{\partial \vec{U}}{\partial \vec{Q}}$ and $\frac{\partial \vec{V}}{\partial \vec{Q}}$, respectively. Their computation depends on the boundary conditions imposed on the flow problem, which affects the $\vec{Q}$ choice. This appendix presents some Jacobian matrices corresponding to inlet or outlet boundary conditions for both compressible or incompressible flows.

For compressible inlet boundary conditions, where $\vec{Q} = (p_t, T_t, |\vec{v}|, \alpha_{pitch}, \alpha_{yaw})$, the Jacobian matrix is expressed as

$$\frac{\partial \vec{U}}{\partial \vec{Q}} = \frac{\partial \vec{U}}{\partial \vec{V}} \frac{\partial \vec{V}}{\partial \vec{Q}}$$

where

$$\frac{\partial \vec{U}}{\partial \vec{V}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ v_1 & \rho & 0 & 0 & 0 \\ v_2 & 0 & \rho & 0 & 0 \\ v_3 & 0 & 0 & \rho & 0 \\ \frac{1}{2}|\vec{v}|^2 & \rho v_1 & \rho v_2 & \rho v_3 & \frac{1}{\gamma-1} \end{bmatrix}, \tag{J.1}$$

$$\frac{\partial \vec{V}}{\partial \vec{Q}} = \begin{bmatrix} \rho/p_t & (r-1)\rho/T_t & 2r\rho/|\vec{v}| & 0 & 0 \\ 0 & 0 & v_1/|\vec{v}| & v_3 cos(\alpha_{yaw}) & -v_2 \\ 0 & 0 & v_2/|\vec{v}| & v_3 sin(\alpha_{yaw}) & v_1 \\ 0 & 0 & v_3/|\vec{v}| & -|\vec{v}| sin(\alpha_{pitch}) & 0 \\ p/p_t & \gamma r p/T_t & 2\gamma r p/|\vec{v}| & 0 & 0 \end{bmatrix}$$

and

$$r = \frac{1}{\gamma - 1}\left(\frac{T_t}{T} - 1\right)$$

The angles α_{pitch} and α_{yaw} are defined in section 3.1. According to the same section, an alternative for $\vec{Q}$ in compressible flows is $\vec{Q} = (s, v_1, v_2, v_3, p)$. Then,

$$\frac{\partial \vec{U}}{\partial \vec{Q}} = \begin{bmatrix} -\frac{1}{\gamma}\frac{\rho}{s} & 0 & 0 & 0 & \frac{1}{\gamma}\frac{\rho}{p} \\ -\frac{1}{\gamma}\frac{\rho v_1}{s} & \rho & 0 & 0 & \frac{1}{\gamma}\frac{\rho v_1}{p} \\ -\frac{1}{\gamma}\frac{\rho v_2}{s} & 0 & \rho & 0 & \frac{1}{\gamma}\frac{\rho v_2}{p} \\ -\frac{1}{\gamma}\frac{\rho v_3}{s} & 0 & 0 & \rho & \frac{1}{\gamma}\frac{\rho v_3}{p} \\ -\frac{1}{2\gamma}\frac{\rho}{s}|\vec{v}|^2 & \rho v_1 & \rho v_2 & \rho v_3 & \frac{1}{\gamma-1} + \frac{1}{2\gamma}\frac{\rho}{p}|\vec{v}|^2 \end{bmatrix}$$

At the outlet, constant pressure is usually imposed, implying $\vec{Q} = \vec{V}$, and therefore, $\frac{\partial \vec{U}}{\partial \vec{Q}} = \frac{\partial \vec{U}}{\partial \vec{V}}$ which is given in eq. J.1.

On the other hand, in incompressible flows, $\vec{Q} = (p_t, |\vec{v}|, a_{pitch}, a_{yaw})$ at the inlet, and the required Jacobian matrix is

$$\frac{\partial \vec{V}}{\partial \vec{Q}} = \begin{bmatrix} 1 & -|\vec{v}| & 0 & 0 \\ 0 & v_1/|\vec{v}| & v_3 cos(\alpha_{yaw}) & -v_2 \\ 0 & v_2/|\vec{v}| & v_3 sin(\alpha_{yaw}) & v_1 \\ 0 & v_3/|\vec{v}| & -|\vec{v}| sin(\alpha_{pitch}) & 0 \end{bmatrix}$$

At the outlet $\vec{Q} = \vec{V}$ and the required Jacobian $\frac{\partial \vec{V}}{\partial \vec{Q}}$ is the identity matrix.

Appendix K

The Continuous Adjoint Method Implemented in Cases with Recirculation at the Outlet

Optimization in problems concerning internal flows is extensively common in real-life applications. Turbomachines, pumps, and heat exchangers are some examples illustrating the great variety and importance of such cases. Their flow simulation can be challenging enough, especially when recirculation appears at the outlet, harming the convergence of the governing equations. This phenomenon escalates during the optimization process, where irregular geometries may appear. The corresponding adjoint equations are also sensitive when recirculation occurs in the flow field, especially when constant static pressure is imposed at the outlet. If eq. 6.13 is also implemented as the adjoint outlet boundary condition, the adjoint problem becomes ill-conditioned, and convergence is impossible. This section presents the mathematical development which proves the previous statement for compressible and incompressible flows.

Without loss of generality, the coordinate system is defined with the x_1 axis perpendicular to the outlet boundary's plane. The unit vector, normal to the plane, is $\vec{n} = (1, 0, 0)$. In the case of recirculation, there is at least one point on the outlet plane having zero normal velocity (e.g., $v_1 = 0$). Subsequently, the compressible adjoint boundary condition at that point is examined.

Firstly, the l.h.s. of the outlet adjoint condition

$$\Psi_i A_{ijk} n_k \frac{\partial U_j}{\partial Q_h} = \frac{\partial F_{S_k}}{\partial Q_h} n_k$$

is expressed in a more convenient form by using the definition of eq. 3.6,

$$\Psi_i A_{ijk} n_k \frac{\partial U_j}{\partial Q_h} = \Psi_i \frac{\partial f_{ik}}{\partial U_j} n_k \frac{\partial U_j}{\partial V_l} \frac{\partial V_l}{\partial Q_h} = \Psi_i \frac{\partial f_{ik}}{\partial V_j} n_k \frac{\partial V_j}{\partial Q_h} = \Psi_i \frac{\partial f_{i1}}{\partial V_j} \frac{\partial V_j}{\partial Q_h}$$

The Jacobian matrix $\frac{\partial f_{i1}}{\partial V_j}$ for $v_1 = 0$ reads

$$\frac{\partial \vec{f_1}}{\partial \vec{V}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & \rho v_2 & 0 & 0 & 0 \\ 0 & \rho v_3 & 0 & 0 & 0 \\ 0 & \rho h_t & 0 & 0 & 0 \end{bmatrix}$$

Then,

$$\frac{\partial \vec{f_1}}{\partial \vec{V}} \frac{\partial \vec{V}}{\partial \vec{Q}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ \frac{\partial V_5}{\partial Q_1} & \frac{\partial V_5}{\partial Q_2} & \frac{\partial V_5}{\partial Q_3} & \frac{\partial V_5}{\partial Q_4} & \frac{\partial V_5}{\partial Q_5} \\ \rho v_2 \frac{\partial V_2}{\partial Q_1} & \rho v_2 \frac{\partial V_2}{\partial Q_2} & \rho v_2 \frac{\partial V_2}{\partial Q_3} & \rho v_2 \frac{\partial V_2}{\partial Q_4} & \rho v_2 \frac{\partial V_2}{\partial Q_5} \\ \rho v_3 \frac{\partial V_2}{\partial Q_1} & \rho v_3 \frac{\partial V_2}{\partial Q_2} & \rho v_3 \frac{\partial V_2}{\partial Q_3} & \rho v_3 \frac{\partial V_2}{\partial Q_4} & \rho v_3 \frac{\partial V_2}{\partial Q_5} \\ \rho h_t \frac{\partial V_2}{\partial Q_1} & \rho h_t \frac{\partial V_2}{\partial Q_2} & \rho h_t \frac{\partial V_2}{\partial Q_3} & \rho h_t \frac{\partial V_2}{\partial Q_4} & \rho h_t \frac{\partial V_2}{\partial Q_5} \end{bmatrix}$$

Vector $\vec{Q}$ represents any set of 5 independent flow variables. However, one of them must be the imposed pressure at the outlet. Suppose $Q_k = p$. As explained in subsection 6.1.8, the k^{th} equation of the above system is excluded. Moreover, in the present special case, the first column of the systems's l.h.s. $\left(\frac{\partial \vec{f_1}}{\partial \vec{V}} \frac{\partial \vec{V}}{\partial \vec{Q}}\right)^T$ is zero, eliminating Ψ_1. The rest variables can be computed by solving a 4×4 system.

According to subsection 3.1, $V_5 = p$. Therefore, $\frac{\partial V_5}{\partial Q_i}$ is zero for $i \neq k$. Consequently, every segment of the second column of $\left(\frac{\partial \vec{f_1}}{\partial \vec{V}} \frac{\partial \vec{V}}{\partial \vec{Q}}\right)^T$ is zero except $\frac{\partial V_5}{\partial Q_k}$, which, however, belongs to the deleted k^{th} line. Therefore, Ψ_2 is also eliminated and the 4×4 system becomes

$$\rho v_2 \frac{\partial V_2}{\partial Q_h} \Psi_3 + \rho v_3 \frac{\partial V_2}{\partial Q_h} \Psi_4 + \rho h_t \frac{\partial V_2}{\partial Q_h} \Psi_5 = \frac{\partial F_{S_1}}{\partial Q_h}, \quad \forall\, h \neq k,\ h = 1, \cdots, 5$$

In case there are two integers $m, n \neq k$ such that $\frac{\partial F_{S_1}}{\partial Q_m} \neq \frac{\partial F_{S_1}}{\partial Q_n}$, the system is inconsistent, and the boundary conditions cannot be imposed. Otherwise, the system downgrades to a simple equation with an infinite number of solutions. In both cases, the adjoint problem is not well defined regardless of the objective's expression or the choice of $\vec{Q}$.

A similar mathematical development implemented in incompressible flows leads to

$$\beta^2 \frac{\partial V_2}{\partial Q_h} \Psi_1 + v_2 \frac{\partial V_2}{\partial Q_h} \Psi_3 + v_3 \frac{\partial V_2}{\partial Q_h} \Psi_4 = \frac{\partial F_{S_1}}{\partial Q_h}, \quad \forall\ h \neq k,\ h = 1, \cdots, 4$$

proving that the adjoint problem is ill-conditioned also in incompressible cases. Consequently, recirculation at the flow field's exit must be always avoided (e.g., by transferring the outlet boundaries away from the solid bodies) when the continuous adjoint method is used in the optimization process.

Appendix L

The adjoint HLLC and FVS schemes

The following sections introduce two different discretization schemes for the 1D adjoint equations inspired by the HLLC [307] and FVS [291] schemes. They aim to approximate the adjoint flux expression defined in eqs. 6.33 and 6.34. Moreover, the final section compares the convergence of the four discretization schemes that appear in this thesis. The mathematical development is based on relations and notations defined in section 6.3.

L.1 The Adjoint HLLC Scheme

The HLLC scheme is studied in the specific case where $N = 3$. According to this method, the (x,t) domain is split into four regions separated by the $x = S^{(L,R,*)}t$ lines, where S^L and S^R are estimated as

$$S^L = -max\{\lambda_1^L, \lambda_2^L, \lambda_3^L\}$$
$$S^R = -min\{\lambda_1^R, \lambda_2^R, \lambda_3^R\}$$

and λ_i are the eigenvalues of the Jacobian matrices A^L and A^R, respectively [76]. Alternatively, the eigenvalues of $\tilde{A}$ can be used instead [76], [84]. Eigenvalue S^* is given by

$$S^* = \frac{\Phi_2^R - \Phi_2^L}{\Phi_1^R - \Phi_1^L}$$

where

$$\Phi_i^L = A_{ij}^L U_j^L - S^L U_i^L$$
$$\Phi_i^R = A_{ij}^R U_j^R - S^R U_i^R$$

It is always true that $S^L < S^* < S^R$. Adjoint variables at each region are uniform and equal to Ψ^L, Ψ^{L0}, Ψ^{R0} and Ψ^R. An example of the (x,t) domain is shown in fig. L.1. The $\vec{f}^0$ flux discretization depends on the S^* sign.

$$\vec{f}^0 = \begin{cases} \vec{f}^{L0}, & S^* > 0 \\ \vec{f}^{R0}, & S^* < 0 \end{cases}$$

In case $S^* > 0$, the unknown integral of eq. 6.33 is required which is

$$\int_{S^L T}^{0} \Psi_i(x,T)dx = \begin{cases} -\Psi_i^L S^L T, & S^L > 0 \\ -\Psi_i^{L0} S^L T, & S^L < 0 \end{cases}$$

and the adjoint flux becomes

$$f_i^{L0} = \begin{cases} -A_{ji}^L \Psi_j^L, & S^L > 0 \\ -A_{ji}^L \Psi_j^L + S^L \left(\Psi_i^{L0} - \Psi_i^L\right), & S^L < 0 \end{cases}$$

Three characteristic lines transfer the adjoint Riemann invariants ($\vec{W}_L$ or $\vec{W}_R$) from the field initialization at $t=0$ to points along the t-axis determining the vector $\vec{\Psi}^{L0}$. By following the same technique shown in subsection 6.3.1 for the diagonalization of the system

$$\frac{\partial \Psi_i}{\partial t} - A_{ji}^L \frac{\partial \Psi_j}{\partial x} = 0$$

an expression similar to eq.6.40 can be obtained,

$$\Psi_i^{L0} = \Psi_i^L + \sum_{m=\hat{m}+1}^{N} P_{mi}^{L,-1}(W_m^R - W_m^L) = \Psi_i^L + P_{3i}^{L,-1}(W_3^R - W_3^L)$$

Then, the flux becomes

$$f_i^{L0} = -A_{ji}^L \Psi_j^L + S^L P_{3i}^{L,-1}(W_3^R - W_3^L)$$

A similar procedure followed for $S^* < 0$ leads to

$$f_i^{R0} = \begin{cases} -A_{ji}^R \Psi_j^R - S^R P_{1i}^{R,-1}(W_1^R - W_1^L), & S^R > 0 \\ -A_{ji}^R \Psi_j^R, & S^R < 0 \end{cases}$$

Consequently,

$$f_i^0 = \begin{cases} -A_{ji}^L \Psi_j^L, & S^* > 0, S^L > 0 \\ -A_{ji}^L \Psi_j^L + S^L P_{3i}^{L,-1}(W_3^R - W_3^L), & S^* > 0, S^L < 0 \\ -A_{ji}^R \Psi_j^R - S^R P_{1i}^{R,-1}(W_1^R - W_1^L), & S^* < 0, S^R > 0 \\ -A_{ji}^R \Psi_j^R, & S^* < 0, S^R < 0 \end{cases} \tag{L.1}$$

where

$$W_i^L = P_{ji}^L \Psi_j^L$$
$$W_i^R = P_{ji}^R \Psi_j^R$$

Another alternative for the $\vec{f}^0$ approximation could be the replacement of $P^{L,-1}$ and $P^{R,-1}$ with $\tilde{P}^{-1}$. The developed adjoint HLLC scheme avoids the computation of the absolute Jacobian, making it a much faster discretization method compared to eq. 6.42.

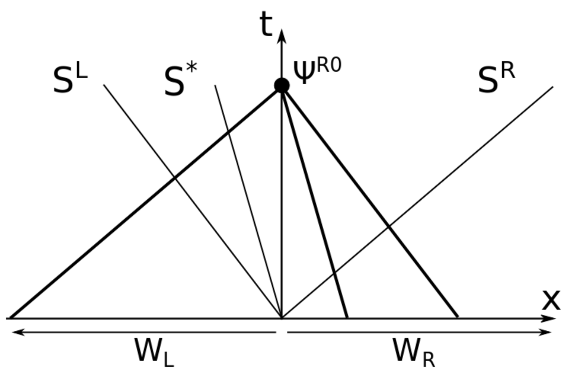


Figure L.1: Three characteristics are intersected at point $(0, t)$, affecting the field $\vec{\Psi}^{R0}$. The left-most characteristic, associated with the first eigenvalue, transfers the invariant W_1^L and the other two transfer W_2^R and W_3^R.

L.2 The Adjoint FVS Scheme

The FVS formula for the adjoint flux can be derived in a straightforward manner by following the procedure used for the flow equations and keeping in mind that the adjoint eigenvalues are the same in magnitude, though opposite in sign with those of the primal problem. However, a different process will be followed, highlighting the close relation between FVS and the already developed adjoint HLLC scheme. Eq. L.1 is used for its generation, simplified by computing all the associated flow quantities as the mean average between the left (L) and right (R) states, marked by an over-bar, and by setting $S^*\!=\!\lambda_2$. The $\bar{\lambda}_i^+$ and $\bar{\lambda}_i^-$ are introduced such that

$$\bar{\lambda}_i^+ = \begin{cases} \bar{\lambda}_i, & \bar{\lambda}_i \geqslant 0 \\ 0, & \bar{\lambda}_i < 0 \end{cases} , \quad \bar{\lambda}_i^- = \begin{cases} 0, & \bar{\lambda}_i > 0 \\ \bar{\lambda}_i, & \bar{\lambda}_i \leqslant 0 \end{cases}$$

Additionally, two matrices are defined,

$$\bar{A}_{ij}^+ = \sum_{m=1}^{3} \bar{P}_{im} \bar{\lambda}_m^+ \bar{P}_{mj}^{-1}, \quad \bar{A}_{ij}^- = \sum_{m=1}^{3} \bar{P}_{im} \bar{\lambda}_m^- \bar{P}_{mj}^{-1}$$

Apparently, $\lambda_i\!=\!\lambda_i^+ + \lambda_i^-$ and $\bar{A}_{ij}\!=\!\bar{A}_{ij}^+ + \bar{A}_{ij}^-$. Subsequently, the four cases presented in eq. L.1 will be re-expressed with the aid of the newly defined quantities. In the first case, all adjoint eigenvalues are positive. Therefore, $\bar{\lambda}_i\!=\!\bar{\lambda}_i^- \ \forall i$, which induces $\bar{A}^+\!=\!0$ and $\bar{A}^-\!=\!\bar{A}$, meaning that $f_i^0\!=\!-\bar{A}_{ji}^- \Psi_j^L - \bar{A}_{ji}^+ \Psi_j^R$. In the second case, $\bar{\lambda}_{1,2} < 0$ and $\bar{\lambda}_3 > 0$. Hence, $\bar{\lambda}_{1,2}^+\!=\!0$, $\bar{\lambda}_3^+\!=\!\bar{\lambda}_3$ and

$$\begin{aligned} f_i^0 &= -\bar{A}_{ji} \Psi_j^L + S^L \bar{P}_{3i}^{-1} (W_3^R - W_3^L) = -\bar{A}_{ji} \Psi_j^L - \bar{\lambda}_3 \bar{P}_{3i}^{-1} (\bar{P}_{j3} \Psi_j^R - \bar{P}_{j3} \Psi_j^L) \\ &= -\bar{A}_{ji}^+ \Psi_j^L - \bar{A}_{ji}^- \Psi_j^L - \sum_{m=1}^{3} \bar{P}_{mi}^{-1} \bar{\lambda}_m^+ (\bar{P}_{jm} \Psi_j^R - \bar{P}_{jm} \Psi_j^L) \\ &= -\bar{A}_{ji}^+ \Psi_j^L - \bar{A}_{ji}^- \Psi_j^L - \bar{A}_{ji}^+ (\Psi_j^R - \Psi_j^L) = -\bar{A}_{ji}^- \Psi_j^L - \bar{A}_{ji}^+ \Psi_j^R \end{aligned}$$

Thirdly, $\bar{\lambda}_1 < 0$ and $\bar{\lambda}_{2,3} > 0$, inducing $\bar{\lambda}_1^-\!=\!\bar{\lambda}_1$, $\bar{\lambda}_{2,3}^-\!=\!0$. Thus,

$$\begin{aligned} f_i^0 &= -\bar{A}_{ji} \Psi_j^R - S^R \bar{P}_{1i}^{-1} (W_1^R - W_1^L) = -\bar{A}_{ji} \Psi_j^R + \bar{\lambda}_1 \bar{P}_{1i}^{-1} (\bar{P}_{j1} \Psi_j^R - \bar{P}_{j1} \Psi_j^L) \\ &= -\bar{A}_{ji}^+ \Psi_j^R - \bar{A}_{ji}^- \Psi_j^R + \sum_{m=1}^{3} \bar{P}_{mi}^{-1} \bar{\lambda}_m^- (\bar{P}_{jm} \Psi_j^R - \bar{P}_{jm} \Psi_j^L) \\ &= -\bar{A}_{ji}^+ \Psi_j^R - \bar{A}_{ji}^- \Psi_j^R + \bar{A}_{ji}^- (\Psi_j^R - \Psi_j^L) = -\bar{A}_{ji}^- \Psi_j^L - \bar{A}_{ji}^+ \Psi_j^R \end{aligned}$$

Lastly, when all adjoint eigenvalues are negative, $\bar{\lambda}_i = \bar{\lambda}_i^+$ meaning that $\bar{A}^+ = \bar{A}$ and $\bar{A}^- = 0$. Then, the flux can be written as $f_i^0 = -\bar{A}_{ji}^- \Psi_j^L - \bar{A}_{ji}^+ \Psi_j^R$.

In all four cases, the final expression of $\vec{f}^0$ is the same, ensuring that

$$f_i^0 = -\bar{A}_{ji}^- \Psi_j^L - \bar{A}_{ji}^+ \Psi_j^R \qquad \text{(L.2)}$$

in any case. Furthermore, considering that

$$\bar{A}_{ij}^+ = \frac{1}{2}\left(\bar{A}_{ij} + |\bar{A}|_{ij}\right), \quad \bar{A}_{ij}^- = \frac{1}{2}\left(\bar{A}_{ij} - |\bar{A}|_{ij}\right)$$

where $|\bar{A}|_{ij} = \sum\limits_{m=1}^{3} \bar{P}_{im} |\bar{\lambda}_m| \bar{P}_{mj}^{-1}$, eq. L.2 becomes

$$f_i^0 = \frac{1}{2}(-\bar{A}_{ji}\Psi_j^L - \bar{A}_{ji}\Psi_j^R) - \frac{1}{2}\sum_{m=1}^{N} \bar{P}_{mi}^{-1} |\bar{\lambda}_m| \bar{P}_{jm} (\Psi_j^R - \Psi_j^L)$$

underlying the close correlation with eq. 6.42. Finally, another alternative for the adjoint FVS scheme could come of by substituting $\bar{A}$ for $\tilde{A}$.

L.3 Comparison of Adjoint Discretization Schemes

The previously described discretization schemes and these studied in section 6.3 are closely related. The convergence rate they provide is tested in a 1D optimization problem governed by the steady compressible inviscid flow equations,

$$\frac{\partial U_i}{\partial t} + \frac{\partial f_i}{\partial x} + S_i = 0, \quad i = 1, \cdots, 3$$

where $x \in [0, l]$, S_i is a source term, which is non-zero only for the continuity equation,

$$S_1 = -10\left[b\xi^4 - 2(b+1)\xi^3 + (b+3)\xi^2\right]$$

and

$$\xi = \begin{cases} 2x/l, & x \leqslant l/2 \\ 2(l-x)/l, & x > l/2 \end{cases}$$

Variable l is the domain's length and $b = 1$ is the only design variable of the optimization problem. Total pressure and temperature are defined at the inlet

($p_t = 1bar, T_t = 300K$) and static pressure at the outlet ($p = 0.98bar$). The objective function is set

$$F = \frac{1}{2}\int_0^l (v - v_t)^2 dx$$

where v is the flow velocity, and v_t is a constant user-defined velocity target set to 60 m/s. A mesh of 300 cells was used for the flow and adjoint PDEs discretization. Fig. L.2 compares the convergence of the field adjoint equations for 4 different discretization schemes. Although the corrected Roe scheme (ROEC) converges in a subsequent pseudo-time iteration compared to the others due to a steep rise of the residual at the beginning of the run, they show the same convergence rate.

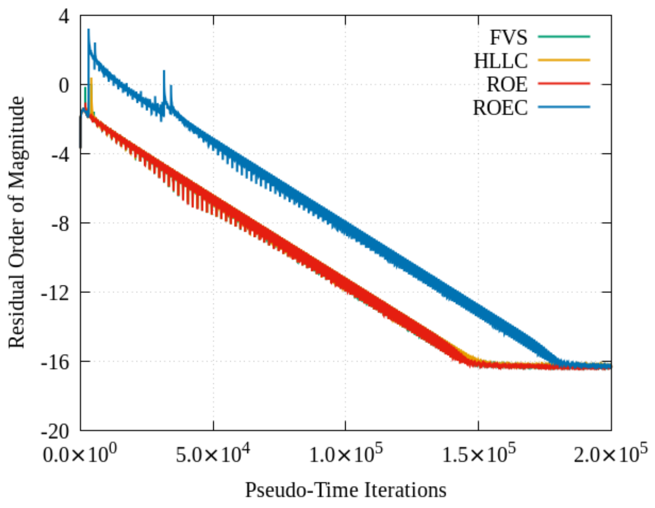


Figure L.2: 1D Inviscid Adjoint Flow: Adjoint velocity convergence history. Comparison between 4 different adjoint discretization schemes. ROEC stands for the corrected Roe scheme explained in subsection 6.3.2 and corresponds to the only curve behaving differently than the others.

Appendix M

Memory Reduction by using the SVD Method

The Singular Value Decomposition (SVD) [112], [113], [323] is used for the unsteady flow field's efficient compression, storage, and reconstruction to be available for the adjoint solver during the backward in time integration of the unsteady adjoint PDEs. The method is better explained by initially assuming that during the adjoint flow simulation, the already computed flow field at each time step is stored, forming a $m \times n$ matrix M, where m is the mesh size and n is the total number of time steps. The SVD method is implemented to reduce the memory requirements by creating a more efficiently stored matrix M' being also the closest approximation of M among all matrices of the same rank. This process is divided into two steps.

Firstly, the SVD is applied to M, according to which each real $m \times n$ matrix can be decomposed as

$$M = USV^T$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices and their columns are called the left and right singular vectors, respectively [32], [294]. Matrix $S \in \mathbb{R}^{m\times n}$ is of the following form

$$S_{ij} = \begin{cases} \sigma_i & i = j \\ 0 & i \neq j \end{cases}$$

where $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_n$ are the singular values of M. Various algorithms have been proposed for the SVD implementation. In this thesis, the Golub-Kahan algorithm [112], [113] is preferred due to its efficiency and stability.

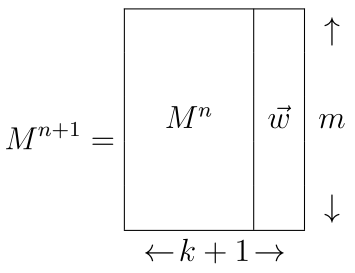
Secondly, S is transformed to the diagonal matrix S' by deleting all but its first k rows forming matrix S'. Similarly, the sizes of U and V are reduced to $m \times k$ and $n \times k$, respectively. Based on the so-truncated matrices $U' \in \mathbb{R}^{m\times k}$, $S' \in \mathbb{R}^{k\times k}$, and $V' \in \mathbb{R}^{n\times k}$, M' becomes

$$M' = U'S'V'^T$$

Considering that in practical applications $k \ll n$, the storage of the $m \times n$ matrix M is avoided by storing U', S', V' instead, reducing the total memory size to $k \times (m + n + 1)$.

Although the aforementioned method can successfully reduce the required memory, M should be available in the first place, and memory reduction is, in fact, unnecessary. Thus, an alternative way should be investigated to compute M' avoiding the storage of M as a whole. This is possible by implementing the so-called incremental SVD (iSVD) technique, which computes U', S', and V' by successive approximations noted as U^n, S^n, and V^n, where n indicates the current time step. Their product is denoted as M^n.

Subsequently, an initial approach to the iSVD is explained. Consider that the available memory capacity is $m \times (k{+}1)$. Once the first k instants of the state field are stored in M^k, the SVD algorithm is applied, and U^k, S^k, and V^k are computed. As the flow solver proceeds in time, U^{n+1}, S^{n+1}, and V^{n+1} are generated based on the corresponding set of matrices computed at the n^{th} time step. Specifically, at time step $n{+}1$ a new solution (w) becomes available, and matrix M^{n+1} is formed as



where M^n is restored as $U^nS^nV^{nT}$. Then, the size of M^{n+1} is reduced to $m \times k$ by applying the SVD algorithm and nullifying the last singular value giving rise to U^{n+1}, S^{n+1}, and V^{n+1}. By repeating the same procedure at each time step, the required memory remains constant and less than its upper limit. However, the computational cost of such a method is very high due to the repeated implementation of the costly SVD algorithm to the large matrix M^{n+1}. A much more efficient method [47], [48], [25] is described below.

Each time the number of columns of M^n is increased by one, an extra column ($\vec{J}$) is added to U^n. Matrix S^n is also extended by the column $\vec{L}$, and an extra element κ is added to its diagonal. Matrix V^n is enlarged by one column, and one row consisted of zero elements except its diagonal, which is set to 1. The new temporary matrices are

$$U^* = \underset{\leftarrow k+1 \rightarrow}{\begin{array}{|c|c|}\hline & \\ U^n & \vec{J} \\ & \\ \hline\end{array}} \begin{array}{c}\uparrow \\ m \\ \downarrow\end{array} \;,\quad S^* = \underset{\leftarrow k+1 \rightarrow}{\begin{array}{|c|c|}\hline S^n & \vec{L} \\ \hline 0 & \kappa \\ \hline\end{array}} \begin{array}{c}\uparrow \\ k+1 \\ \downarrow\end{array} \;,\quad V^* = \underset{\leftarrow k+1 \rightarrow}{\begin{array}{|c|c|}\hline V^n & \vec{0} \\ \hline \vec{0} & 1 \\ \hline\end{array}} \begin{array}{c}\uparrow \\ k+1 \\ \downarrow\end{array}$$

The unknown elements are computed under the requirement $M^{n+1} = U^* S^* V^{*T}$, which implies

$$\vec{w} = U^* S^* [\vec{0} \mid 1]^T = U^* [\vec{L} \mid \kappa]^T = U^n \vec{L} + \kappa \vec{J} \tag{M.1}$$

Moreover, U^* should be orthogonal, meaning that $\vec{J}$ is unitary and perpendicular to any column of U^n,

$$U^{nT} \vec{J} = \vec{0}$$

Vector $\vec{L}$ is easily computed by multiplying eq. M.1 with U^n and considering that U^n is orthogonal ($U^{nT} U^n = I$),

$$U^{nT} \vec{w} = (U^{nT} U) \vec{L} + \kappa (U^{nT} \vec{J}) \Leftrightarrow \vec{L} = U^{nT} \vec{w} \tag{M.2}$$

Then, vector $\vec{J}$ is computed from eq. M.1,

$$\vec{J} = \frac{1}{\kappa} \left(\vec{w} - U^n \vec{L} \right) \tag{M.3}$$

The unknown number κ is computed by considering that $||\vec{J}||_2 = 1$,

$$\kappa = ||\vec{w} - U^n \vec{L}||_2 \tag{M.4}$$

The computation of $\vec{L}$, $\vec{J}$, and κ from eqs. M.2, M.3, and M.4 ensures that matrices U^* and V^* are orthogonal and $M^{n+1} = U^* S^* V^{*T}$. The last step is the transformation of S^* to a diagonal form by applying the SVD algorithm to this relatively small matrix, $S^* = U_s S_s V_s^T$. Finally,

$$M^{n+1} = U^* U_s S_s V_s^T V^{*T} = U^{n+1} S^{n+1} V^{n+1^T}$$

where $U^{n+1} = U^* U_s$ and $V^{n+1} = V^* V_s$ are orthogonal as products of orthogonal matrices and $S^{n+1} = S_s$ is diagonal containing the singular values of M^{n+1}. The process is repeated during the flow equations' time integration, and after the total number of N time steps is completed, U', S', and V' are available as U^N, S^N, and V^N, respectively. During the solution of the adjoint PDEs, the flow field $\vec{w}^n$ at each time step n is restored as

$$w_i^n \simeq \sum_{j=1}^{k} U'_{ij} \sigma'_j V'_{nj}$$

where σ'_j are the singular values stored in the diagonal of S'.

Appendix N

Memory Reduction by using the PGD Method

The Proper Generalized Decomposition (PGD) [60], [9], [170] is able to support the unsteady adjoint algorithm to overcome the storage of the entire unsteady flow field, which becomes prohibitive in large-scale simulations. The first section presents the mathematical development of the method and its incremental counterpart. The second section gives information about the proper adaptation of the PGD to the ghost-cell method and its implementation to the lift maximization of an isolated airfoil. Subsequently, the effect of PGD on the sensitivity derivatives accuracy is studied in section N.2.

N.1 The PGD and Incremental PGD Theory

The method's main idea is to represent a multi-dimensional unsteady field as the sum of 1D function products. For instance, an arbitrary unsteady 2D scalar field u is written as

$$u(x,y,t) \simeq \sum_{\mu=1}^{M} \phi^{\mu}(x)\theta^{\mu}(y)\tau^{\mu}(t) \tag{N.1}$$

Assuming that a small number of modes (M) represents the initial field accurately enough, a noticeable gain in memory usage is expected since scalar modes ϕ^{μ}, θ^{μ}, and τ^{μ} are stored instead of the entire $u(x,y,t)$ field.

Consider that $u(x, y, t)$ is discretized in a structured mesh as $u_{i,j,k}$, where (i, j) enumerate the nodes and k identifies the current time step. The corresponding 1D functions are computed so as to minimize the reconstruction error, defined as

$$E_m = \frac{1}{2}\sum_{k=1}^{K}\sum_{i=1}^{I}\sum_{j=1}^{J}\left[\sum_{\mu=1}^{M}\phi_i^\mu\theta_j^\mu\tau_k^\mu - u_{i,j,k}\right]^2 \tag{N.2}$$

Through its differentiation, the final equations for updating the m^{th} modes emerge

$$\begin{aligned}
\phi_i^m &= \frac{\sum\limits_{j=1}^{J}\sum\limits_{k=1}^{K} u_{i,j,k}\theta_j^m t_k^m - \sum\limits_{\mu=1}^{m-1}\phi_i^\mu\left(\sum\limits_{j=1}^{J}\sum\limits_{k=1}^{K}\theta_j^m\theta_j^\mu\tau_k^m\tau_k^\mu\right)}{\sum\limits_{j=1}^{J}\sum\limits_{k=1}^{K}(\theta_j^m)^2(\tau_k^m)^2}\\
\theta_j^m &= \frac{\sum\limits_{i=1}^{I}\sum\limits_{k=1}^{K} u_{i,j,k}\phi_i^m t_k^m - \sum\limits_{\mu=1}^{m-1}\theta_j^\mu\left(\sum\limits_{i=1}^{I}\sum\limits_{k=1}^{K}\phi_i^m\phi_i^\mu\tau_k^m\tau_k^\mu\right)}{\sum\limits_{i=1}^{I}\sum\limits_{k=1}^{K}(\phi_i^m)^2(\tau_k^m)^2}\\
\tau_k^m &= \frac{\sum\limits_{j=1}^{J}\sum\limits_{i=1}^{I} u_{i,j,k}\phi_i^m\theta_j^m - \sum\limits_{\mu=1}^{m-1}\tau_k^\mu\left(\sum\limits_{j=1}^{J}\sum\limits_{i=1}^{I}\phi_i^m\phi_i^\mu\theta_j^m\theta_j^\mu\right)}{\sum\limits_{j=1}^{J}\sum\limits_{i=1}^{I}(\phi_i^m)^2(\theta_j^m)^2}
\end{aligned} \tag{N.3}$$

However, eq. N.3 requires the whole field time series, which should have been stored beforehand. A new method called incremental PGD (iPGD), firstly presented in [236], overcomes this drawback by redefining the error function as

$$\begin{aligned}
E_m = &\frac{1}{2}\sum_{i=1}^{I}\sum_{j=1}^{J}\left[\sum_{\mu=1}^{M}\phi_i^\mu\theta_j^\mu\tau_{K+1}^\mu - u_{i,j,K+1}\right]^2\\
&+\frac{w}{2}\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\left[\sum_{\mu=1}^{M}\left(\phi_i^\mu\theta_j^\mu\tau_k^\mu - \tilde{\phi}_i^\mu\tilde{\theta}_j^\mu\tilde{\tau}_k^\mu\right)\right]^2
\end{aligned} \tag{N.4}$$

where the first term on the r.h.s. corresponds to the approximation error at the current time step, whereas the second one to the overall error for all the previous time steps, which have already been processed through the iPGD yielding modes $\tilde{\phi}_i^\mu$, $\tilde{\theta}_j^\mu$, $\tilde{\tau}_k^\mu$. The contribution to the error is weighted by w, which is user-defined. At each time step, modes $(\phi_i^m, \theta_j^m, \tau_k^m)$ are updated, and new values τ_{K+1}^m are appended.

The unknown quantities are calculated by setting the derivatives of E_m against zero, getting

$$\begin{aligned}
\phi_i^m &= Q_{1x}^i / Q_{2x}^i, \qquad i = 1, \cdots, I \\
\theta_j^m &= Q_{1y}^j / Q_{2y}^j, \qquad j = 1, \cdots, J \\
\tau_k^m &= Q_{1t}^k / Q_{2t}^k, \qquad k = 1, \cdots, K \\
\tau_{K+1}^m &= Q_{1T}^{K+1} / Q_{2t}^{K+1}
\end{aligned} \tag{N.5}$$

where

$$\begin{aligned}
Q_{1x}^i &= \tau_{K+1}^m \sum_{j=1}^{J} \theta_j^m \, u_{i,j,K+1} - \tau_{K+1}^m \sum_{\mu=1}^{m-1} \left[\left(\sum_{j=1}^{J} (\theta_j^\mu \theta_j^m) \right) \phi_i^\mu \tau_{K+1}^\mu \right] \\
&\quad + w \tilde{\phi}_i^m \sum_{k=1}^{K} \sum_{j=1}^{J} \tilde{\theta}_j^m \theta_j^m \tilde{\tau}_k^m \tau_k^m - w \sum_{k=1}^{K} \sum_{j=1}^{J} \left[\sum_{\mu=1}^{m-1} \left(\phi_i^\mu \theta_j^\mu \tau_k^\mu - \tilde{\phi}_i^\mu \tilde{\theta}_j^\mu \tilde{\tau}_k^\mu \right) \theta_j^m \tau_k^m \right] \\
Q_{2x}^i &= (\tau_{K+1}^m)^2 \sum_{j=1}^{J} (\theta_j^m)^2 + w \sum_{k=1}^{K} \sum_{j=1}^{J} (\theta_j^m)^2 (\tau_k^m)^2
\end{aligned}$$

$$\begin{aligned}
Q_{1y}^j &= \tau_{K+1}^m \sum_{i=1}^{I} \phi_i^m \, u_{i,j,K+1} - \tau_{K+1}^m \sum_{\mu=1}^{m-1} \left[\left(\sum_{i=1}^{I} (\phi_i^\mu \phi_i^m) \right) \theta_j^\mu \tau_{K+1}^\mu \right] \\
&\quad + w \tilde{\theta}_j^m \sum_{k=1}^{K} \sum_{i=1}^{I} \tilde{\phi}_i^m \phi_i^m \tilde{\tau}_k^m \tau_k^m - w \sum_{k=1}^{K} \sum_{i=1}^{I} \left[\sum_{\mu=1}^{m-1} \left(\phi_i^\mu \theta_j^\mu \tau_k^\mu - \tilde{\phi}_i^\mu \tilde{\theta}_j^\mu \tilde{\tau}_k^\mu \right) \phi_i^m \tau_k^m \right] \\
Q_{2y}^j &= (\tau_{K+1}^m)^2 \sum_{i=1}^{I} (\phi_i^m)^2 + w \sum_{k=1}^{K} \sum_{i=1}^{I} (\phi_i^m)^2 \, (\tau_k^m)^2
\end{aligned}$$

$$\begin{aligned}
Q_{1t}^k &= \tilde{\tau}_k^m \sum_{j=1}^{J} \sum_{i=1}^{I} \tilde{\phi}_i^m \phi_i^m \tilde{\theta}_j^m \theta_j^m - \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{\mu=1}^{m-1} \left(\phi_i^\mu \theta_j^\mu \tau_k^\mu - \tilde{\phi}_i^\mu \tilde{\theta}_j^\mu \tilde{\tau}_k^\mu \right) \phi_i^m \theta_j^m \\
Q_{2t}^k &= \sum_{i=1}^{I} \sum_{j=1}^{J} (\phi_i^m)^2 (\theta_j^m)^2 \\
Q_{1T}^{K+1} &= \tau_{K+1}^m = \sum_{i=1}^{I} \sum_{j=1}^{J} \phi_i^m \theta_j^m u_{i,j,K+1} - \sum_{\mu=1}^{m-1} \left[\tau_{K+1}^\mu \sum_{i=1}^{I} \sum_{j=1}^{J} \phi_i^\mu \phi_i^m \theta_j^m \theta_j^\mu \right]
\end{aligned}$$

Eqs. N.5 are coupled and must be solved iteratively. At the first time step, the ϕ,

θ and τ functions are initialized by implementing the PGD algorithm to the 2D spatial field $u(x,y,t\!=\!0)$.

N.2 Implementation of the Incremental PGD Based on the Ghost-Cell Method

Herein, the iPGD method supports the shape optimization of an isolated airfoil parameterized by Bézier curves, where the design variables are their control points' coordinates. Two applications will be demonstrated. Firstly, a stationary airfoil is studied, in which unsteadiness is introduced by the time-varying far-field flow angle. Secondly, the airfoil is pitching following a sinusoidal motion over constant far-field conditions. The unsteady Euler equations are solved in both cases.

By definition, the PGD is applied only to structured meshes. The lack of structure in the used Cartesian mesh due to refinement techniques is overcome through a reference uniform mesh constituted by cells belonging to the higher refinement level of the original mesh. Hence, at every time step, each cell of the reference mesh stores the flow variables corresponding to the cell of the unstructured mesh they belong to by using an efficient searching algorithm based on a quad-tree data structure. After that, the iPGD algorithm is implemented to the reference mesh, as explained in section N.1. The opposite process is followed for the flow field reconstruction during the adjoint equations' inverse time integration.

In the first application, the far-field flow conditions are $M_\infty\!=\!0.3$ and $a_\infty\!=\!Asin(\omega t)$ with amplitude $A=3°$ and period $T=0.015s$.The mesh used for the simulation consists of 10500 cells and a constant time step equal to $T/20$ is used. The iPGD is carried out by setting $M\!=\!10$ and $w\!=\!1000$.

Fig. N.1 shows the effect of the flow solution approximation through the iPGD method on the accuracy of the sensitivity derivatives. Sensitivities computed by the posteriori PGD compression, eq. N.3, are also shown, obtaining a good indication of the best accuracy the iPGD could ideally attain. Furthermore, the deviation in the derivatives due to the incremental algorithm between the iPGD and the reference values computed by the full storage is negligible, demonstrating the capabilities of the proposed incremental algorithm.

In the pitching airfoil case, the mesh is changing in time, and the average number of cells is about 7000. The airfoil exposed to $M_\infty = 0.3$ is oscillating around the 1/4 of the chord with the position angle following a sinusoidal function of amplitude $A = 3°$ and period equal to $T = 0.015s$. The impact of the compressed flow fields on the sensitivity derivatives was examined by solving the adjoint equations twice with full storage and the flow data retrieved by the iPGD algorithm. For the two aforementioned cases, the sensitivity derivatives are computed and presented in fig. N.2. Two extra curves for 20 and 30 modes are shown. As the number of modes increases, the deviation in the computed derivatives diminishes. The saving in memory using the proposed iPGD algorithm with $M = 30$ is around 21% of the full storage, which needs an average of $140K$ values to be stored in memory.

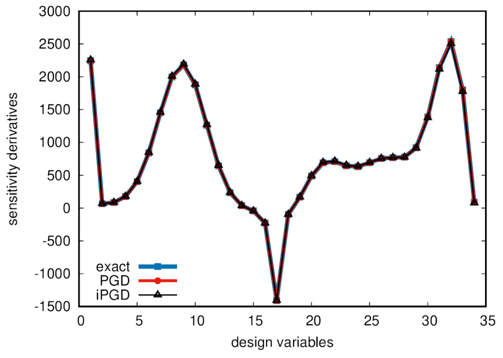


Figure N.1: PGD Inviscid Flow Reconstruction around a Stationary Airfoil: Comparison of the sensitivity derivatives computed using full storage (blue), the a posteriori PGD (red), and the iPGD (black) method.

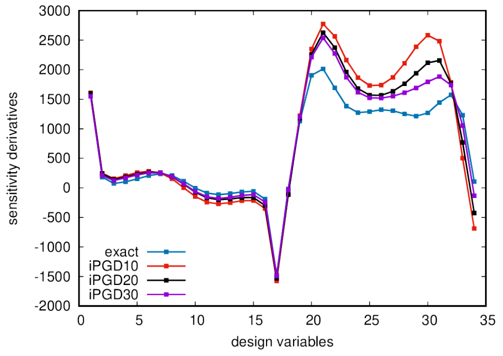


Figure N.2: PGD Inviscid Flow Reconstruction around a Moving Airfoil: Comparison of the sensitivity derivatives computed using full storage and the flow solution retrieved from the iPGD with $M = 10,\ 20,\ 30$.

Appendix O

The Absolute Roe Jacobian Derivative For Compressible Flows

The absolute Jacobian, defined in subsection 3.2.2, plays a central role in the Roe scheme used to discretize the compressible flow equations. Its derivative w.r.t. the primitive flow variables is necessary for the compressible discrete adjoint equations and, specifically, for the computation of matrix $A^{d,C}$ defined in eq. 7.10. The derivative consists of two parts examined separately. Firstly, the derivative of the absolute Jacobian w.r.t. the Roe averages is presented, followed by the computation of the Roe averages' derivatives w.r.t. the primitive variables,

$$\sum_k \frac{|\partial \tilde{A}_{imk} n_k^F|}{\partial V_j^C} = \sum_n \sum_k \frac{|\partial \tilde{A}_{imk} n_k^F|}{\partial \tilde{V}_n^F} \frac{\partial \tilde{V}_n^F}{\partial V_j^C}$$

$$\sum_k \frac{|\partial \tilde{A}_{imk} n_k^F|}{\partial \tilde{V}_j^F} = \sum_k \frac{\partial \tilde{P}_{ik}}{\partial \tilde{V}_j^F} |\tilde{\lambda}_k| \tilde{P}_{km}^{-1} + \sum_k \tilde{P}_{ik} \frac{\partial |\tilde{\lambda}_k|}{\partial \tilde{V}_j^F} \tilde{P}_{km}^{-1} + \sum_k \tilde{P}_{ik} |\tilde{\lambda}_k| \frac{\partial \tilde{P}_{km}^{-1}}{\partial \tilde{V}_j^F}$$

Therefore, the derivatives of the eigenvalues and eigenvectors of the absolute Jacobian are required. They are expressed as

$$\frac{\partial |\vec{\tilde{\lambda}}|}{\partial \tilde{V}_m} = \begin{bmatrix} sign(\tilde{v}_n)\ \partial \tilde{v}_n / \partial \tilde{V}_m \\ sign(\tilde{v}_n)\ \partial \tilde{v}_n / \partial \tilde{V}_m \\ sign(\tilde{v}_n)\ \partial \tilde{v}_n / \partial \tilde{V}_m \\ sign(\tilde{v}_n + \tilde{c})(n_m + \delta_{m5}) \\ sign(\tilde{v}_n - \tilde{c})(n_m - \delta_{m5}) \end{bmatrix}$$

where

$$
\begin{aligned}
\tilde{v}_n &= \sum_k \tilde{v}_k n_k, \\
\frac{\partial \tilde{v}_n}{\partial \tilde{V}_m} &= \begin{cases} 0, & m = 1,5 \\ n_{m-1}, & m = 2,3,4, \end{cases} \\
sign(x) &= \begin{cases} 1, & x \geqslant 0 \\ -1, & x < 0, \end{cases}
\end{aligned}
$$

and

$$
\frac{\partial \tilde{P}}{\partial \tilde{V}_m} = \begin{bmatrix}
0 & 0 & 0 & \alpha_m & \alpha_m \\
\delta_{m2} n_1 & \delta_{m2} n_2 - \delta_{m1} n_3 & \delta_{m2} n_3 + \delta_{m1} n_2 & b_{m2} + \frac{1}{2}\delta_{m1} n_1 & b_{m2} - \frac{1}{2}\delta_{m1} n_1 \\
\delta_{m3} n_1 + \delta_{m1} n_3 & \delta_{m3} n_2 & \delta_{m3} n_3 - \delta_{m1} n_1 & b_{m3} + \frac{1}{2}\delta_{m1} n_2 & b_{m3} - \frac{1}{2}\delta_{m1} n_2 \\
\delta_{m4} n_1 - \delta_{m1} n_2 & \delta_{m3} n_2 + \delta_{m1} n_1 & \delta_{m4} n_3 & b_{m4} + \frac{1}{2}\delta_{m1} n_3 & b_{m4} - \frac{1}{2}\delta_{m1} n_3 \\
g_1^m & g_2^m & g_3^m & b_{m5} + d_m & b_{m5} - d_m
\end{bmatrix}
$$

where

$$
\begin{aligned}
\alpha_m &= \frac{1}{2}\tilde{c}\delta_{m_1} - \frac{1}{2}\frac{\tilde{\rho}}{\tilde{c}^2}\delta_{m5} \\
b_{ml} &= a_m \tilde{v}_l + \frac{1}{2}\frac{\tilde{\rho}}{\tilde{c}}\delta_{ml}
\end{aligned}
$$

$$
\begin{aligned}
d_m &= \frac{1}{2}\delta_{m1}\tilde{v}_n + \begin{cases} 0, & m = 1,5 \\ \frac{1}{2}\tilde{\rho} n_{m-1}, & m = 2,3,4 \end{cases} \\
\vec{g}^m &= \begin{cases} \vec{\tilde{v}} \times \vec{n}, & m = 1 \\ \rho \vec{q}^{m-1} + \tilde{v}_{m-1}\vec{n}, & m = 2,3,4 \\ 0, & m = 5 \end{cases} \\
q_i^j &= \sum_k \epsilon_{ijk} n_k
\end{aligned}
$$

The derivative of $\tilde{P}^{-1}$ is given by differentiating the $\tilde{P}^{-1}\tilde{P} = I$ identity,

$$
\frac{\partial \tilde{P}^{-1}_{im}}{\partial \tilde{V}_j} = -\sum_l \sum_n \tilde{P}^{-1}_{il} \frac{\partial \tilde{P}_{ln}}{\partial \tilde{V}_j} \tilde{P}^{-1}_{nm}
$$

Finally, the derivatives of the Roe averages, eq. 3.8, w.r.t. the flow variables $\vec{V}^P$ and $\vec{V}^Q$ are

$$\frac{\partial \tilde{V}_i^F}{\partial V_j^P} = \frac{\partial \tilde{V}_i}{\partial V_j}\left(\vec{V}^P, \vec{V}^Q\right)$$
$$\frac{\partial \tilde{V}_i^F}{\partial V_j^Q} = \frac{\partial \tilde{V}_i}{\partial V_j}\left(\vec{V}^Q, \vec{V}^P\right)$$

where

$$\frac{\partial \vec{\tilde{V}}}{\partial \vec{V}}\left(\vec{V}^L, \vec{V}^R\right) = \begin{bmatrix} \frac{1}{2}\frac{V_1^R}{\tilde{V}_1} & 0 & 0 & 0 & 0 \\ r_1(V_2^L - \tilde{V}_2) & r_2 & 0 & 0 & 0 \\ r_1(V_3^L - \tilde{V}_3) & 0 & r_2 & 0 & 0 \\ r_1(V_4^L - \tilde{V}_4) & 0 & 0 & r_2 & 0 \\ r_3(\frac{\partial \tilde{h}}{\partial V_1^L} - \frac{\partial |\vec{\tilde{v}}|^2}{\partial V_1^L}) & r_3(\frac{\partial \tilde{h}}{\partial V_2^L} - r_2\tilde{V}_2) & r_3(\frac{\partial \tilde{h}}{\partial V_3^L} - r_2\tilde{V}_3) & r_3(\frac{\partial \tilde{h}}{\partial V_4^L} - r_2\tilde{V}_4) & r_3\frac{\partial \tilde{h}}{\partial V_5^L} \end{bmatrix}$$

and

$$\frac{\partial \tilde{h}}{\partial V_i^L} = \frac{\partial \tilde{h}}{\partial \tilde{V}_i} r_2 + (h^L - h^R) r_1 \delta_{i1}$$
$$\frac{\partial |\vec{\tilde{v}}|^2}{\partial V_1^L} = r_1 \sum_k (V_{k+1}^L - \tilde{V}_{k+1})\tilde{V}_{k+1}$$

where

$$r_1 = \frac{1}{2\sqrt{\rho^L}} \frac{1}{\sqrt{\rho^L} + \sqrt{\rho^R}}$$
$$r_2 = \frac{\sqrt{\rho^L}}{\sqrt{\rho^L} + \sqrt{\rho^R}}$$
$$r_3 = \frac{\gamma - 1}{2\tilde{c}}$$

The derivatives of enthalpy w.r.t. the primitive variables are

$$\frac{\partial \tilde{h}}{\partial \vec{\tilde{V}}} = \begin{bmatrix} -\frac{\tilde{c}^2}{\tilde{p}(\gamma-1)} & \tilde{v}_1 & \tilde{v}_2 & \tilde{v}_3 & \frac{\tilde{c}^2}{\tilde{p}(\gamma-1)} \end{bmatrix}$$

Appendix P

The Absolute Roe Jacobian Derivative For Incompressible Flows

The absolute Jacobian matrix is part of the Roe scheme, adjusted and used to discretize the incompressible flow equations. The scheme's formulation is shown in eq. 3.28. Its differentiation w.r.t. the flow variables $\vec{V}$, presented in eq. 3.26, is necessary to compute the discrete adjoint flux, defined in eq. 7.26.

The differentiation process starts with implementing of a chain rule based on the Jacobian matrix dependency on the Roe averages, followed by the differentiation of eq. 3.30,

$$\sum_k \frac{|\partial \tilde{A}_{imk} n_k^F|}{\partial V_j^C} = \sum_n \sum_k \frac{|\partial \tilde{A}_{imk} n_k^F|}{\partial \tilde{V}_n^F} \frac{\partial \tilde{V}_n^F}{\partial V_j^C} = \frac{1}{2} \sum_k \frac{|\partial \tilde{A}_{imk} n_k^F|}{\partial \tilde{V}_j^C}$$
$$= \frac{1}{2} \left(\sum_k \frac{\partial \tilde{P}_{ik}}{\partial \tilde{V}_j^F} |\tilde{\lambda}_k| \tilde{P}_{km}^{-1} + \sum_k \tilde{P}_{ik} \frac{\partial |\tilde{\lambda}_k|}{\partial \tilde{V}_j^F} \tilde{P}_{km}^{-1} + \sum_k \tilde{P}_{ik} |\tilde{\lambda}_k| \frac{\partial \tilde{P}_{km}^{-1}}{\partial \tilde{V}_j^F} \right)$$

Subsequently, the derivatives of the absolute eigenvalues vector $(|\vec{\tilde{\lambda}}|)$ and the matrices comprising the right $(\tilde{P})$ and left $(\tilde{P}^{-1})$ eigenvectors are presented,

$$\frac{\partial |\vec{\tilde{\lambda}}|}{\partial \tilde{p}} = \vec{0},$$

$$\frac{\partial|\vec{\tilde{\lambda}}|}{\partial\tilde{v}_k} = \begin{bmatrix} sign(\tilde{v}_n)n_k \\ sign(\tilde{v}_n)n_k \\ sign(\tilde{v}_n+\tilde{c})(n_k+\alpha_k) \\ sign(\tilde{v}_n-\tilde{c})(n_k-\alpha_k) \end{bmatrix}$$

where

$$\tilde{v}_n = \sum_k \tilde{v}_k n_k,$$

$$sign(x) = \begin{cases} 1, & x \geqslant 0 \\ -1, & x < 0, \end{cases}$$

and

$$\frac{\partial\tilde{P}}{\partial\tilde{p}} = 0,$$

$$\frac{\partial\tilde{P}}{\partial\tilde{v}_k} = \frac{1}{\beta^2}\begin{bmatrix} 0 & 0 & \alpha_k\beta^2 & -\alpha_k\beta^2 \\ 0 & 0 & \delta_{k1}(\tilde{v}_n+\tilde{c})+\tilde{v}_1(n_k+\alpha_k) & \delta_{k1}(\tilde{v}_n-\tilde{c})+\tilde{v}_1(n_k-\alpha_k) \\ 0 & 0 & \delta_{k2}(\tilde{v}_n+\tilde{c})+\tilde{v}_2(n_k+\alpha_k) & \delta_{k2}(\tilde{v}_n-\tilde{c})+\tilde{v}_2(n_k-\alpha_k) \\ 0 & 0 & \delta_{k3}(\tilde{v}_n+\tilde{c})+\tilde{v}_3(n_k+\alpha_k) & \delta_{k3}(\tilde{v}_n-\tilde{c})+\tilde{v}_3(n_k-\alpha_k) \end{bmatrix}$$

where $\tilde{v}_n = \sum_k \tilde{v}_k n_k$. The dashed variables $\vec{\tilde{V}}$ are defined in eq. 3.31 and the dashed speed of sound is

$$\tilde{c} = \sqrt{\tilde{v}_n^2 + \beta^2}$$

Moreover, vector $\vec{\alpha}$ is defined as

$$\alpha_k = \frac{\partial\tilde{c}}{\partial\tilde{v}_k} = \frac{\tilde{v}_n n_k}{\tilde{c}}$$

After computing $\partial\tilde{P}/\partial\tilde{V}_j$, the derivative of $\tilde{P}^{-1}$ is easily found by differentiating the $\tilde{P}^{-1}\tilde{P} = I$ identity,

$$\frac{\partial\tilde{P}^{-1}_{im}}{\partial\tilde{V}_j} = -\sum_l\sum_n \tilde{P}^{-1}_{il}\frac{\partial\tilde{P}_{ln}}{\partial\tilde{V}_j}\tilde{P}^{-1}_{nm}$$

Bibliography

[1] The EASY (Evolutionary Algorithms SYstem) software. http://velos0.ltt.mech.ntua.gr/EASY.

[2] T. AbdelMigid, K. Saqr, M. Kotb, and A. Aboelfarag. Revisiting the Lid-Driven Cavity Flow Problem: Review and New Steady State Benchmarking Results Using GPU Accelerated Code. *Alexandria Engineering Journal*, 56(1):123–135, 2017.

[3] D. Abel and D. Mark. A Comparative Analysis of Some Two-Dimensional Orderings. *International Journal of Geographical Information Systems*, 4(1):21–31, 1990.

[4] G. Adomavicius, M. Aftosmis, and M. Berger. A Parallel Cartesian Approach for External Aerodynamics of Vehicles with Complex Geometry. In *The tenth Thermal and Fluids Analysis Workshop, Huntsville, AL*, September 1999.

[5] M. Aftosmis, M. Berger, and G. Adomavicius. A Parallel Multilevel Method for Adaptively Refined Cartesian Grids with Embedded Boundaries. In *38th Aerospace Sciences Meeting and Exhibit*, 2000.

[6] M. Aftosmis, M. Berger, and J. Melton. Robust and Efficient Cartesian Mesh Generation for Component-Based Geometry. *AIAA Journal*, 36(6):952–960, 1998.

[7] M. Aftosmis, M. Berger, and S. Murman. Applications of Space-Filling-Curves to Cartesian Methods for CFD. In *42nd AIAA Aerospace Sciences Meeting and Exhibit*, January 2004.

[8] C. Albone. Embedded Meshes of Controllable Quality Synthesised from Elementary Geometric Features. In *30th Aerospace Sciences Meeting and Exhibit*, 1992.

[9] A. Ammar, F. Chinesta, E. Cueto, and M. Doblaré. Proper Generalized Decomposition of Time-Multiscale Models. *International Journal for Numerical Methods in Engineering*, 90(5):569–596, 2012.

[10] J. Anagnostopoulos. Discretization of Transport Equations on 2D Cartesian Unstructured Grids Using Data from Remote Cells for the Convection Terms. *International Journal of Numerical Methods in Fluids*, 42:297–321, 2003.

[11] J. Anagnostopoulos. A Cartesian Grid Method for the Simulation of Flows in Complex Geometries. October 2007.

[12] J. Anagnostopoulos. A Fast Numerical Method for Flow Analysis and Blade Design in Centrifugal Pump Impellers. *Computers & Fluids*, 38(2):284–289, 2009.

[13] J. Anagnostopoulos and D. Mathioulakis. Numerical Simulation and Hydrodynamic Design Optimization of a Tesla-Type Valve for Micropumps. August 2005.

[14] J.D. Anderson. *Fundamentals of Aerodynamics*. International student edition. McGraw-Hill, 1984.

[15] K. Anderson and J. Batina. Accurate Solutions, Parameter Studies and Comparisons for the Euler and Potential Flow Equations. AGARD, Validation of Computational Fluid Dynamics. Volume 1: Symposium Papers and Round Table Discussion, January 1989.

[16] K. Anderson and D. Bonhaus. Aerodynamic Design on Unstructured Grids for Turbulent Flows. 1997.

[17] K. Anderson, J. Newman, D. Whitfield, and E. Nielsen. Sensitivity Analysis for Navier-Stokes Equations on Unstructured Meshes Using Complex Variables. *AIAA Journal*, 39, November 1999.

[18] W. Anderson and V. Venkatakrishnan. Aerodynamic Design Optimization on Unstructured Grids with a Continuous Adjoint Formulation. *Computers & Fluids*, 28(4):443–480, 1999.

[19] P. Angot. A Fictitious Domain Model for the Stokes/Brinkman Problem with Jump Embedded Boundary Conditions. *Comptes Rendus Mathematique*, 348(11):697–702, 2010.

[20] P. Angot, C. Bruneau, and P. Fabrie. A Penalization Method to Take into Acount Obstacles in Viscous Flows. *Numerische Mathematik*, 81:497–520, February 1999.

[21] S. Asao, S. Ishihara, K. Matsuno, and M. Yamakawa. Progressive Development of Moving-Grid Finite-Volume Method for Three-Dimensional Incompressible Flows. pages 127–134, January 2010.

[22] M. Bader. *Space-filling Curves. An Introduction With Applications in Scientific Computing*, volume 9. January 2013.

[23] T. Baker and P. Cavallo. Dynamic Adaptation for Deforming Tetrahedral Meshes. In *14th Computational Fluid Dynamics Conference*, 1999.

[24] E. Balaras. Modeling Complex Boundaries Using an External Force Field on Fixed Cartesian Grids in Large-Eddy Simulations. *Computers & Fluids*, 33(3):375–404, 2004.

[25] L. Balzano and S. Wright. On GROUSE and incremental SVD. In *2013 5th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 1–4, 2013.

[26] T. Barth and D. Jespersen. The Design and Application of Upwind Schemes on Unstructured Meshes. In *27th Aerospace Sciences Meeting*, 1989.

[27] P. Barton, B. Obadia, and D. Drikakis. A Conservative Level-Set Based Method for Compressible Solid/Fluid Problems on Fixed Grids. *Journal of Computational Physics*, 230(21):7867–7890, 2011.

[28] O. Baysal and M. Eleshaky. Aerodynamic Sensitivity Analysis Methods for the Compressible Euler Equations. *Journal of Fluids Engineering-transactions of The Asme*, 113:681–688, 1991.

[29] S. Bayyuk, K. Powell, and B. van Leer. A Simulation Technique for 2-D Unsteady Inviscid Flows Around Arbitrarily Moving and Deforming Bodies of Arbitrary Geometry. July 1993.

[30] J. Behrens and J. Zimmermann. Parallelizing an Unstructured Grid Generator with a Space-Filling Curve Approach. pages 815–823, August 2000.

[31] I. Bell. *Theoretical and Experimental Analysis of Liquid Flooded Compression in Scroll Compressors*. PhD thesis, Purdue University, West Lafayette, Indiana, 2011.

[32] E. Beltrami. Sulle Funzioni Bilineari. *Giornale di Matematiche ad Uso degli Studenti Delle Università Italiane*, 11:98–106, 1873. English translation by D. Boley, 1990.

[33] J. Benek, J. Steger, and F. Dougherty. A Chimera Grid Scheme. page 59–69, 1983.

[34] J. Benk, H. Bungartz, M. Mehl, and M. Ulbrich. *Immersed Boundary Methods for Fluid-Structure Interaction and Shape Optimization within an FEM-Based PDE Toolbox*, volume 93, pages 25–56. January 2013.

[35] W. Bennett, N. Nikiforakis, and R. Klein. A Moving Boundary Flux Stabilization Method for Cartesian Cut-Cell Grids Using Directional Operator Splitting. *Journal of Computational Physics*, 368, November 2017.

[36] M. Berger and M. Aftosmis. Progress Towards a Cartesian Cut-Cell Method for Viscous Compressible Flow. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2012.

[37] M. Berger and M. Aftosmis. An ODE-Based Wall Model for Turbulent Flow Simulations. *AIAA Journal*, 56(2):700–714, 2018.

[38] M. Berger, M. Aftosmis, and S. Murman. Analysis of Slope Limiters on Irregular Grids. *43rd AIAA Aerospace Sciences Meeting and Exhibit - Meeting Papers*, February 2005.

[39] M. Berger and P. Colella. Local Adaptive Mesh Refinement for Shock Hydrodynamics. *Journal of Computational Physics*, 82(1):64–84, 1989.

[40] M. Berger, C. Helzel, and R. Leveque. H-Box Methods for the Approximation of Hyperbolic Conservation Laws on Irregular Grids. *SIAM Journal on Numerical Analysis*, 41(3):893–918, 2003.

[41] M. Berger and J. Oliger. Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations. *Journal of Computational Physics*, 53(3):484–512, 1984.

[42] A. Bernland, E. Wadbro, and M. Berggren. Acoustic Shape Optimization Using Cut Finite Elements. *International Journal for Numerical Methods in Engineering*, 113(3):432–449, 2018.

[43] P. Berthelsen and O. Faltinsen. A Local Directional Ghost Cell Approach for Incompressible Viscous Flow Problems with Irregular Boundaries. *Journal of Computational Physics*, 227(9):4354–4397, 2008.

[44] R. Beyer and R. Leveque. Analysis of a One-Dimensional Model for the Immersed Boundary Method. *SIAM J. Numer. Anal.*, 29(2):332–364, April 1992.

[45] D. Böhm, S. Berchtold, and D. Keim. Searching in High-Dimensional Spaces: Index Structures for Improving the Performance of Multimedia Databases. *ACM Computing Surveys*, 33(3):322–373, 2001.

[46] K. Boopathy and M. Rumpfkeil. A Multivariate Interpolation and Regression Enhanced Kriging Surrogate Model. June 2013.

[47] M. Brand. Incremental Singular Value Decomposition Of Uncertain Data With Missing Values. In *Computer Vision – ECCV 2002*, pages 707–720. Springer Berlin Heidelberg, 2002.

[48] M. Brand. Fast Low-Rank Modifications of the Thin Singular Value Decomposition. *Linear Algebra and its Applications*, 415(1):20–30, 2006. Special Issue on Large Scale Linear and Nonlinear Eigenvalue Problems.

[49] A. F. Browne and A. B. Paustian. Noise Analysis Methodology for a Dual-Diaphragm Medical Device Air Pump. In *SoutheastCon 2016*, pages 1–7, 2016.

[50] A. Butz. Space Filling Curves and Mathematical Programming. *Information and Control*, 12(4):314–330, 1968.

[51] F. Capizzano. Automatic Generation of Locally Refined Cartesian Meshes: Data Management and Algorithms. *International Journal for Numerical Methods in Engineering*, 113, August 2017.

[52] J. Caridad and F. Kenyery. CFD Analysis of Electric Submersible Pumps (ESP) Handling Two-Phase Mixtures. *Journal of Energy Resources Technology-transactions of The Asme - J. Energy Resour. Technol*, 126, 06 2004.

[53] D. Causon, D. Ingram, and C. Mingham. A Cartesian Cut Cell Method for Shallow Water Flows with Moving Boundaries. *Advances in Water Resources*, 24(8):899–911, 2001.

[54] D. Cecere and E. Giacomazzi. An Immersed Volume Method for Large Eddy Simulation of Compressible Flows Using a Staggered-Grid Approach. *Computer Methods in Applied Mechanics and Engineering*, 280:1–27, 2014.

[55] E. Charlton and K. Powell. An Octree Solution to Conservation Laws over Arbitrary Regions (OSCAR). In *35th Aerospace Sciences Meeting and Exhibit*, 1997.

[56] Z. Chen, S. Hickel, A. Devesa, J. Berland, and N. Adams. Wall Modeling for Implicit Large-Eddy Simulation and Immersed-Interface Methods. *Theoretical and Computational Fluid Dynamics*, 28, March 2013.

[57] Y. Cheny and O. Botella. The LS-STAG method: A New Immersed Boundary/Level-Set Method for the Computation of Incompressible Viscous Flows in Complex Moving Geometries with Good Conservation Properties. *Journal of Computational Physics*, 229(4):1043–1076, 2010.

[58] C. Chevalier and F. Pellegrini. PT-Scotch: A Tool for Efficient Parallel Graph Ordering. *Parallel Computing*, 34(6):318–331, 2008.

[59] Y. Chiang, B. van Leer, and K. Powell. Simulation of Unsteady Inviscid Flow on an Adaptively Refined Cartesian Grid. In *30th AIAA Aerospace Sciences Meeting and Exhibit*, January 1992.

[60] F. Chinesta, R. Keunings, and A. Leygue. *The Proper Generalized Decomposition for Advanced Numerical Simulations, A Primer.* Springer International Publishing, Nantes, France, 2014.

[61] Y. Cho and A. Aessopos. Similarity Transformation Methods in the Analysis of the Two Dimensional Steady Compressible Laminar Boundary Layer. *Term Paper, 2.26 Compressible Fluid Dynamics, Massachusetts Institute of Technology*, Spring 2004.

[62] J. Choi, R. Oberoi, J. Edwards, and J. Rosati. An Immersed Boundary Method for Complex Incompressible Flows. *Journal of Computational Physics*, 224(2):757–784, 2007.

[63] A. Chorin. A Numerical Method for Solving Incompressible Viscous Flow Problems. *Journal of Computational Physics*, 2(1):12–26, 1967.

[64] H. Chun-Wei and H. Song-Bin. A Microfluidic Device for Precise Pipetting. *Journal of Micromechanics and Microengineering*, 18:035004, 01 2008.

[65] D. Clarke, M. Salas, and H. Hassan. Euler Calculations for Multielement Airfoils Using Cartesian Grids. *AIAA Journal*, 24(3):353–358, 1986.

[66] S. Cliff, S. Thomas, T. Baker, A. Jameson, and R. Hicks. Aerodynamic Shape Optimization Using Unstructured Grid Methods. In *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2002.

[67] W. Coirier. An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations. Ph.D. Thesis - Michigan Univ. 1994.

[68] W. Coirier and K. Powell. An Accuracy Assessment of Cartesian-Mesh Approaches for the Euler Equations. *Journal of Computational Physics*, 117(1):121–131, 1995.

[69] W. Coirier and K. Powell. Solution-Adaptive Cartesian Cell Approach for Viscous and Inviscid Flows. *AIAA Journal*, 34(5):938–945, 1996.

[70] P. Colella, D. Graves, B. Keen, and D. Modiano. A Cartesian Grid Embedded Boundary Method for Hyperbolic Conservation Laws. *Journal of Computational Physics*, 211(1):347–366, 2006.

[71] F. Courty, A. Dervieux, B. Koobus, and L. Hascoët. Reverse Automatic Differentiation for Optimum Design: From Adjoint State Assembly to Gradient Computation. *Optimization Methods and Software*, 18:615–627, Octover 2003.

[72] L. Creux. Rotary Engine, U.S. Patent 801,182, October 1905.

[73] A. Dadone and B. Grossman. Efficient Fluid Dynamic Design Optimization Using Cartesian Grids. In *16th AIAA Computational Fluid Dynamics Conference*, 2003.

[74] A. Dadone and B. Grossman. Ghost-Cell Method for Inviscid Two-Dimensional Flows on Cartesian Grids. *AIAA Journal*, 42(12):2499–2507, 2004.

[75] P.-E. Danielsson. Euclidean Distance Mapping. *Computer Graphics and Image Processing*, 14(3):227–248, 1980.

[76] S. Davis. Simplified Second-Order Godunov-Type Methods. *SIAM Journal on Scientific and Statistical Computing*, 9(3):445–473, 1988.

[77] M. Delanaye, M. Aftosmis, M. Berger, Y. Liu, and T. Pulliman. Automatic Hybrid-Cartesian Grid Generation for High-Reynolds Number Flows around Complex Geometries. In *37th Aerospace Sciences Meeting and Exhibit*, 1999.

[78] I. Demirdzic. On the Discretization of the Diffusion Term in Finite-Volume Continuum Mechanics. *Numerical Heat Transfer Part B: Fundamentals*, 68, July 2015.

[79] J. Dèsidèri and A. Janka. Hierarchical Parametrization for Multilevel Evolutionary Shape Optimization with Application to Aerodynamics. 2003.

[80] O. Desjardins, J. McCaslin, M. Owkes, and P. Brady. Direct Numerical and Large-Eddy Simulation of Primary Atomization in Complex Geometries. *Atomization and Sprays*, 23:1001–1048, January 2013.

[81] P. Dhananchezhiyan and S. Hiremath. Optimization of Multiple Micro Pumps to Maximize the Flow Rate and Minimize the Flow Pulsation. *Procedia Technology, 1st Global Colloquium on Recent Advancements and Effectual Researches in Engineering, Science and Technology - RAEREST*, 25:1226–1233, April 2016.

[82] S. Dilgen, J. Jensen, and N. Aage. Shape Optimization of the Time-Harmonic Response of Vibroacoustic Devices Using Cut Elements. *Finite Elements in Analysis and Design*, 196, 2021.

[83] E. Duque, R. Biswas, and R. Strawn. A Solution Adaptive Structured/Unstructured Overset Grid Flow Solver with Applications to Helicopter Rotor Flows. In *13th Applied Aerodynamics Conference*, August 2012.

[84] B. Einfeldt. On Godunov-Type Methods for Gas Dynamics. *Siam Journal on Numerical Analysis - SIAM J NUMER ANAL*, 25:294–318, April 1988.

[85] M. Eldred and J. Burkardt. Comparison of Non-Intrusive Polynomial Chaos and Stochastic Collocation Methods for Uncertainty Quantification. *47th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, January 2009.

[86] J. Elliott and J. Peraire. Aerodynamic Design Using Unstructured Meshes. 1996.

[87] D. Elsworth and E. Toro. Riemann Solvers for Solving the Incompressible Navier-Stokes Equations Using the Artificial Compressibility Method. *NASA STI/Recon Technical Report N*, pages 25778–, June 1992.

[88] E. Fadiga, N. Casari, A. Suman, and M. Pinelli. Structured Mesh Generation and Numerical Analysis of a Scroll Expander in an Open-Source Environment. *Energies*, 13:666, 02 2020.

[89] 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*, 161(1):35–60, 2000.

[90] R. Fedkiw. Coupling an Eulerian Fluid Calculation to a Lagrangian Solid Calculation with the Ghost Fluid Method. *Journal of Computational Physics*, 175(1):200–224, 2002.

[91] R. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A Non-Oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (the Ghost Fluid Method). *Journal of Computational Physics*, 152(2):457–492, 1999.

[92] R. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A Non-oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (the Ghost Fluid Method). *Journal of Computational Physics*, 152(2):457–492, 1999.

[93] J. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics.* New York:Springer-Verlag, 1996.

[94] U. Fey, M. König, and H. Eckelmann. A New Strouhal–Reynolds-Number Relationship for the Circular Cylinder in the Range $47 < Re < 2\times105$. *Physics of Fluids*, 10(7):1547–1549, 1998.

[95] R. Fletcher. *Practical Methods of Optimization.* John Wiley & Sons, Ltd, 2000.

[96] R. Fletcher and C. M. Reeves. Function Minimization by Conjugate Gradients. *The Computer Journal*, 7(2):149–154, January 1964.

[97] North Atlantic Treaty Organization. Advisory Group for Aerospace Research and Development. Fluid Dynamics Panel. Working Group 07. *Test Cases for Inviscid Flow Field Methods: Report of Fluid Dynamics Panel Working Group 07.* AGARD advisory report. AGARD, 1985.

[98] N. Foster and G. Dulikravich. Three-Dimensional Aerodynamic Shape Optimization Using Genetic and Gradient Search Algorithms. *Journal of Spacecraft and Rockets - J SPACECRAFT ROCKET*, 34:36–42, January 1997.

[99] R. Gaffney, H. Hassan, and M. Salas. Euler Calculations for Wings Using Cartesian Grids. *AIAA Paper 87-0356*, 1987.

[100] T. Gao, Y. Tseng, and X. Lu. An Improved Hybrid Cartesian/Immersed Boundary Method for Fluid–Solid Flows. *International Journal for Numerical Methods in Fluids*, 55(12):1189–1211, 2007.

[101] U. Ghia, K.N. Ghia, and C.T. Shin. High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method. *Journal of Computational Physics*, 48(3):387–411, 1982.

[102] F. Gibou, R. Fedkiw, L. Cheng, and M. Kang. A Second-Order-Accurate Symmetric Discretization of the Poisson Equation on Irregular Domains. *Journal of Computational Physics*, 176(1):205–227, 2002.

[103] M. Giles, M. Duta, J. Muller, and N. Pierce. Algorithm Developments for Discrete Adjoint Methods. *AIAA Journal*, 41(2):198–205, 2003.

[104] M. Giles and N. Pierce. An Introduction to the Adjoint Approach to Design. *Flow, Turbulence and Combustion*, 65, April 2000.

[105] A. Gilmanov, F. Sotiropoulos, and E. Balaras. A General Reconstruction Algorithm for Simulating Flows with Complex 3D Immersed Boundaries on Cartesian Grids. *Journal of Computational Physics*, 191(2):660–669, 2003.

[106] P. Glaister. An Approximate Linearised Riemann Solver for the Euler Equations for Real Gases. *Journal of Computational Physics*, 74(2):382–408, 1988.

[107] R. Glowinski, T.W. Pan, and J. Periaux. A Fictitious Domain Method for Dirichlet Problem and Applications. *Computer Methods in Applied Mechanics and Engineering*, 111(3):283–303, 1994.

[108] S. Godunov. A difference Scheme for Numerical Solution of Discontinuous Solution of Hydrodynamic Equations. *Math. Sbornik*, 47:271–306, 1959.

[109] S. Godunov and I. Bohachevsky. Finite Difference Method for Numerical Computation of Discontinuous Solutions of the Equations of Fluid Dynamics. *Matematičeskij sbornik*, 47(89)(3):271–306, 1959.

[110] N. Gokhale, N. Nikiforakis, and R. Klein. A Dimensionally Split Cartesian Cut Cell Method for the Compressible Navier–Stokes Equations. *Journal of Computational Physics*, 375:1205–1219, 2018.

[111] D. Goldstein, R. Handler, and L. Sirovich. Modeling a no-slip flow boundary with an external force field. *Journal of Computational Physics*, 105(2):354–366, 1993.

[112] G. Golub and W. Kahan. Calculating the Singular Values and Pseudo-Inverse of a Matrix. *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis*, 2(2):205–224, 1965.

[113] G. Golub and C. Van Loan. *Matrix Computations.* Johns Hopkins series in the mathematical sciences / in association with the Department of mathematical sciences, The Johns Hopkins University. Johns Hopkins University Press, 1996.

[114] Y. Gorsse, A. Iollo, H. Telib, and L. Weynans. A Simple Second Order Cartesian Scheme for Compressible Euler Flows. *Journal of Computational Physics*, 231(23):7780–7794, 2012.

[115] M. Griebel, T. Neunhoeffer, and H. Regler. Algebraic Multigrid Methods for the Solution of the Navier–Stokes Equations in Complicated Geometries. *International Journal for Numerical Methods in Fluids*, 26(3):281–301, 1998.

[116] A. Griewank and A. Walther. Algorithm 799: Revolve: An Implementation of Checkpointing for the Reverse or Adjoint Mode of Computational Differentiation. *ACM Trans. Math. Softw.*, 26(1):19–45, March 2000.

[117] B. Griffith and C. Peskin. On the Order of Accuracy of the Immersed Boundary Method: Higher Order Convergence Rates for Sufficiently Smooth Problems. *Journal of Computational Physics*, 208(1):75–105, 2005.

[118] M. Grinfeld and P. Grinfeld. The Gibbs Method in Thermodynamics of Heterogeneous Substances Carrying Electric Charges. *Results in Physics*, 6:194–195, 2016.

[119] P Grinfeld. *Introduction to Tensor Analysis and the Calculus of Moving Surfaces.* January 2013.

[120] C. Günther, M. Meinke, and W. Schröder. A Flexible Level-Set Approach for Tracking Multiple Interacting Interfaces in Embedded Boundary Methods. *Computers & Fluids*, 102:182–202, 2014.

[121] A. Harten. High Resolution Schemes for Hyperbolic Conservation Laws. *Journal of Computational Physics*, 49(3):357–393, 1983.

[122] A. Harten, J. Hyman, P. Lax, and B. Keyfitz. On Finite-Difference Approximations and Entropy Conditions for Shocks. *Communications on Pure and Applied Mathematics*, 29(3):297–322, 1976.

[123] D. Hartmann, M. Meinke, and W. Schröder. An Adaptive Aultilevel Multigrid Formulation for Cartesian Hierarchical Grid Methods. *Computers & Fluids*, 37(9):1103–1125, 2008.

[124] D. Hartmann, M. Meinke, and W. Schröder. A Strictly Conservative Cartesian Cut-Cell Method for Compressible Viscous Flows on Adaptive Grids. *Computer Methods in Applied Mechanics and Engineering*, 200(9):1038–1052, 2011.

[125] L. Hascoët, J. Utke, and U. Naumann. Cheaper Adjoints by Reversing Address Computations. *Sci. Program.*, 16(1):81–92, January 2008.

[126] H. Haverkort. *An Inventory of Three-Dimensional Hilbert Space-Filling Curves*, volume 1109.2323 of *arXiv.org [cs.CG]*. s.n., 2011.

[127] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, USA, 2nd edition, 1998.

[128] D. Hilbert. Über die stetige abbildung einer linie auf ein flächenstück. In *Dritter Band: Analysis · Grundlagen der Mathematik · Physik Verschiedenes: Nebst Einer Lebensgeschichte*, Berlin, Heidelberg, 1891. Springer Berlin Heidelberg.

[129] C. Hinterberger and M. Olesen. Automatic Geometry Optimization of Exhaust Systems Based on Sensitivities Computed by a Continuous Adjoint CFD Method in OpenFOAM. April 2010.

[130] Charles Hirsch. *Numerical Computation of Internal and External Flows (Second Edition)*. Butterworth-Heinemann, Oxford, second edition edition, 2007.

[131] C Hirt, A. Amsden, and J. Cook. An Arbitrary Lagrangian-Eulerian Computing Method for all Flow Speeds. *Journal of Computational Physics*, 14(3):227–253, 1974.

[132] X. Hu, B. Khoo, N. Adams, and F. Huang. A Conservative Interface Method for Compressible Flows. *Journal of Computational Physics*, 219(2):553–578, 2006.

[133] Z. Hu, D. Causon, C. Mingham, and L. Qian. A Cartesian Cut Cell Free Surface Capturing Method for 3D Water Impact Problems. *International Journal for Numerical Methods in Fluids*, 71(10):1238–1259, 2013.

[134] D. Ingram, D. Causon, and C. Mingham. Developments in Cartesian cut cell methods. *Mathematics and Computers in Simulation*, 61(3):561–572, 2003. MODELLING 2001 - Second IMACS Conference on Mathematical Modelling and Computational Methods in Mechanics, Physics, Biomechanics and Geodynamics.

[135] M. Inoue and M. Kuroumaru. Structure of Tip Clearance Flow in an Isolated Axial Compressor Rotor. *Journal of Turbomachinery*, 111(3):250–256, July 1989.

[136] M. Inoue, M. Kuroumaru, and M. Fukuhara. Behavior of Tip Leakage Flow Behind an Axial Compressor Rotor. *Journal of Engineering for Gas Turbines and Power*, 108(1):7–14, January 1986.

[137] Lee J., J. Kim, H. Choi, and K. Yang. Sources of Spurious Force Oscillations from an Immersed Boundary Method for Moving-Body Problems. *Journal of Computational Physics*, 230(7):2677–2695, 2011.

[138] P. Jacques and R. Dwight. Numerical Sensitivity Analysis for Aerodynamic Optimization: A Survey of Approaches. *Computers & Fluids*, 39(3):373–391, 2010.

[139] A. Jameson. Aerodynamic Design via Control Theory. *Journal of Scientific Computing*, 3, December 1988.

[140] A. Jameson and J. Reuther. Control Theory Based Airfoil Design Using the Euler Equations. October 1994.

[141] H. Jasak. *Error Analysis and Estimation for the Finite Volume Method With Applications to Fluid Flows.* PhD thesis, ImperialCollegeof Science,Technologyand Medicine, January 1996.

[142] N. Jenkins and K. Maute. An Immersed Boundary Approach for Shape and Topology Optimization of Stationary Fluid-Structure Interaction Problems. *Structural and Multidisciplinary Optimization*, 54, November 2016.

[143] J. Jeong and F. Hussain. On the Identification of a Vortex. JFM 285, 69-94. *Journal of Fluid Mechanics*, 285:69 – 94, February 1995.

[144] H. Ji, F. Lien, and E. Yee. A New Adaptive Mesh Refinement Data Structure with an Application to Detonation. *Journal of Computational Physics*, 229(23):8981–8993, 2010.

[145] H. Ji, F. Lien, and E. Yee. Numerical Simulation of Detonation Using an Adaptive Cartesian Cut-Cell Method Combined with a Cell-Merging Technique. *Computers & Fluids*, 39:1041–1057, June 2010.

[146] M. Jiang, R. Machiraju, and D. Thompson. Detection and Visualization of Vortices. In *The Visualization Handbook*, 2005.

[147] G. Jin and J. Mellor-Crummey. SFCGen: A Framework for Efficient Generation of Multi-Dimensional Space-Filling Curves by Recursion. *ACM Trans. Math. Softw.*, 31:120–148, March 2005.

[148] Y. Kallinderis. A Finite Volume Navier-Stokes Algorithm for Adaptive Grids. *International Journal for Numerical Methods in Fluids*, 15(2):193–217, 1992.

[149] I. Kampolis and K. Giannakoglou. A Multilevel Approach to Single- and Multiobjective Aerodynamic Optimization. *Computer Methods in Applied Mechanics and Engineering*, 197:2963–2975, June 2008.

[150] C. Kapellos. *The Continuous Adjoint Method for Automotive Aeroacoustic Shape Optimization*. PhD thesis, National Technical University of Athens, 2019.

[151] D. Kapsoulis. *Low-Cost Metamodel-Assisted Evolutionary Algorithms with Application in Shape Optimization in Fluid Dynamics*. PhD thesis, National Technical University of Athens, 2019.

[152] D. Kapsoulis, K. Tsiakas, X. Trompoukis, V. Asouti, and K. Giannakoglou. A PCA-assisted Hybrid Algorithm Combining EAs and Adjoint Methods for CFD-based Optimization. *Applied Soft Computing*, 73:520–529, 2018.

[153] D. Kapsoulis, K. Tsiakas, X. Trompoukis, V. Asouti, and K. Giannakoglou. A pca-assisted hybrid algorithm combining eas and adjoint methods for cfd-based optimization. *Applied Soft Computing*, 73:520–529, 2018.

[154] M. Karakasis and K. Giannakoglou. On the use of metamodel-assisted, multi-objective evolutionary algorithms. *Engineering Optimization*, 38(8):941–957, 2006.

[155] M. Karakasis and Giannakoglou K. On the Use of Metamodel-Assisted, Multi-Objective Evolutionary Algorithms. *Engineering Optimization*, 38(8):941–957, 2006.

[156] M. Karakasis, D. Koubogiannis, and K. Giannakoglou. Hierarchical Distributed Metamodel-Assisted Evolutionary Algorithms in Shape Optimization. *International Journal for Numerical Methods in Fluids*, 53(3):455–469, 2007.

[157] S. Karman. SPLITFLOW - A 3D Unstructured Cartesian/Prismatic Grid CFD Code for Complex Geometries. In *33rd Aerospace Sciences Meeting and Exhibit*, 1995.

[158] G. Karypis and V. Kumar. METIS—A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes and Computing Fill-Reducing Ordering of Sparse Matrices. January 1997.

[159] I. Kavvadias. *Continuous Adjoint Methods for Steady and Unsteady Turbulent flows with Emphasis on the Accuracy of Sensitivity Derivatives.* PhD thesis, National Technical University of Athens, 2016.

[160] I. Kavvadias, E. Papoutsis-Kiachagias, and K. Giannakoglou. On the Proper Treatment of Grid Sensitivities in Continuous Adjoint Methods for Shape Optimization. *Journal of Computational Physics*, 301:1–18, August 2015.

[161] K. Khadra, P. Angot, S. Parneix, and J. Caltagirone. Fictitious domain approach for numerical modelling of navier–stokes equations. *International Journal for Numerical Methods in Fluids*, 34(8):651–684, 2000.

[162] A. Khokhlov. Fully Threaded Tree Algorithms for Adaptive Refinement Fluid Dynamics Simulations. *Journal of Computational Physics*, 143(2):519–543, 1998.

[163] J. Kim, B. Ovgor, K. Cha, J. Kim, S. Lee, and K. Kim. Optimization of the Aerodynamic and Aeroacoustic Performance of an Axial-Flow Fan. *AIAA Journal*, 52:2032–2043, August 2014.

[164] S. Kim, J. Alonso, and A. Jameson. A gradient accuracy study for the adjoint-based navier-stokes design method. January 1999.

[165] M. Kirkpatrick, S. Armfield, and J. 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*, 184(1):1–36, 2003.

[166] E. Kontoleontos, V. Asouti, and K. Giannakoglou. An Asynchronous Metamodel-Assisted Memetic Algorithm for CFD-Based Shape Optimization. *Engineering Optimization*, 44(2):157–173, February 2012.

[167] D. Krause and F. Kummer. An Incompressible Immersed Boundary Solver for Moving Body Flows Using a Cut Cell Discontinuous Galerkin Method. *Computers & Fluids*, 153:118–129, 2017.

[168] N. Kroll, N. Gauger, J. Brezillon, R. Dwight, A. Fazzolari, D. Vollmer, K. Becker, H. Barnewitz, V. Schulz, and S. Hazra. Flow simulation and shape optimization for aircraft design. *Journal of Computational and Applied Mathematics*, 203(2):397–411, 2007.

[169] S. Kyriakou. *Evolutionary Algorithm-based Design-Optimization Methods in Turbomachinery*. PhD thesis, National Technical University of Athens, 2013.

[170] P. Ladevèze. PGD in Linear and Nonlinear Computational Solid Mechanics. In *Separated Representations and PGD-Based Model Reduction: Fundamentals and Applications*, Vienna, 2014. Springer.

[171] E. Ladopoulos. Four-dimensional Petroleum Exploration & Non-linear ESP Artificial Lift by Multiple Pumps for Petroleum Well Development. *Universal Journal of Hydraulics*, 3:1–14, 01 2015.

[172] M. Lai and C. Peskin. An Immersed Boundary Method with Formal Second-Order Accuracy and Reduced Numerical Viscosity. *Journal of Computational Physics*, 160(2):705–719, 2000.

[173] L. Landau and E. Lifshitz. *Fluid Mechanics*, volume 6 of *Course of Theoretical Physics*. Pergamon Press, 1987.

[174] R. Landon. NACA 0012 Oscillatory and Transient Pitching. page 16, October 2000.

[175] V. Lemort, S. Quoilin, Cuevas C., and Lebrun J. Testing and Modeling a Scroll Expander Integrated into an Organic Rankine Cycle. *Applied Thermal Engineering*, 29(14):3094–3102, 2009.

[176] M. Lesoinne and C. Farhat. Geometric Conservation Laws for Flow Problems with Moving Boundaries and Deformable Meshes, and their Impact on Aeroelastic Computations. *Computer Methods in Applied Mechanics and Engineering*, 134(1):71–90, 1996.

[177] R. LeVeque. *Finite Volume Methods for Hyperbolic Problems.* Cambridge Texts in Applied Mathematics. Cambridge University Press, 2002.

[178] Y. Levy, D. Degani, and A. Seginer. Graphical Visualization of Vortical Flows by Means of Helicity. *AIAA Journal*, 28(8):1347–1352, 1990.

[179] C. LI. Numerical Solution of Viscous Reacting Blunt Body Flows of a Multicomponent Mixture. In *11th Aerospace Sciences Meeting*, 1973.

[180] Y. Liang and B. Barsky. An Analysis and Algorithm for Polygon Clipping. *Commun. of the ACM*, 26:868–877, 1983.

[181] H. Liepmann and A. Roshko. *Elements of Gas Dynamics*, volume 10. January 2001.

[182] P. Lindstrom. Fixed-Rate Compressed Floating-Point Arrays. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2674–2683, 2014.

[183] J. Lions. *Optimal Control of Systems Governed by Partial Differential Equations*. Springer, Berlin, Heidelberg, 1971.

[184] X. Liu and G. Schrack. Encoding and Decoding the Hilbert Order. *Software: Practice and Experience*, 26(12):1335–1346, 1996.

[185] R. Löhner. Adaptive Remeshing for Transient Problems. *Computer Methods in Applied Mechanics and Engineering*, 75(1):195–214, 1989.

[186] R. Löhner, K. Morgan, and O. C. Zienkiewicz. *Effective Programming of Finte Element Methods for Computational Fluid Dynamics on Supercomputers*, pages 117–125. Vieweg+Teubner Verlag, Wiesbaden, 1986.

[187] D. Lovely and R. Haimes. Shock Detection from Computational Fluid Dynamics Results. In *14th Computational Fluid Dynamics Conference*, 1999.

[188] J. Lu. *An a posteriori Error Control Framework for Adaptive Precision Optimization Using Discontinuous Galerkin Finite Element Method.* PhD thesis, Massachusetts Institute of Technology, 2005.

[189] H. Luo, H. Dai, and P. Ferreira de Sousa. A Hybrid Formulation to Suppress the Numerical Oscillations Caused by Immersed Moving Boundaries. In *62nd Annual Meeting of the APS Division of Fluid Dynamics*, November 2009.

[190] S. Majumdar, G. Iaccarino, and P. Durbin. RANS Solvers with Adaptive Structured Boundary Non-Conforming Grids. *Annual Research Briefs*, January 2001.

[191] D. Mangolis, S. Craig, G. Nowakowski, and M. Inada. Modeling and Simulation of a Scroll Compressor Using Bond Graphs. In *Proceedings of the International Compressor Engineering Conference at Purdue*, 1992.

[192] A.-S. Margetis, E. Papoutsis-Kiachagias, and K. Giannakoglou. Lossy Compression Techniques Supporting Unsteady Adjoint on 2D/3D Unstructured Grids. *Computer Methods in Applied Mechanics and Engineering*, 387:114152, 2021.

[193] J. Martins, J. Alonso, and J. Reuther. High-Fidelity Aerostructural Design Optimization of a Supersonic Business Jet. *Journal of Aircraft*, 41:523–530, May 2004.

[194] J. Martins, P. Sturdza, and J. Alonso. The Complex-Step Derivative Approximation. *ACM Trans. Math. Softw.*, 29:245–262, 2003.

[195] A. Massing, M. Larson, A. Logg, and M. Rognes. A Nitsche-Based Cut Finite Element Method for a Fluid–Structure Interaction Problem. *Communications in Applied Mathematics and Computational Science*, 10, November 2013.

[196] Argonne National Laboratory Computer Science Division Mathematics and Rice University Center for Research on Parallel Computation. `https://www.mcs.anl.gov/research/projects/adifor/`.

[197] D. Mavriplis. Multigrid Solution of the Discrete Adjoint for Optimization Problems on Unstructured Meshes. *Aiaa Journal - AIAA J*, 44:42–50, January 2006.

[198] M. Maxey. Simulation Methods for Particulate Flows and Concentrated Suspensions. *Annual Review of Fluid Mechanics*, 49(1):171–193, 2017.

[199] R. Meakin and N. Suhs. Unsteady Aerodynamic Simulation of Multiple Bodies in Relative Motion. In *9th Computational Fluid Dynamics Conference*, 1989.

[200] M. Meinke, L. Schneiders, C. Günther, and W. Schröder. A Cut-Cell Method for Sharp Moving Boundaries in Cartesian Grids. *Computers & Fluids*, 85:135–142, 2013. International Workshop on Future of CFD and Aerospace Sciences.

[201] J. Melton, F. Enomoto, and M. Berger. 3D Automatic Cartesian Grid Generation for Euler Flows. In *11th Computational Fluid Dynamics Conference*, 1993.

[202] C. Merkle. Time-Accurate Unsteady Incompressible Flow Algorithms Based on Artificial Compressibility. In *8th Computational Fluid Dynamics Conference*, 1987.

[203] C. Merlin, P. Domingo, and L. Vervisch. Immersed Boundaries in Large Eddy Simulation of Compressible Flows. *Flow, Turbulence and Combustion*, 90, January 2012.

[204] M. Meyer, A. Devesa, S. Hickel, X. Hu, and N. Adams. A Conservative Immersed Interface Method for Large-Eddy Simulation of Incompressible Flows. *Journal of Computational Physics*, 229(18):6300–6317, 2010.

[205] Z. Michalewicz and B. Fogel. *How to Solve It: Modern Heuristics*. Springer, Berlin, Heidelberg, 2nd edition, 2004.

[206] C. Michler, E. van Brummelen, S. Hulshoff, and R. de Borst. The Relevance of Conservation for Stability and Accuracy of Numerical Methods for Fluid–Structure Interaction. *Computer Methods in Applied Mechanics and Engineering*, 192(37):4195–4215, 2003.

[207] R. Mittal, C. Bonilla, and H. Udaykumar. Cartesian Grid Methods for Simulating Flows with Moving Boundaries. *Computational Engineering*, 4:557–566, January 2003.

[208] 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*, 227(10):4825–4852, 2008.

[209] R. Mittal and G. Iaccarino. Immersed Boundary Methods. *Annual Review of Fluid Mechanics*, 37(1):239–261, 2005.

[210] R. Mittal, V. Seshadri, and H. Udaykumar. Flutter, Tumble and Vortex Induced Autorotation. *Theoretical and Computational Fluid Dynamics*, 17:165–170, January 2004.

[211] R. Mittal, Y. Utturkar, and H. Udaykumar. Computational Modeling and Analysis of Biomimetic Flight Mechanisms. *40th AIAA Aerospace Sciences Meeting and Exhibit*, January 2002.

[212] B. Mohammadi and O. Pironneau. Applied Shape Optimization in Fluids. *Applied Shape Optimization for Fluids*, May 2001.

[213] J. Mohd-Yosuf. Combined Immersed Boundary/B-spline Methods for Simulation of Flow in Complex Geometries. *Annu. Res. Briefs, Cent. Turbul. Res.*, page 317–28, 1997.

[214] Y. Moigne. Adaptive Mesh Refinement Sensors for Vortex Flow Simulations. January 2004.

[215] G. Morgan. Numerical Simulation of Moving Boundary Problems Related to Fracture. Master's thesis, University of Cambridge, Cambridge, 2013.

[216] E. Morishita and Sugihara M. Scroll Compressor Analytical Model. In *1984 International Compressor Engineering Conference at Purdue University*, page 487, 1984.

[217] G. Morton. A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing. *IBM Ltd.*, 1966.

[218] B. Muralidharan and S. Menon. A High-Order Adaptive Cartesian Cut-Cell Method for Simulation of Compressible Viscous Flow over Immersed Bodies. *Journal of Computational Physics*, 321:342–368, 2016.

[219] S. Murman, M. Aftosmis, M. Berger, and D. Kwak. Implicit Approaches for Moving Boundaries in a 3-D Cartesian Method. In *41st Aerospace Sciences Meeting and Exhibit*, February 2003.

[220] S. Nadarajah and A. Jameson. A Comparison of the Continuous and Discrete Adjoint Approach to Automatic Aerodynamic Optimization. November 2014.

[221] A. Nelson, M. Aftosmis, M. Nemec, and T. Pulliam. Aerodynamic Optimization of Rocket Control Surfaces Using Cartesian Methods and CAD Geometry. volume 1, June 2005.

[222] M. Nemec and M. Aftosmis. Aerodynamic Shape Optimization Using a Cartesian Adjoint Method and CAD Geometry. 2006.

[223] M. Nemec and M. Aftosmis. Adjoint Sensitivity Computations for an Embedded-Boundary Cartesian Mesh Method and CAD Geometry. volume 227, pages 2724–2742, 2008.

[224] M. Nemec, M. Aftosmis, S. Murman, and T. Pulliam. Adjoint Formulation for an Embedded-Boundary Cartesian Method. 2005.

[225] M. Nemec, M. Aftosmis, and T. Pulliam. CAD-Based Aerodynamic Design of Complex Configurations Using a Cartesian Method. *42nd AIAA Aerospace Sciences Meeting and Exhibit*, January 2004.

[226] W. Newman and R. Sproull, editors. *Principles of Interactive Computer Graphics (2nd Ed.)*. McGraw-Hill, Inc., USA, 1979.

[227] A. Nisar, N. Afzulpurkar, B. Mahaisavariya, and A. Tuantranont. MEMS-Based Micropumps in Drug Delivery and Biomedical Applications. *Sensors and Actuators B: Chemical*, 130(2):917–942, 2008.

[228] M. Nishioka and H. Sato. Measurements of Velocity Distributions in the Wake of a Circular Cylinder at Low Reynolds Numbers. *Journal of Fluid Mechanics*, 65(1):97–112, 1974.

[229] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, New York, NY, USA, 2006.

[230] S. Ojeda, H. Sun, S. Allmaras, and D. Darmofal. An Adaptive Simplex Cut-Cell Method for High-Order Discontinuous Galerkin Discretizations of Conjugate Heat Transfer Problems. *International Journal for Numerical Methods in Engineering*, 110, August 2016.

[231] G. Okubo and T. Imamura. Characteristics of Adjoint-Based Shape Optimization on Hierarchical Cartesian Mesh with Immersed Boundary Method. 2018.

[232] F. Örley, V. Pasquariello, S. Hickel, and Nikolaus A. Cut-Element Based Immersed Boundary Method for Moving Geometries in Compressible Liquid Flows with Cavitation. *Journal of Computational Physics*, 283:1–22, 2015.

[233] C. Othmer. Adjoint Methods for Car Aerodynamics. *Journal of Mathematics in Industry*, 4:6, December 2014.

[234] D. Pan and T. Shen. Computation of Incompressible Flows with Immersed Bodies by a Simple Ghost Cell Method. *International Journal for Numerical Methods in Fluids*, 60(12):1378–1401, 2009.

[235] D. Papadimitriou and K. Giannakoglou. A Continuous Adjoint Method with Objective Function Derivatives Based on Boundary Integrals, for Inviscid and Viscous Flows. *Computers & Fluids*, 36(2):325–341, 2007.

[236] V. Papageorgiou, K. Samouchos, and K. Giannakoglou. The Unsteady Continuous Adjoint Method Assisted by the Proper Generalized Decomposition Method. In *Evolutionary and Deterministic Methods for Design Optimization and Control With Applications to Industrial and Societal Problems*, pages 109–125, Cham, 2019. Springer International Publishing.

[237] E. Papoutsis-Kiachagias. *Adjoint Methods for Turbulent Flows, Applied to Shape or Topology Optimization and Robust Design.* PhD thesis, National Technical University of Athens, 2013.

[238] E. Papoutsis-Kiachagias, V. Asouti, K. Giannakoglou, K. Gkagkas, S. Shimokawa, and E. Itakura. Multi-Point Aerodynamic Shape Optimization of Cars Based on Continuous Adjoint. *Struct Multidisc Optim*, 59(2):675–694, 2019.

[239] E. Papoutsis-Kiachagias and K. Giannakoglou. Continuous Adjoint Methods for Turbulent Flows, Applied to Shape and Topology Optimization: Industrial Applications. *Archives of Computational Methods in Engineering*, 23, December 2014.

[240] N. Patankar. A Formulation for Fast Computations of Rigid Particulate Flows. *Center for Turbulence Research Annual Research Briefs*, January 2001.

[241] G. Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, 36:157–160, 1890.

[242] R. Pember, J. Bell, P. Colella, W. Curtchfield, and M. Welcome. An Adaptive Cartesian Grid Method for Unsteady Compressible Flow in Irregular Regions. *Journal of Computational Physics*, 120(2):278–304, 1995.

[243] J. Persson. Performance Mapping vs Design Parameters for Screw Compressors and other Displacement Compressor Types. *VDI Berichte*, 859, 1990.

[244] C. Peskin. Flow Patterns around Heart Valves: A Numerical Method. *Journal of Computational Physics*, 10(2):252–271, 1972.

[245] C. Peskin. The Fluid Dynamics of Heart Valves: Experimental, Theoretical, and Computational Methods. *Annual Review of Fluid Mechanics*, 14(1):235–259, 1982.

[246] J. Pilkington and S. Baden. Dynamic partitioning of non-uniform structured workloads with spacefilling curves. *IEEE Transactions on Parallel and Distributed Systems*, 7(3):288–300, 1996.

[247] O. Pironneau. *Optimal Shape Design for Elliptic Systems.* Springer, Berlin, Heidelberg, 1982.

[248] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, USA, 1988.

[249] T. Pulliam and J. Steger. Recent Improvements in Efficiency, Accuracy, and Convergence for Implicit Approximate Factorization Algorithms. February 1985.

[250] J. Purvis and J. Burkhalter. Prediction of Critical Mach Number for Store Configurations. *AIAA Journal*, 17(11):1170–1177, 1979.

[251] J. Quirk. *An Adaptive Grid Algorithm for Computational Shock Hydrodynamics.* PhD thesis, Cranfield University, 1991.

[252] J. Quirk. An Alternative to Unstructured Grids for Computing Gas Dynamic Flows around Arbitrarily Complex Two-Dimensional Bodies. *Computers & Fluids*, 23(1):125–142, 1994.

[253] K. Rahbar, S. Mahmoud, R. K. Al-Dadah, N. Moazami, and S. A. Mirhadizadeh. Review of Organic Rankine Cycle for Small-Scale Applications. *Energy Conversion and Management*, 134:135–155, 2017.

[254] N. Ramaswamy, N. Karanth, S. Kulkarni, and V. Desai. Modeling of Micropump Performance and Optimization of Diaphragm Geometry. *IJCA Proceedings on International Symposium on Devices MEMS, Intelligent Systems & Communication (ISDMISC)*, (5):14–19, 2011. Full text available.

[255] B. Re, C. Dobrzynski, and A. Guardone. An Interpolation-Free ALE Scheme for Unsteady Inviscid Flows Computations with Large Boundary Displacements over Three-Dimensional Adaptive Grids. *Journal of Computational Physics*, 340:26–54, September 2017.

[256] J. Reuther. *Aerodynamic Shape Optimization Using Control Theory.* PhD thesis, University of California Davis, 1996.

[257] J. Reuther and A. Jameson. Control Theory Based Airfoil Design for Potential Flow and a Finite Volume Discretization. February 1994.

[258] J. Reuther and A. Jameson. Aerodynamic Shape Optimization of Wing and Wing-Body Configurations Using Control Theory. February 1995.

[259] J. Reuther, A. Jameson, J. Farmer, L. Martinelli, and D. Saunders. Aerodynamic Shape Optimization of Complex Aircraft Configurations via an Adjoint Formulation. February 1996.

[260] O. Reynolds. *Papers on Mechanical and Physical Subjects: The sub-mechanics of the universe*, volume 3. Cambridge University Press, 1903.

[261] D. Rodriguez. Propulsion/Airframe Integration and Optimization on a Supersonic Business Jet. In *45th AIAA Aerospace Sciences Meeting and Exhibit*, 2007.

[262] P. Roe. Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes. *Journal of Computational Physics*, 43(2):357–372, 1981.

[263] P. Roe and J. Pike. Efficient Construction and Utilisation of Approximate Riemann Solutions. In *Proc. of the Sixth Int'l. Symposium on Computing Methods in Applied Sciences and Engineering, VI, North-Holland Publishing Co.*, page 499–518, NLD, 1985.

[264] P. Rubbert, J. Bussoletti, F. Johnson, K. Sidwell, W. Rowe, S. Samant, G. SenGupta, W. Weatherill, R. Burkhart, B. Everson, D. Young, and A. Woo. A

New Approach to the Solution of Boundary Value Problems Involving Complex Configurations. *Computational Mechanics - Advances and Trends*, pages 49–84, 1986.

[265] M. Rutkowski, W. Gryglas, J. Szumbarski, C. Leonardi, and Ł. Łaniewski Wołłk. Open-Loop Optimal Control of a Flapping Wing Using an Adjoint Lattice Boltzmann Method. *Computers & Mathematics with Applications*, 79(12):3547–3569, 2020.

[266] Y. Saad. *Iterative Methods for Sparse Linear Systems*. January 2003.

[267] R.H. Sabersky, A.J. Acosta, and E.G. Hauptmann. *Fluid Flow: A First Course in Fluid Mechanics*. Macmillan, 1989.

[268] J. Sachdev and C. Groth. A Mesh Adjustment Scheme for Embedded Boundaries. volume 2, pages 109–114, January 2006.

[269] I. Sadrehaghighi. *Essentials of CFD*. February 2021.

[270] E. Saiki and S. Biringen. Numerical simulation of a cylinder in uniform flow: Application of a virtual boundary method. *Journal of Computational Physics*, 123(2):450–465, 1996.

[271] J. Salmon, M. Warren, and G. Winckelmans. Fast Parallel Tree Codes for Gravitational and Fluid Dynamical N-Body Problems. *International Journal of Supercomputer Applications and High Performance Computing*, 8, May 1994.

[272] K. Samouchos. Development of the Computational Tools for the FlowSimulation into Scroll Turbomachines, used in Supercritical OrganicRankine Cycles. Programming of Analysis-Optimization Software forthe above Cycles. Master's thesis, National Technical University of Athens, 2013.

[273] K. Samouchos, S. Katsanoulis, and K. Giannakoglou. Unsteady Adjoint to the Cut-Cell Method Using Mesh Adaptation on GPU's. In *ECCOMAS Congress 2016, Crete, Greece*, June 2016.

[274] S. Sastry, E. Kultursay, S. Shontz, and M. Kandemir. Improved Cache Utilization and Preconditioner Efficiency through Use of a Space-Filling Curve Mesh Element- and Vertex-Reordering Technique. *Engineering with Computers*, 30:535–547, 2014.

[275] V. Schmitt and F. Charpin. Pressure Distributions on the ONERA M6 Wing at Transonic Mach Numbers. Report of the Fluid Dynamics Panel Working Group 04, AGARD AR 138, May 1979.

[276] L. Schneiders, C. Günther, J. Grimmen, M. Meinke, and W. Schroeder. *Sharp Resolution of Complex Moving Geometries Using a Multi-Cut-Cell Viscous Flow Solver*. 2015.

[277] L. Schneiders, C. Günther, M. Meinke, and W. Schröder. An Efficient Conservative Cut-Cell Method for Rigid Bodies Interacting with Viscous Compressible Flows. *Journal of Computational Physics*, 311:62–86, 2016.

[278] L. Schneiders, D. Hartmann, M. Meinke, and W. Schröder. An accurate moving boundary formulation in cut-cell methods. *Journal of Computational Physics*, 235:786–809, 2013.

[279] J. Seo and R. Mittal. A Sharp-Interface Immersed Boundary Method with Improved Mass Conservation and Reduced Spurious Pressure Oscillations. *Journal of computational physics*, 230:7347–7363, August 2011.

[280] J. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996.

[281] J. Sethian and A. Vladimirsky. Fast Methods for the Eikonal and Related Hamilton-Jacobi Equations on Unstructured Meshes. *Proceedings of the National Academy of Sciences of the United States of America*, 97:5699–703, June 2000.

[282] T. Sharp and L. Sirovich. Constructing a Continuous Parameter Range of Computational Flows. *AIAA Journal*, 27:1326–1331, 1989.

[283] S. Shin. *Reynolds-Averaged Navier-Stokes Computation of Tip Clearance Flow in a Compressor Cascade Using an Unstructured Grid*. PhD thesis, Virginia Polytechnic Institute and State University, 2001.

[284] A. Shinn, M. Goodwin, and S. Vanka. Immersed Boundary Computations of Shear- and Buoyancy-Driven Flows in Complex Enclosures. *International Journal of Heat and Mass Transfer*, 52(17):4082–4089, 2009. Special Issue Honoring Professor D. Brian Spalding.

[285] K. Sindhya, K. Miettinen, and K. Deb. A Hybrid Framework for Evolutionary Multi-Objective Optimization. *IEEE Transactions on Evolutionary Computation*, 17(4):495–511, 2013.

[286] J. Slater. `https://www.grc.nasa.gov/WWW/wind/valid/m6wing/m6wing01/m6wing01.html`.

[287] S. Smolyak. Quadrature and Interpolation Formulas for Tensor Products of Certain Classes of Functions. *Dokl. Akad. Nauk SSSR*, 148:1042–1045, 1963.

[288] L. Songjing, J. Liu, and D. Jiang. Dynamic Characterization of a Valveless Micropump Considering Entrapped Gas Bubbles. *Journal of Heat Transfer*, 135:091403, 09 2013.

[289] F. Sotiropoulos and X. Yang. Immersed Boundary Methods for Simulating Fluid–Structure Interaction. *Progress in Aerospace Sciences*, 65:1–21, 2014.

[290] J. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control.* John Wiley & Sons, 2003.

[291] J. Steger and R. Warming. Flux Vector Splitting of the Inviscid Gasdynamic Equations with Application to Finite-Difference Methods. *Journal of Computational Physics*, 40(2):263–293, 1981.

[292] E. Stemme and Stemme G. A Valveless Diffuser/Nozzle-Based Fluid Pump. *Sensors and Actuators A: Physical*, 39(2):159–167, 1993.

[293] R. Stevens, A. Lehar, and F. Preston. Manipulation and Presentation of Multidimensional Image Data Using the Peano Scan. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(5):520–526, 1983.

[294] G. Stewart. On the Early History of the Singular Value Decomposition. *SIAM Rev.*, 35:551–566, 1993.

[295] S. Su and C. Lai, M.and Lin. An immersed boundary technique for simulating complex flows with rigid boundary. *Computers & Fluids*, 36(2):313–324, 2007.

[296] I. Sutherland and G. Hodgman. Reentrant Polygon Clipping. *Commun. ACM*, 17(1):32–42, January 1974.

[297] R.C. Swanson and S. Langer. Steady-state Laminar Flow Solutions for NACA 0012 Airfoil. *Computers & Fluids*, 126:102 – 128, 2016.

[298] A. Syrakos, S. Varchanis, Y. Dimakopoulos, A. Goulas, and J. Tsamopoulos. A Critical Analysis of some Popular Methods for the Discretisation of the Gradient Operator in Finite Volume Methods. *Physics of Fluids*, 29(12):127103, 2017.

[299] H. Takami and H. Keller. Steady Two–Dimensional Viscous Flow of an Incompressible Fluid past a Circular Cylinder. *The Physics of Fluids*, 12(12):II–51–II–56, 1969.

[300] INRIA Sophia-Antipolis. TAPENADE. `https://www-sop.inria.fr/tropics/tapenade.html`.

[301] A. M. K. P. Taylor, J. H. Whitelaw, and M. Yianneskis. Developing Flow in S-shaped Ducts. 1: Square Cross-Section Duct. Final Report Imperial Coll. of Science and Technology, May 1982.

[302] L. Taylor and D. Whitfield. Unsteady Three-dimensional Incompressible Euler and Navier-Stokes Solver for Stationary and Dynamic Grids. In *22nd Fluid Dynamics, Plasma Dynamics and Lasers Conference*, 1991.

[303] B. F. Tchanche, G. Lambrinos, A. Frangoudakis, and G. Papadakis. Low-Grade Heat Conversion into Power Using Organic Rankine Cycles – A Review of Various Applications. *Renewable and Sustainable Energy Reviews*, 15(8):3963–3979, 2011.

[304] H. Terashima and G. Tryggvason. A Front-Tracking/Ghost-Fluid Method for Fluid Interfaces in Compressible Flows. *Journal of Computational Physics*, 228(11):4012–4037, 2009.

[305] J. Thompson, B. Soni, and N. Weatherill. *Handbook of Grid Generation*. CRC Press., first edition, 1998.

[306] E. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*. January 2009.

[307] E. Toro, M. Spruce, and W. Speares. Restoration of the Contact Surface in the HLL-Riemann Solver. *Shock Waves*, 4:25–34, 1994.

[308] C. Towne. Computation of Viscous Flow in Curved Ducts and Comparison with Experimental Data. In *22nd Aerospace Sciences Meeting*.

[309] X. Trompoukis. *Solving Aerodynamic-Aeroelastic Problems on Graphics Processing Units*. PhD thesis, National Technical University of Athens, 2012.

[310] Y. Tseng and J.r Ferzige. A Ghost-Cell Immersed Boundary Method for Flow in Complex Geometry. *Journal of Computational Physics*, 192(2):593–623, 2003.

[311] K. Tsiakas. *Development of Shape Parameterization Techniques, a Flow Solver and its Adjoint, for Optimization on GPUs. Turbomachinery and External Aerodynamics Applications.* PhD thesis, National Technical University of Athens, 2019.

[312] E. Turkel. Acceleration to Steady State for the Euler Equations. February 1985.

[313] E. Turkel. Preconditioned Methods for Solving the Incompressible and Low Speed Compressible Equations. *Journal of Computational Physics*, 72(2):277–298, 1987.

[314] E. Turkel. Review of Preconditioning Methods for Fluid Dynamics. *Applied Numerical Mathematics*, 12:257–284, October 1992.

[315] Turkel, E. Preconditioning Techniques in Computational Fluid Dynamics. *Annual Review of Fluid Mechanics*, 31(1):385–416, 1999.

[316] M. Uhlmann. An Immersed Boundary Method with Direct Forcing for the Simulation of Particulate Flows. *Journal of Computational Physics*, 209(2):448–476, 2005.

[317] Y. Utturkar, R. Mittal, P. Rampunggoon, and L. Cattafesta. Sensitivity of Synthetic Jets to the Design of the Jet Cavity. January 2002.

[318] B. van Leer. Towards the Ultimate Conservative Difference Scheme. V. A Second-Order Sequel to Godunov's Method. *Journal of Computational Physics*, 32(1):101–136, 1979.

[319] B. van Leer, W. Lee, and P. Roe. Characteristic Time-Stepping or Local Preconditioning of the Euler Equations. In *10th Computational Fluid Dynamics Conference*, 1991.

[320] H.T.G. van Lintel, F.C.M. van De Pol, and S. Bouwstra. A Piezoelectric Micropump Based on Micromachining of Silicon. *Sensors and Actuators*, 15(2):153–167, 1988.

[321] V. Venkatakrishnan. On the Accuracy of Limiters and Convergence to Steady State Solutions. In *31st Aerospace Sciences Meeting*, 1993.

[322] R. Verzicco, J. Mohd-Yusof, P. Orlandi, and D. Haworth. LES in Complex Geometries Using Boundary Body Forces. *AIAA J.*, 38:427–33, 2000.

[323] C. Vezyris, E. Papoutsis-Kiachagias, and K. Giannakoglou. On the Incremental Singular Value Decomposition Method to Support Unsteady Adjoint-Based Optimization. *International Journal for Numerical Methods in Fluids*, 91(7):315–331, 2019.

[324] R. von Flatern. https://www.slb.com/-/media/files/oilfield-review/defining-esp.ashx.

[325] P.-Y. Vrionis. *Shape and Topology Optimization using the Cut-Cell Method and its Continuous Adjoint for Single– and Two–phase Turbulent flows, in a Multiprocessor Environment.* PhD thesis, National Technical University of Athens. In progress.

[326] P.-Y. Vrionis, K. Samouchos, and K. Giannakoglou. The Continuous Adjoint Cut-Cell Method for Shape Optimization in Cavitating Flows. *Computers & Fluids*, 224:104974, 04 2021.

[327] P.-Y. Vrionis, K. Samouchos, and K. Giannakoglou. Topology Optimization in Fluid Mechanics Using Continuous Adjoint and the Cut-Cell Method. *Computers & Mathematics with Applications*, 97:286–297, 2021.

[328] A. Walther. Getting Started with ADOL-C. *Combinatorial Scientific Computing*, January 2009.

[329] Q. Wang, P. Moin, and G. Iaccarino. Minimal Repetition Dynamic Checkpointing Algorithm for Unsteady Adjoint Calculation. *SIAM Journal on Scientific Computing*, 31(4):2549–2567, 2009.

[330] Z. Wang. A Quadtree-Based Adaptive Cartesian/Quad Grid Flow Solver for Navier-Stokes Equations. *Computers & Fluids*, 27(4):529–549, 1998.

[331] B. Wedan and J. South. A Method for Solving the Transonic Full-Potential Equation for General Configurations. 1983.

[332] M. Wintzer, M. Nemec, and M. Aftosmis. Adjoint-Based Adaptive Mesh Refinement for Sonic Boom Prediction. 2008.

[333] G. Xiao and G. Liu. Computer Simulation for Transient Flow in Oil-free Scroll Compresso. *International Journal of Control and Automation*, 7(9), 2014.

[334] Z. Xie and T. Stoesser. A Three-Dimensional Cartesian Cut-Cell/Volume-of-Fluid Method for Two-Phase Flows with Moving Bodies. *Journal of Computational Physics*, 416:109536, 2020.

[335] D. Xiu and G. Karniadakis. The Wiener–Askey Polynomial Chaos for Stochastic Differential Equations. *SIAM Journal on Scientific Computing*, 24(2):619–644, 2002.

[336] M. Xu and M. Wei. Using Adjoint-Based Approach to Study Flapping Wings, 2013.

[337] T. Yanagisawa T.and T Shimizu. Leakage Losses with a Rolling Piston Type Rotary Compressor. I. Radical Clearance on the Rolling Piston. *International Journal of Refrigeration*, 8(2):75–84, 1985.

[338] G. Yang, D. Causon, D. Ingram, R. Saunders, and P. Battent. A Cartesian Cut Cell Method for Compressible Flows Part B: Moving Body Problems. *The Aeronautical Journal (1968)*, 101(1002):57–65, 1997.

[339] J. Yang and E. Balaras. An Embedded-Boundary Formulation for Large-Eddy Simulation of Turbulent Flows Interacting with Moving Boundaries. *Journal of Computational Physics*, 215(1):12–40, 2006.

[340] 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*, 156(2):209–240, 1999.

[341] Y. Yuan. Recent Advances in Trust Region Algorithms. *Mathematical Programming*, 151, June 2015.

[342] R. Zanelli and D. Favrat. Experimental Investigation of a Hermetic Scroll Expander–Generator. In *Proceedings of the International Compressor Engineering Conference at Purdue*, 01 1994.

[343] D. Zeeuw and K. Powell. An Adaptively Refined Cartesian Mesh Solver for the Euler Equations. *Journal of Computational Physics*, 104:56–68, 1993.

[344] X. Zhang, P. Theissen, and J. Schlüter. A Lagrangian Method for the Treatment of Freshly Cleared Cells in Immersed Boundary Techniques. *International Journal of Computational Fluid Dynamics*, 23(9):667–670, 2009.

[345] H. Zhao. A Fast Sweeping Method for Eikonal Equations. *Mathematics of Computation*, 74(250):603–627, 2005.

[346] J. Zhu, H. Banjar, Z. Xia, and H. Zhang. CFD Simulation and Experimental Study of Oil Viscosity Effect on Multi-Stage Electrical Submersible Pump (ESP) Performance. *Journal of Petroleum Science and Engineering*, 146:735–745, 2016.

[347] J. Zhu, H. Zhu, J. Zhang, and H. Zhang. A Numerical Study on Flow Patterns inside an Electrical Submersible Pump (ESP) and Comparison with Visualization Experiments. *Journal of Petroleum Science and Engineering*, 173:339–350, 2019.

[348] W. Ziniu, X. Yizhe, W. Wenbin, and H. Ruifeng. Review of Shock Wave Detection Method in CFD Post-Pocessing. *Chinese Journal of Aeronautics*, 26(3):501–513, 2013.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Εργαστήριο Θερμικών Στροβιλομηχανών
Μονάδα Παράλληλης Υπολογιστικής
Ρευστοδυναμικής & Βελτιστοποίησης

Η Μέθοδος των Τεμνόμενων Κυψελών για την Πρόλεξη 2Δ/3Δ Ροών σε Σύνθετες Γεωμετρίες και τη Βελτιστοποίηση Μορφής με τη Συζυγή Μέθοδο

Διδακτορική Διατριβή
(Εκτεταμένη Περίληψη)

Κωνσταντίνος Δ. Σαμούχος

Επιβλέπων: Κυριάκος Χ. Γιαννάκογλου
Καθηγητής ΕΜΠ

Αθήνα, 2022

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Εργαστήριο Θερμικών Στροβιλομηχανών
Μονάδα Παράλληλης Υπολογιστικής
Ρευστοδυναμικής & Βελτιστοποίησης

Η Μέθοδος των Τεμνόμενων Κυψελών για την Πρόλεξη 2Δ/3Δ Ροών σε Σύνθετες Γεωμετρίες και τη Βελτιστοποίηση Μορφής με τη Συζυγή Μέθοδο

Διδακτορική Διατριβή

Κωνσταντίνος Δ. Σαμούχος

Εξεταστική Επιτροπή:

1. Κυριάκος Γιαννάκογλου* (Επιβλέπων), Καθηγητής, ΕΜΠ, Σχολή Μηχανολόγων Μηχανικών
2. Ιωάννης Αναγνωστόπουλος*, Καθηγητής, ΕΜΠ, Σχολή Μηχανολόγων Μηχανικών
3. Σπυρίδων Βουτσινάς*, Καθηγητής, ΕΜΠ, Σχολή Μηχανολόγων Μηχανικών
4. Κωνσταντίνος Μαθιουδάκης, Καθηγητής, ΕΜΠ, Σχολή Μηχανολόγων Μηχανικών
5. Κωνσταντίνος Μπελιμπασάκης, Καθηγητής, ΕΜΠ, Σχολή Ναυπηγών Μηχανολόγων Μηχανικών
6. Δημήτριος Μπούρης, Αναπληρωτής Καθηγητής, ΕΜΠ, Σχολή Μηχανολόγων Μηχανικών
7. Γεώργιος Παπαδάκης, Επίκουρος Καθηγητής, ΕΜΠ, Σχολή Ναυπηγών Μηχανολόγων Μηχανικών

* Μέλος της Συμβουλευτικής Επιτροπής

Αθήνα, 2022

Περίληψη

Η διδακτορική διατριβή αναπτύσσει εκ του μηδενός ένα αυτοτελές σύνολο εργαλείων με σκοπό τη ρευστοδυναμική ανάλυση και βελτιστοποίηση μορφής σε εφαρμογές της μηχανικής των ρευστών. Η διατριβή στηρίζεται στη μέθοδο των τεμνόμενων κυψελών, προκειμένου να άρει τις δυσκολίες που μπορεί να εισάγει η γένεση πλέγματος στην ανάλυση και σχεδιασμό μηχανολογικών προϊόντων πολύπλοκης μορφής. Η μέθοδος αυτή χρησιμοποιεί καρτεσιανά πλέγματα τα οποία καλύπτουν ολόκληρο το υπολογιστικό χωρίο συμπεριλαμβανομένου του τμήματος που καταλαμβάνεται από τα στερεά σώματα. Κατά την επίλυση της ροής, το στερεό μέρος του πλέγματος δεν χρησιμοποιείται και, συνεπώς, αποκόπτεται εισάγοντας την έννοια των τεμνόμενων κυψελών. Πρόκειται για ορθογώνιες παραλληλόγραμμες (2Δ) ή παραλληλεπίπεδες (3Δ) κυψέλες που τέμνονται από το όριο της γεωμετρίας και τα οποία ανασχηματίζονται αποβάλλοντας το στερεό τους τμήμα.

Η μέθοδος των τεμνόμενων κυψελών παρουσιάζει πολλά πλεονεκτήματα με βασικότερο αυτό της γρήγορης και αυτόματης πλεγματοποίησης που παραμένει ανεξάρτητη από την πολυπλοκότητα του υπολογιστικού χωρίου. Επιπλέον, η χρήση τους είναι πλεονεκτική σε εφαρμογές που περιλαμβάνουν κινούμενα στερεά σώματα, καθώς αυτά μπορούν να κινούνται ελεύθερα πάνω από το απαραμόρφωτο καρτεσιανό πλέγμα. Έτσι, αποφεύγεται εκ νέου πλεγματοποίηση ή χρήση εργαλείων παραμόρφωσης πλεγμάτων, των οποίων η αποτελεσματικότητα είναι αμφίβολη σε περιπτώσεις έντονης μετατόπισης των στερεών ορίων. Όσον αφορά τη βελτιστοποίηση μορφής, η χρήση καρτεσιανών πλεγμάτων κρίνεται ιδιαίτερα επωφελής. Σε αυτά τα προβλήματα, η αποφυγή διαρκούς γένεσης ή παραμόρφωσης οριόδετων πλεγμάτων και των εγγενών δυσκολιών τους επιτρέπει την αναζήτηση βέλτιστων λύσεων επιτρέποντας την ανάδειξη πιο εξεζητημένων γεωμετρικών σχημάτων.

Αρχικά, η διατριβή παρουσιάζει μεθόδους αυτόματης και γρήγορης γένεσης πλέγματος για την υποστήριξη της μεθόδου των τεμνόμενων κυψελών, με χαμηλές απαιτήσεις σε υπολογιστική μνήμη. Η υψηλή ποιότητα των καρτεσιανών πλεγμάτων εξασφαλίζεται με την ομαλή μεταβολή της πυκνότητάς τους κοντά στα στερεά όρια και σε περιοχές που λαμβάνουν χώρα ροϊκά φαινόμενα ιδιαίτερου ενδιαφέροντος. Στο ίδιο πλαίσιο, εισάγονται νέοι αλγόριθμοι, ικανοί να υπολογίσουν την ακριβή τομή των καρτεσιανών κυψελών με τα στερεά όρια και να κατασκευάσουν τις αντίστοιχες τεμνόμενες κυψέλες καλύπτοντας όλο το φάσμα των πιθανών γεωμετρικών υποπεριπτώσεων. Επιπλέον, αποφεύγονται αριθμητικές αστάθειες κατά την αριθμητική επίλυση της ροής μέσω της

συνένωσης γειτονικών κυψελών αρκετά διαφορετικού μεγέθους. Ακόμα, παρουσιάζονται μέθοδοι γρήγορης ανίχνευσης γειτονικών κυψελών, αρίθμησης κόμβων και εδρών, καθώς και τεχνικές διάσπασης του πλέγματος σε επιμέρους τμήματα με σκοπό την επίλυση της ροής σε πολυεπεξεργαστικό περιβάλλον.

Στη συνέχεια, παρουσιάζονται τα λογισμικά αριθμητικής επίλυσης συμπιεστής και ασυμπίεστης ροής όπου, για ασυμπίεστες ροές, εφαρμόζεται η τεχνική της ψευδοσυμπιεστότητας. Το προτεινόμενο σχήμα διακριτοποίησης επωφελείται από την ιδιαίτερη δομή του καρτεσιανού πλέγματος και βασίζεται σε μια κεντροκυψελική διατύπωση πεπερασμένων όγκων, εφαρμόζοντας το σχήμα MUSCL και την κατά Roe προσεγγιστική λύση του προβλήματος Riemann. Σε περιπτώσεις κινούμενων στερεών ορίων, η πύκνωση του πλέγματος μεταβάλλεται με το χρόνο ακολουθώντας την κίνησή τους. Εφαρμόζονται καινοτόμες μέθοδοι μεταφοράς του πεδίου ροής στο πλέγμα της επόμενης χρονικής στιγμής καθώς και τεχνικές χειρισμού των καρτεσιανών κυψελών, τα οποία μεταπηδούν από τη στερεή στη ρευστή περιοχή του πλέγματος και αντίστροφα.

Η ακρίβεια του αναπτυχθέντος λογισμικού πιστοποιείται μέσω της σύγκρισης των υπολογισμών του με αντίστοιχες πειραματικές μετρήσεις σε εφαρμογές που καλύπτουν ένα ευρύ φάσμα περιπτώσεων εσωτερικής και εξωτερικής, ατριβούς ή στρωτής ροής. Επιπλέον, παρουσιάζονται βιομηχανικές εφαρμογές που αναδεικνύουν τη χρηστικότητα και αποτελεσματικότητα της μεθόδου. Αρχικά, μελετάται η ροή μέσα σε μια μηχανή κύλισης, κάτι το οποίο σπανίζει στη βιβλιογραφία. Στη συνέχεια, εξετάζεται η ρευστοδυναμική συμπεριφορά μιας διαφραγματικής αντλίας χωρίς βαλβίδες, όπου παρά τον μεγάλο αριθμό κυψελών που σαρώνονται από το στερεό όριο κάθε χρονική στιγμή, το λογισμικό εγγυάται τη διατήρηση της μάζας. Τέλος, προσομοιώνεται η ροή σε βαθμίδα αντλίας εξόρυξης πετρελαίου μικτού τύπου όπου η μέθοδος των τεμνόμενων κυψελών προτείνεται ως εναλλακτικός τρόπος αντιμετώπισης του προβλήματος αλληλεπίδρασης της κινούμενης και της ακίνητης πτερύγωσης.

Σε προβλήματα βελτιστοποίησης μορφής, εφαρμόζονται η συνεχής και διακριτή συζυγής διατύπωση για τον υπολογισμό της κλίσης της συνάρτησης στόχου. Οι μέθοδοι αυτές είναι ιδιαίτερα προσφιλείς λόγω του ιδιαίτερα χαμηλού υπολογιστικού τους κόστους, το οποίο παραμένει ανεξάρτητο του πλήθους των μεταβλητών σχεδιασμού που ελέγχουν το σχήμα της εκάστοτε γεωμετρίας. Αξίζει να σημειωθεί, ότι η μαθηματική διατύπωση των μεθόδων αυτών και ανάπτυξη του αντίστοιχου λογισμικού για συνεκτικές ή/και μη-μόνιμες ροές σε πλέγματα τεμνόμενων κυψελών παρουσιάζεται για πρώτη φορά στη βιβλιογραφία. Όσον αφορά τη συνεχή διατύπωση, πραγματοποιείται διερεύνηση των τρόπων διακριτοποίησης των συζυγών εξισώσεων και προτείνονται τα συζυγή ισοδύνα-

μα των σχημάτων FVS, HLLC και Roe.

Η ανάπτυξη του λογισμικού της διακριτής συζυγούς μεθόδου βασίζεται στη δια χειρός διαφόριση του αντίστοιχου λογισμικού επίλυσης της συμπιεστής και ασυμπίεστης ροής. Ιδιαίτερη έμφαση δίδεται στο σωστό χειρισμό του χρονικού όρου, κάτι που συνεπάγεται τη διαφόριση των αλγορίθμων που είναι υπεύθυνοι για τη σωστή μεταφορά του στιγμιαίου πεδίου ροής στο πλέγμα της επόμενης χρονικής στιγμής. Στη συνέχεια, το λογισμικό εφαρμόζεται σε προβλήματα βιομηχανικού σχεδιασμού, όπως η ελαχιστοποίηση των απωλειών ολικής πίεσης ενός αγωγού, η μεγιστοποίηση της άνωσης πτέρυγας και η ελαχιστοποίηση της εφαπτομενικής ταχύτητας στην έξοδο βαθμίδας αντλίας εξόρυξης πετρελαίου. Τέλος, χρησιμοποιείται στη βελτιστοποίηση πολλών στόχων υπό περιορισμούς μιας διαφραγματικής αντλίας. Σε όλες τις περιπτώσεις, το λογισμικό παρήγαγε γεωμετρικά σχήματα αυξημένης απόδοσης, επιβεβαιώνοντας την αποτελεσματικότητα της αναπτυχθείσας μεθόδου.

Λέξεις κλειδιά: Εξισώσεις Navier-Stokes, Υπολογιστική Ρευστοδυναμική , Μέθοδος Τεμνόμενων Κυψελών Συμπιεστή Ροή, Ασυμπίεστη Ροή, Μη-Μόνιμη Ροή, Βελτιστοποίηση Μορφής, Συνεχής Συζυγής Μέθοδος, Διακριτή Συζυγής Μέθοδος

Ακρωνύμια

ΜΔΕ	Μερική Διαφορική Εξίσωση
ΜΕΣ	Μέθοδος Εμβαπτιζόμενων Σωμάτων
ΜΤΚ	Μέθοδος Τεμνόμενων Κυψελών
ΠΔ	Πεπερασμένες Διαφορές
ΥΡ	Υπολογιστική Ρευστοδυναμική
EASY	Evolutionary Algorithms SYstem
ESP	Electrical Submersible Pump
PCE	Polynomial Chaos Expansion
PCA	Principal Component Analysis

Περιεχόμενα

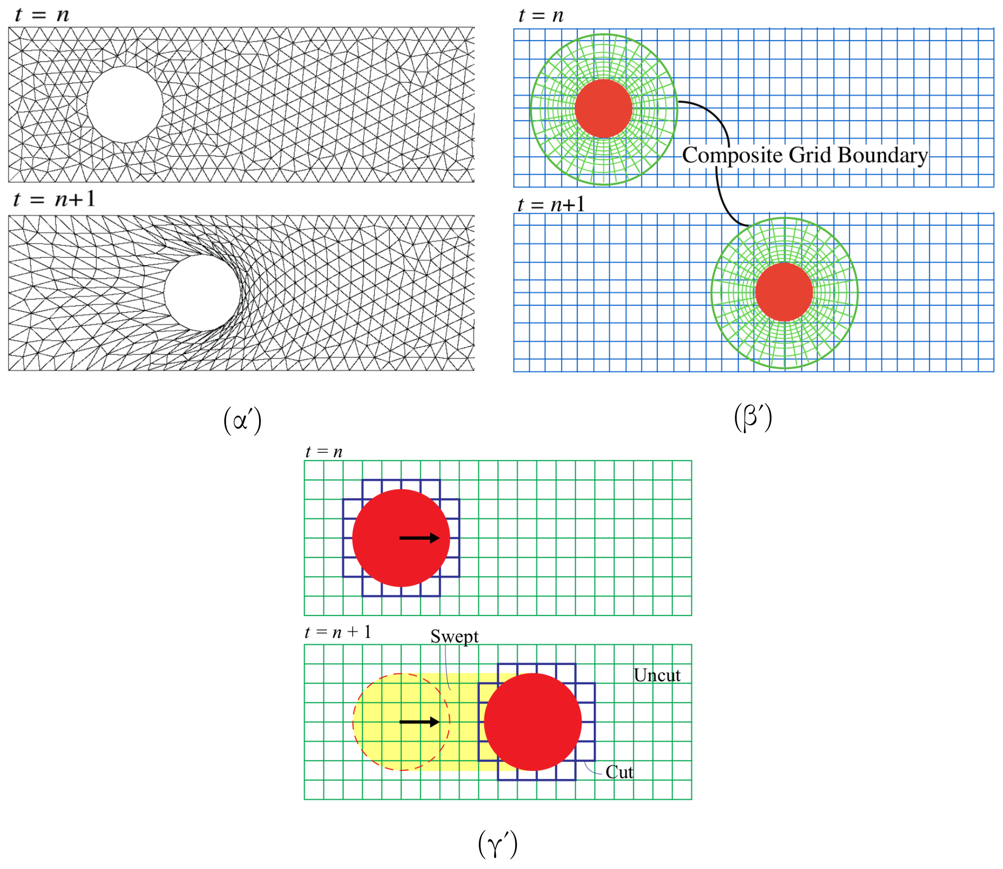
Κεφάλαιο 1

Εισαγωγή

Ο επιστημονικός τομέας της Υπολογιστικής Ρευστοδυναμικής (ΥΡ) έχει γνωρίσει σημαντική ανάπτυξη κατά τη διάρκεια των τελευταίων δεκαετιών λόγω της αύξησης της ισχύος των σύγχρονων υπολογιστικών συστημάτων και της ωρίμανσης των αριθμητικών μεθόδων επίλυσης συστημάτων μη-γραμμικών Μερικών Διαφορικών Εξισώσεων (ΜΔΕ). Η εξέλιξη αυτή επέτρεψε τη χρήση λογισμικών ΥΡ στη ρευστοδυναμική ανάλυση και βελτιστοποίηση σε όλο και πιο απαιτητικές βιομηχανικές εφαρμογές. Η συνεπαγόμενη αύξηση της ζήτησης για αξιόπιστα και αυτοματοποιημένα λογισμικά που να μπορούν να ανταποκριθούν σε προβλήματα, τα οποία περιλαμβάνουν περίπλοκες κινούμενες ή μη γεωμετρίες, έθεσε τη γένεση υπολογιστικού πλέγματος ως ένα από τα κυριότερα προβλήματα προς επίλυση ή βελτίωση.

Μια από τις πιο ελπιδοφόρες τεχνικές αυτόματης πλεγματοποίησης περίπλοκων χωρίων είναι η Μέθοδος των Εμβαπτιζόμενων Σωμάτων (ΜΕΣ). Σύμφωνα με αυτήν, το υπολογιστικό χωρίο καλύπτεται εξ ολοκλήρου με ένα εύκολα κατασκευάσιμο καρτεσιανό πλέγμα, το οποίο εκτείνεται τόσο στην περιοχή της ροής όσο και στην περιοχή που καταλαμβάνεται από στερεά σώματα. Συνεπώς, αντίθετα με τις συνήθεις πρακτικές, το πλέγμα δεν είναι προσδεδεμένο στην επιφάνεια της γεωμετρίας. Έτσι, το πρόβλημα πλεγματοποίησης αντικαθίσταται από το πρόβλημα διαχείρισης των εμβαπτιζόμενων στερεών ορίων. Η διαφορά των καρτεσιανών πλεγμάτων από ένα οριόδετο πλέγμα γίνεται κατανοητή μέσω του σχήματος 1.2, όπου οι δύο τεχνικές εφαρμόζονται για πρόλεξη της ροής γύρω από μια αεροτομή.

Ένα μεγάλο πλεονέκτημα της μεθόδου αυτής έγκειται στην αντιμετώπιση κινούμενων γεωμετριών, οι οποίες μπορούν να μετακινούνται πάνω στο Καρτεσιανό πλέγμα χωρίς να το παραμορφώνουν. Αυτό επιτρέπει τη διαχείριση έντονα κινούμενων στερεών σωμάτων χωρίς να επηρεάζεται η ποιότητα του πλέγματος. Μια αντιπαραβολή μεταξύ της μεθόδου ΜΕΣ και της παραδοσιακής τεχνικής προσδεδεμένων στο όριο πλεγμάτων φαίνεται στο σχήμα 1.1. Στο σχήμα παρουσιάζεται και η μέθοδος των επικαλυπτόμενων πλεγμάτων [5], η οποία, ενώ μπορεί εξ ίσου καλά να αντιμετωπίσει έντονες μεταβολές του ορίου, δυσκολεύεται να επιβάλει τους νόμους διατήρησης μάζας, ορμής και ενέργειας χωρίς τη χρήση περίπλοκων σχημάτων μεταφοράς της πληροφορίας από το ένα πλέγμα στο άλλο.



Σχήμα 1.1: (α΄) Η κίνηση του κυλίνδρου οδηγεί στην έντονη παραμόρφωση του μη δομημένου οριόδετου πλέγματος, οδηγώντας στη δημιουργία μη αποδεκτών αναδιπλωμένων κυψελών. (β΄) Ένα οριόδετο πλέγμα (πράσινο) κινείται μαζί με τον κύλινδρο, στον οποίο είναι προσδεδεμένο, πάνω σε ένα απαραμόρφωτο καρτεσιανό πλέγμα (μπλε). (γ΄) Ο κύλινδρος κινείται πάνω σε ένα καρτεσιανό πλέγμα σαρώνοντας κυψέλες, οι οποίες μεταπηδούν από τη στερεά στη ρευστή περιοχή του πλέγματος και αντίστροφα. Ανατύπωση σχημάτων από [31].

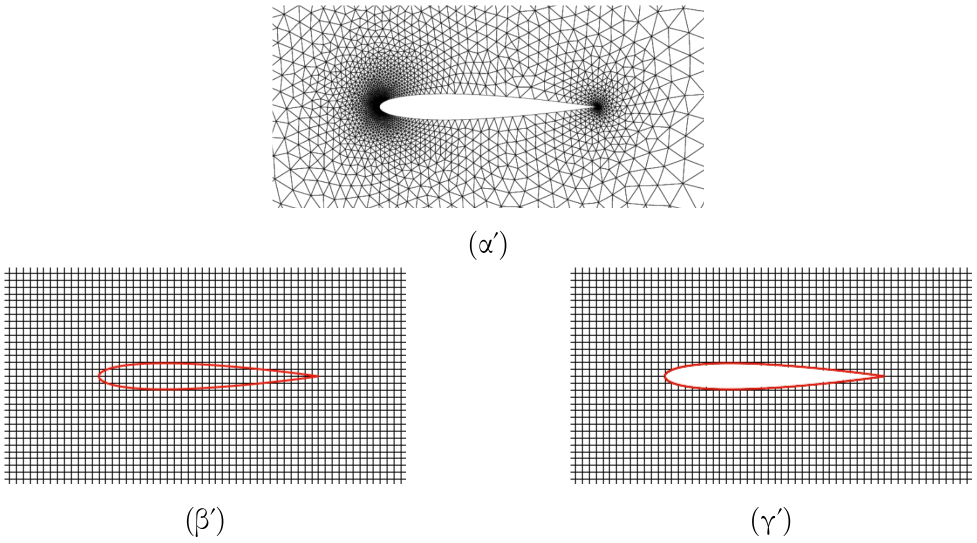
1.1 Η Μέθοδος των Τεμνόμενων Κυψελών

Η μεγαλύτερη δυσκολία που αντιμετωπίζουν οι ΜΕΣ είναι η επιβολή των συνθηκών μη-εισχώρησης και μη-ολίσθησης κατά μήκος των στερεών τοιχωμάτων. Πολλές τεχνικές έχουν προταθεί για την επίλυση του προβλήματος, [30]. Μια από τις πιο ακριβείς μεθόδους της οικογένειας των ΜΕΣ είναι η μέθοδος των τεμνόμενων κυψελών (ΜΤΚ), σύμφωνα με την οποία το τμήμα του πλέγματος που καταλαμβάνεται από τη γεωμετρία αποκόπτεται δημιουργώντας τις λεγόμενες τεμνόμενες κυψέλες.

Συγκεκριμένα, πρόκειται για κυψέλες του καρτεσιανού πλέγματος, όπου ένα τμήμα τους βρίσκεται στην περιοχή της ροής και το άλλο καλύπτεται από το στερεό σώμα, το οποίο κι αποβάλλουν σχηματίζοντας πολυεδρικούς όγκους ελέγχου. Το σχήμα 1.2 συγκρίνει ένα οριόδετο πλέγμα, ένα καρτεσιανό πλέγμα όπως χρησιμοποιείται στις περισσότερες ΜΕΣ και το ίδιο πλέγμα, αφότου έχει αποκοπεί το τμήμα του στη στερεή περιοχή. Συνεπώς, η ΜΤΚ μπορεί να θεωρηθεί ως μια επέκταση των οριόδετων πλεγμάτων διατηρώντας παράλληλα και τα πλεονεκτήματα της χρήσης καρτεσιανών πλεγμάτων.

Η ΜΤΚ προτάθηκε πρώτη φορά στη βιβλιογραφία το 1979 από τους Purvis et al. [35] και στη συνέχεια εφαρμόστηκε από τους Wedan et al. [47] το 1983 για εξισώσεις τύπου δυναμικού. Αργότερα, το 1986, οι Clarke et al. [9] εφάρμοσαν την ΜΤΚ στις 2Δ εξισώσεις Euler και οι Gaffney et al. [16] επέκτειναν τη μέθοδο σε 3Δ εφαρμογές. Στα τέλη της δεκαετίας του ‘80 εμφανίστηκαν οι πρώτες τεχνικές προσαρμογής του καρτεσιανού πλέγματος τόσο στη διεπιφάνεια μεταξύ στερεού και ρευστού όσο και σε ροϊκά φαινόμενα όπως τα κύματα κρούσης, που απαιτούν υψηλότερης πυκνότητας πλέγματα [37]. Η πρώτη εφαρμογή σε συνεκτικές ροές έγινε από τους Quirk [36] και Coirier et al. [10] για 2Δ ασυμπίεστα ρευστά και από τους Hartmann et al. [18] για 3Δ συμπιεστά ρευστά. Τέλος, η ΜΤΚ έχει χρησιμοποιηθεί για την επίλυση τυρβωδών ροών [6].

Η πρόλεξη μη-μόνιμων ροών γύρω από κινούμενα στερεά όρια κατέδειξε νέα προβλήματα που πρέπει να αντιμετωπιστούν, προκειμένου να διατηρηθεί η ακρίβεια της μεθόδου. Η κυριότερη δυσκολία έγκειται στην ξαφνική εμφάνιση ή εξαφάνιση κυψελών από το υπολογιστικό χωρίο λόγω της σάρωσής τους από τα κινούμενα σώματα. Η διακριτοποίηση των εξισώσεων ροής σε αυτές τις κυψέλες χρειάζεται προσοχή, έτσι ώστε να αποφευχθούν τεχνητές πηγές ή καταβόθρες μάζας, ορμής ή ενέργειας. Έχουν προταθεί διάφορες μέθοδοι για την αντιμετώπιση αυτού του φαινομένου, όπως ο διαμοιρασμός της απώλειας ή περίσσειας μάζας σε γειτονικές κυψέλες [40], η συνένωση των εμφανιζόμενων ή εξαφανιζόμενων κυψελών με γειτονικές τους [4].



Σχήμα 1.2: (α′) Οριόδετο μη-δομημένο πλέγμα γύρω από μεμονωμένη αεροτομή. (β′) Εμβαπτιζόμενη αεροτομή εντός καρτεσιανού πλέγματος. (γ′) Το στερεό τμήμα του καρτεσιανού πλέγματος έχει αποκοπεί σχηματίζοντας το πλέγμα της ΜΤΚ.

Παρά την ανάπτυξη διάφορων παραλλαγών της ΜΤΚ και της εφαρμογής τους σε μεγάλο φάσμα προβλημάτων, παραμένει ακόμα η ανάγκη για ακόμα ακριβέστερες τεχνικές τόσο σε μόνιμες όσο και σε μη-μόνιμες ροές. Η διατριβή συμβάλλει σε αυτήν την προσπάθεια, προτείνοντας αλγορίθμους αυτόματης γένεσης πλέγματος και κατασκευής των τεμνόμενων κυψελών σε περίπλοκα χωρία, που ελαχιστοποιούν την αλληλεπίδραση λογισμικού-χρήστη. Για εφαρμογές κινούμενων γεωμετριών, αναπτύσσεται μια νέα μέθοδος που επιτρέπει τη διατήρηση της μάζας, ορμής και ενέργειας ακόμα και κατά την έντονη μετατόπιση των στερεών σωμάτων. Το προκύπτον λογισμικό πιστοποιείται σε εφαρμογές συμπιεστού ή ασυμπίεστου ρευστού, εσωτερικής ή εξωτερικής ρευστοδυναμικής. Τέλος, εφαρμόζεται σε μια σειρά από βιομηχανικές εφαρμογές αναδεικνύοντας τα πλεονεκτήματα της ΜΤΚ έναντι των παραδοσιακών τεχνικών βασισμένων σε οριόδετα πλέγματα.

1.2 Αεροδυναμική Βελτιστοποίηση σε Καρτεσιανά Πλέγματα

Η βελτιστοποίηση μορφής αφορά το μετασχηματισμό της υπό εξέταση γεωμετρίας προκειμένου να αυξηθεί η αεροδυναμική ή υδροδυναμική της απόδοση. Το σχήμα της γεωμετρίας ελέγχεται από ένα σύνολο μεταβλητών, τις επονομαζόμενες μεταβλητές σχεδιασμού ($\vec{b}$), ενώ η προς μεγιστοποίηση ή ελαχιστοποίηση ρευστοδυναμική ποσότητα ονομάζεται συνάρτηση στόχος (F) και ο υπολογισμός της προϋποθέτει την επίλυση των εξισώσεων ροής. Η διδακτορική διατριβή εφαρμόζει μεθόδους βελτιστοποίησης που υπολογίζουν και χρησιμοποιούν τις παραγώγους της F ως προς $\vec{b}$, τις λεγόμενες και παραγώγους ευαισθησίας. Συγκεκριμένα, επιλέγεται η μέθοδος των συζυγών κλίσεων [14], η οποία ξεκινά από μια αρχική γεωμετρία ακολουθεί την κατεύθυνση που προσδιορίζεται από το διάνυσμα $\vec{p}$, οι συνιστώσες του οποίου υπολογίζονται σε κάθε κύκλο ως

$$p_i^{new} = - \left. \frac{\partial F}{\partial b_i} \right|^{new} + \beta^{new} p_i^{old}$$

όπου β^{new} είναι πραγματικός αριθμός εξαρτώμενος από τις παραγώγους της F σε προηγούμενους κύκλους βελτιστοποίησης.

Σε πρακτικές εφαρμογές, ο υπολογισμός των παραγώγων $\partial F/\partial b_i$ είναι αρκετά δύσκολος λόγω της απουσίας αναλυτικής έκφρασης για την F. Ένας άμεσος αλλά πολύ ακριβός τρόπος για την προσέγγιση της τιμής τους είναι η μέθοδος των Πεπερασμένων Διαφορών (ΠΔ), που απαιτεί την επίλυση των ροϊκών ΜΔΕ $2N$ φορές, όπου N ο αριθμός των b_i. Εδώ, λόγω του κόστους της, η μέθοδος αυτή χρησιμοποιείται μόνο επιλεκτικά για την επαλήθευση της ακρίβειας των παραγώγων της συζυγούς τεχνικής, της οποίας το υπολογιστικό κόστος είναι ανεξάρτητο του N [34].

Η βασική στρατηγική που ακολουθεί η συζυγής μέθοδος βασίζεται στον ορισμό των συζυγών πεδιακών μεταβλητών, οι οποίες επαληθεύουν τις συζυγείς ΜΔΕ, το κόστος της αριθμητικής επίλυσης των οποίων είναι εφάμιλλο του κόστους επίλυσης των ΜΔΕ του πρωτεύοντος προβλήματος, δηλαδή των εξισώσεων ροής. Σύμφωνα με τη μέθοδο αυτή, οι $\partial F/\partial b_i$ εκφράζονται συναρτήσει των μεταβλητών ροής και των αντίστοιχων συζυγών μεταβλητών και ο υπολογισμός τους επιτυγχάνεται με απλή αντικατάστασή τους στην έκφραση των παραγώγων ευαισθησίας.

Η συζυγής μέθοδος συναντάται στη βιβλιογραφία με δύο εναλλακτικές διατυπώσεις, τη συνεχή [2] και τη διακριτή [17]. Η συνεχής διατύπωση παραγωγίζει τις ΜΔΕ του πρωτεύοντος προβλήματος καταλήγοντας στην αναλυτική έκφραση των συζυγών ΜΔΕ και της έκφρασης των παραγώγων ευαισθησίας. Στη συνέχεια, επιλέγεται ένα σχήμα διακριτοποίησης αντίστοιχο του σχήματος που χρησιμοποιήθηκε για της εξισώσεις ροής. Έτσι, η επίλυση των συζυγών ΜΔΕ παρουσιάζει πολλές ομοιότητες με το ροϊκό πρόβλημα καθιστώντας εύκολη την ανάπτυξη του συζυγούς λογισμικού. Αντίθετα, ένα μειονέκτημα της μεθόδου έγκειται στο γεγονός ότι η επιλογή του συζυγούς σχήματος διακριτοποίησης δεν είναι προφανής και μια ατυχής επιλογή πιθανώς να οδηγήσει σε λανθασμένη εκτίμηση των παραγώγων ευαισθησίας.

Σε αντιδιαστολή με τη συνεχή διατύπωση, η διακριτή τεχνική παραγωγίζει τη διακριτή έκφραση των ΜΔΕ του πρωτεύοντος προβλήματος, οδηγώντας στη διακριτή έκφραση των συζυγών ΜΔΕ και της έκφρασης των παραγώγων ευαισθησίας. Έτσι, η μέθοδος υπολογίζει την ακριβή τιμή των $\partial F/\partial b_i$. Παρόλ' αυτά το συζυγές σχήμα προκύπτει αρκετά πιο περίπλοκο, δυσχεραίνοντας την ανάπτυξη του αντίστοιχου συζυγούς λογισμικού, με συγκριτικά μεγαλύτερες απαιτήσεις σε υπολογιστική μνήμη, [33].

Η διατριβή υιοθετεί και τις δύο διατυπώσεις της συζυγούς μεθόδου. Επιπλέον, συνδυάζει τα πλεονεκτήματα της ΜΤΚ με το χαμηλό κόστος των συζυγών τεχνικών, παρουσιάζοντας ένα ισχυρό υπολογιστικό εργαλείο για τη βελτιστοποίηση σε πρακτικές εφαρμογές. Συγκεκριμένα, η χρήση καρτεσιανών πλεγμάτων υπερέχει έναντι άλλων τεχνικών, καθώς δεν απαιτεί την παραμόρφωση του πλέγματος λόγω της μεταβολής του σχήματος της γεωμετρίας κατά τη βελτιστοποίηση. Έτσι, αποφεύγει τον πρόωρο τερματισμό της βελτιστοποίησης λόγω αστοχίας του πλέγματος, κάτι που ενίοτε συμβαίνει σε μεθόδους βασισμένες σε οριόδετα πλέγματα. Η εφαρμογή των συζυγών τεχνικών σε ΜΕΣ εμφανίζεται ιδιαίτερα σπάνια στη βιβλιογραφία [12], [19] και ακόμα λιγότερο σε ΜΤΚ. Η μόνη γνωστή στο συγγραφέα εργασία παρουσιάζει τη διακριτή διατύπωση εφαρμοσμένη σε πλέγματα τεμνόμενων κυψελών για τη βελτιστοποίηση σε μόνιμες ατριβείς ροές [32]. Συνεπώς, μια από τις μεγαλύτερες καινοτομίες της διατριβής συνιστά το συνδυασμό της συνεχούς και διακριτής διατύπωσης με τη ΜΤΚ σε προβλήματα βελτιστοποίησης συνεκτικών ή/και μη-μόνιμων ροών.

Κεφάλαιο 2

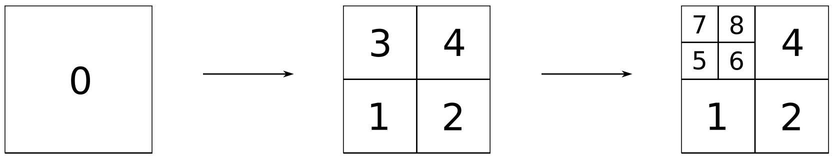
Γένεση Καρτεσιανού Πλέγματος Τεμνόμενων Κυψελών

Το κεφάλαιο αυτό παρουσιάζει τις γενικές αρχές που διέπουν τη διαδικασία της πλεγματοποίησης και εξετάζει κάποια ιδιαίτερα χαρακτηριστικά της.

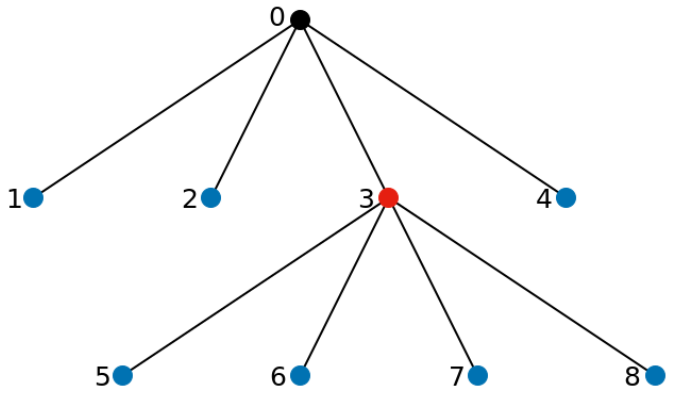
2.1 Δενδρική Γένεση Πλέγματος

Η διαδικασία δημιουργίας ενός 3Δ πλέγματος ξεκινά με τον ορισμό του υπολογιστικού χωρίου σχήματος ορθογώνιου παραλληλεπιπέδου διαστάσεων $d_x \times d_y \times d_z$. Η επιφάνεια της υπό μελέτη γεωμετρίας διακριτοποιείται με τη βοήθεια τριγωνικών στοιχείων. Το χωρίο διασπάται σε 8 ίσες κυψέλες (4 κυψέλες για 2Δ πλέγματα), οι οποίες θα ονομάζονται απόγονοι της αρχικής κυψέλης. Η διάσπαση μπορεί να αποτυπωθεί με τη βοήθεια μιας δενδρικής δομής, όπου η κάθε κυψέλη αντιπροσωπεύεται από ένα σημείο, το οποίο συνδέεται με τους απογόνους του μέσω ευθύγραμμων τμημάτων. Ένα παράδειγμα ενός 2Δ πλέγματος φαίνεται στο σχήμα 2.1.

Η διαδοχική διαίρεση των κυψελών συνεχίζει έως ότου το μέγεθός τους να μην υπερβαίνει μια συγκεκριμένη τιμή που ορίζεται από τον χρήστη. Η διαδικασία της διχοτόμησης καθοδηγείται από την παρουσία των στερεών σωμάτων. Έτσι, εντοπίζονται οι κυψέλες που τέμνονται από το στερεό όριο, οι οποίες παράγουν νέες γενιές κελιών. Οι νέες κυψέλες ελέγχονται ξανά, έτσι ώστε να εντοπιστούν οι τεμνόμενες, οι οποίες θα υποβληθούν στην ίδια διαδικασία. Ο αλγόριθμος σταματά όταν όλες οι τεμνόμενες κυψέλες γίνουν μικρότερες ενός προκαθορισμένου μεγέθους. Προκειμένου να αποφευχθεί η γειτνίαση αρκετά διαφορετικών σε μέγεθος κυψελών, επιβάλλεται ο περιορισμός του μέγιστου αριθμού γειτόνων μέσω της ίδιας έδρας σε τέσσερις. Έτσι, κατά τη διάρκεια της γένεσης πλέγματος, κάθε κυψέλη που παραβιάζει αυτόν τον περιορισμό διασπάται σε μικρότερες κυψέλες και η πύκνωση του πλέγματος πάνω στο στερεό όριο μεταφέρεται και σε εσωτερικές κυψέλες. Το αποτέλεσμα αυτής της μεθόδου παρουσιάζεται στο σχήμα 2.2, όπου δημιουργείται πλέγμα γύρω από αεροτομή εντός τετραγωνικού χωρίου.

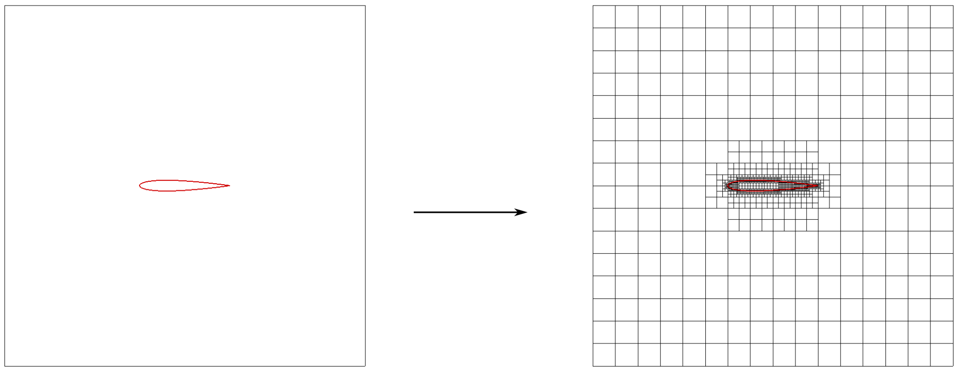


(α΄)



(β΄)

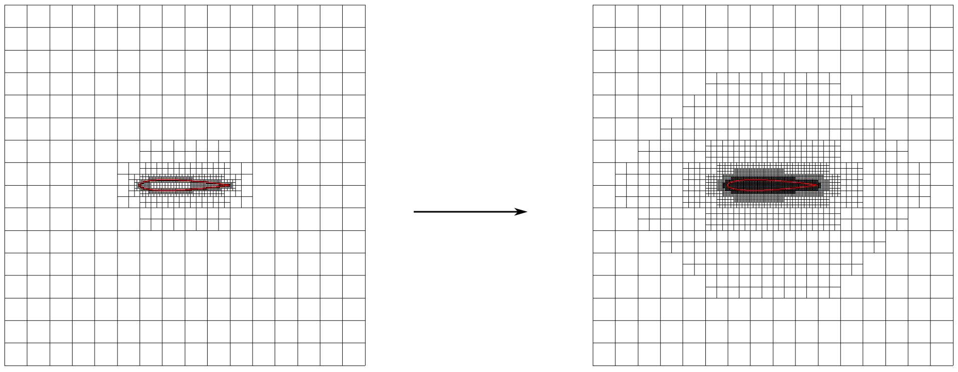
Σχήμα 2.1: (α΄) Η αρχική κυψέλη 0 διαχωρίζεται σε 4 απογόνους με αύξοντες αριθμούς 1, 2, 3, και 4. Στη συνέχεια, η κυψέλη 3 διαχωρίζεται εκ νέου στις κυψέλες 5, 6, 7, και 8. (β΄) Η παραπάνω διαδικασία αποτυπώνεται με τη βοήθεια της δενδρικής δομής, όπου κάθε σημείο αντιστοιχεί σε μια κυψέλη. Το μπλε χρώμα σηματοδοτεί τις κυψέλες του τελικού πλέγματος.



(α′) (β′)

Σχήμα 2.2: (α) Το τετραγωνικό χωρίο γύρω από την αεροτομή καταλαμβάνεται από την αρχική κυψέλη της δενδρικής δομής. (β′) Προκύπτον καρτεσιανό πλέγμα, το οποίο έχει προσαρμοστεί στο στερεό όριο.

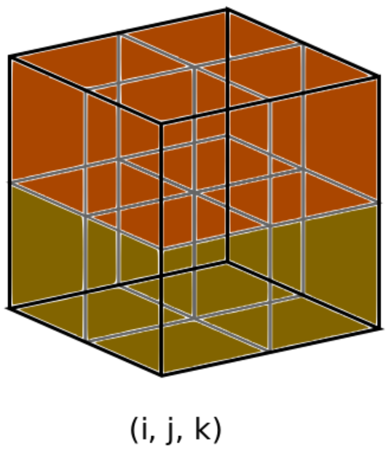
Όπως φαίνεται και από το ανωτέρω σχήμα, η αραίωση του πλέγματος με την αύξηση της απόστασης από το στερεό όριο είναι αρκετά απότομη για να υποστηρίξει την ακριβή επίλυση των εξισώσεων ροής. Έτσι, ακολουθεί επεξεργασία του πλέγματος με σκοπό την ομαλότερη μεταβολή της πυκνότητάς του. Στην περίπτωση του παραδείγματος της αεροτομής, το αποτέλεσμα της εξομάλυνσης φαίνεται στο σχήμα 2.3.



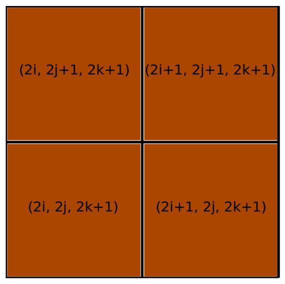
(α′) (β′)

Σχήμα 2.3: (α′) Πλέγμα του παραδείγματος του σχήματος 2.2 πριν (α′) και μετά (β′) την εξομάλυνση της μεταβολής της πυκνότητάς του.

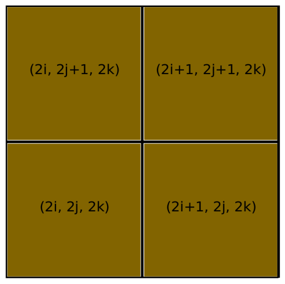
Η δενδρική δομή και καρτεσιανή φύση του πλέγματος επιτρέπουν την πολύ οικονομική αποθήκευσή του. Συγκεκριμένα, για την αρίθμηση των κυψελών χρησιμοποιείται ένα σύστημα τριών (ή δύο για 2Δ) ακεραίων εμπνευσμένο από την ονοματολογία των κόμβων ενός δομημένου πλέγματος. Έτσι, για κάθε κυψέλη αποθηκεύονται μόνο οι αριθμοί (i, j, k), οι οποίοι χρησιμοποιούνται για τον υπολογισμό όλων των απαραίτητων γεωμετρικών μεγεθών [21]. Η ονοματολογία ξεκινά με την αρχική κυψέλη του πλέγματος, η οποία αντιστοιχεί εξ ορισμού στην τριάδα $(1, 1, 1)$. Η αρίθμηση κάθε νέας κυψέλης εξαρτάται από αυτήν του γονέα της σύμφωνα με τον κανόνα που εικονίζεται στο σχήμα 2.4.



(α′)



(β′)



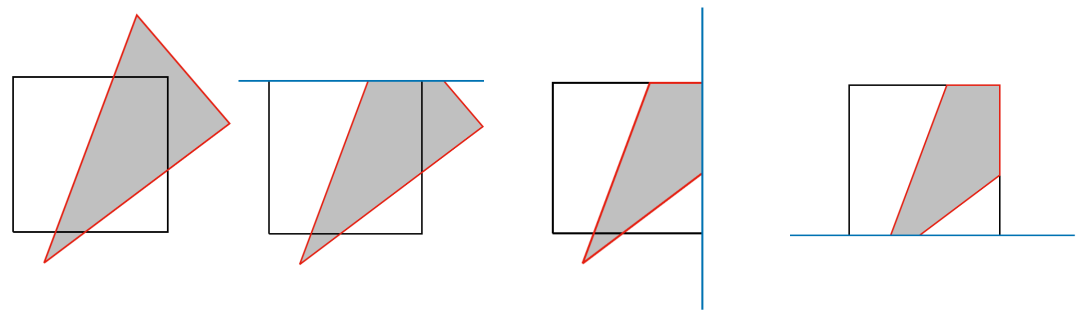
(γ′)

Σχήμα 2.4: (α′) Μια τυχαία κυψέλη που αντιστοιχεί στην τριάδα αρίθμησης (i, j, k) υποδιαιρείται σε 8 απογόνους. Οι τριάδες των αριθμών που τους αντιστοιχούν φαίνονται στα σχήματα (β′) και (γ′) για την άνω και κάτω τετράδα αντίστοιχα.

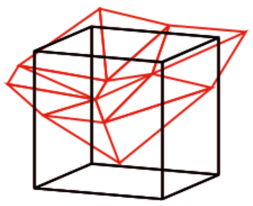
2.2 Κατασκευή των Τεμνόμενων Κυψελών

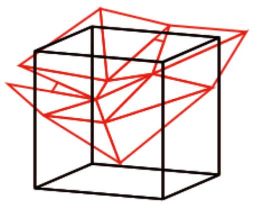
Η κατασκευή των πολυεδρικών τεμνόμενων κυψελών ξεκινά με την ανίχνευση των τριγωνικών επιφανειακών στοιχείων της γεωμετρίας, τα οποία συμπεριλαμβάνονται εξ ολοκλήρου ή κατά μέρος εντός κάθε κυψέλης. Στη συνέχεια, το τμήμα κάθε τριγώνου που "βρέχεται" από το ρευστό αποκόπτεται προκειμένου να οριοθετηθεί η έδρα του τελικού όγκου ελέγχου. Κατά τη διαδικασία αυτή, εφαρμόζεται ο αλγόριθμος των Sutherland & Hodgman [43], προκειμένου να υπολογιστεί το ορατό τμήμα ενός πολυγωνικού αντικειμένου μέσα από ένα παράθυρο κυρτού πολυγωνικού σχήματος. Στην περίπτωση των τεμνόμενων κυψελών το αντικείμενο είναι το τριγωνικό επιφανειακό στοιχείο και το παράθυρο είναι η κυψέλη.

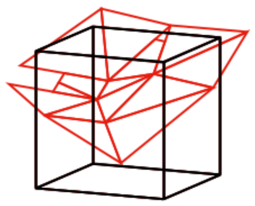
Η λειτουργία του αλγόριθμου γίνεται πιο κατανοητή μέσα από το παράδειγμα του σχήματος 2.5, όπου εντοπίζεται το τμήμα του σκιαζόμενου τριγώνου, το οποίο είναι ορατό μέσα από ένα τετραγωνικό παράθυρου. Η διαδικασία χωρίζεται σε τέσσερα βήματα, δηλαδή όσα και οι ακμές του παράθυρου. Σε κάθε βήμα επεκτείνεται η αντίστοιχη ακμή χωρίζοντας το επίπεδο σε δύο ημιεπίπεδα με το πρώτο εξ αυτών να συμπεριλαμβάνει το τετράγωνο. Υπολογίζονται οι τομές της επαυξημένης ακμής με το τρίγωνο και αποκόπτεται το τμήμα το οποίο ανήκει στο δεύτερο ημιεπίπεδο. Το προκύπτον σχήμα εισάγεται στο δεύτερο βήμα, όπου επαναλαμβάνεται η ίδια διαδικασία για την επόμενη ακμή. Η εφαρμογή του αλγορίθμου σε μια καρτεσιανή κυψέλη φαίνεται στο σχήμα 2.6.

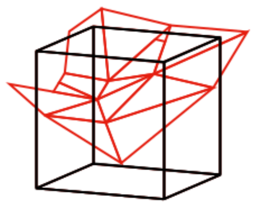


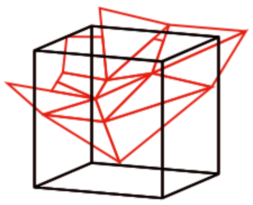
Σχήμα 2.5: Γεωμετρική αναπαράσταση του αλγορίθμου Sutherland & Hodgman. Κάθε ακμή του τετραγώνου επεκτείνεται και το μη-ορατό τμήμα του σκιαζόμενου αντικειμένου αποκόπτεται.

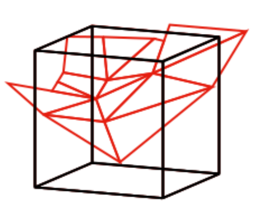

(α΄)

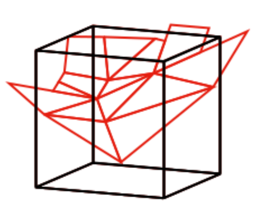

(β΄)

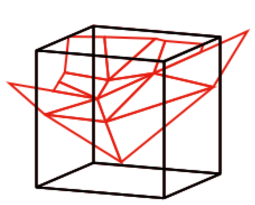

(γ΄)

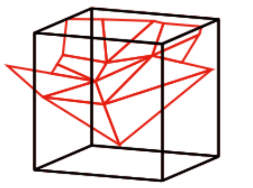

(δ΄)

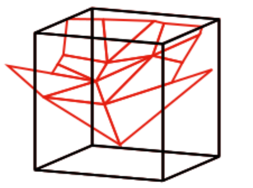

(ε΄)

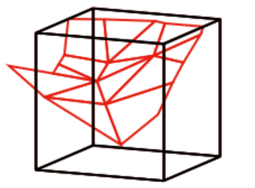

(ϛ΄)

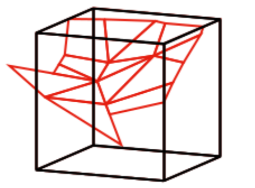

(ζ΄)

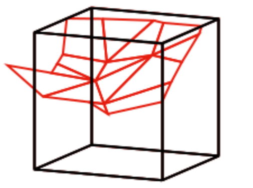

(η΄)

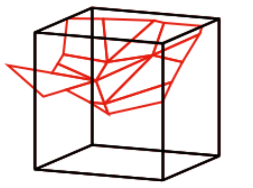

(ϑ΄)

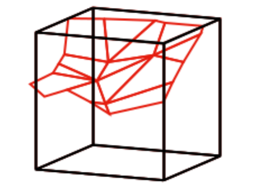

(ι΄)

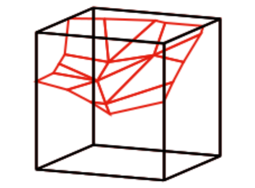

(ια΄)


(ιβ΄)


(ιγ΄)

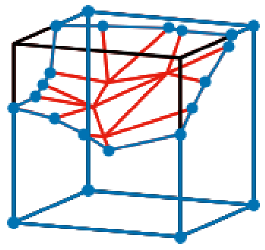

(ιδ΄)

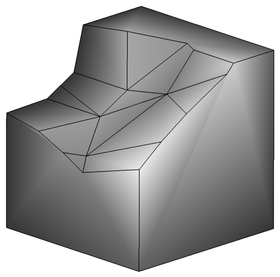

(ιε΄)


(ιϛ΄)

Σχήμα 2.6: Η επιφάνεια του στερεού σώματος, που εικονίζεται με κόκκινο χρώμα, αποκόπτεται σε διαδοχικά στάδια διατηρώντας μόνο το τμήμα της γεωμετρίας που βρίσκεται εντός της καρτεσιανής κυψέλης.

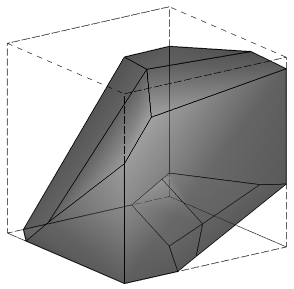
Τέλος, σχηματίζονται οι υπόλοιπες πολυγωνικές έδρες με τις οποίες επικοινωνεί η κυψέλη με τις γειτονικές της. Το σχήμα 2.7α΄ παρουσιάζει με μπλε χρώμα τις εν λόγω έδρες της τεμνόμενης κυψέλης του σχήματος 2.6. Έτσι, ολοκληρώνεται η κατασκευή του όγκου ελέγχου, ο οποίος φαίνεται με τριδιάστατη σκίαση στο σχήμα 2.7β΄.

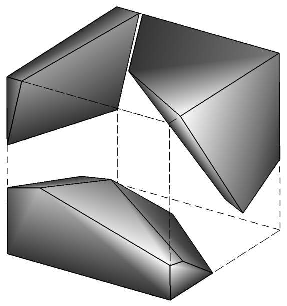

(α′)

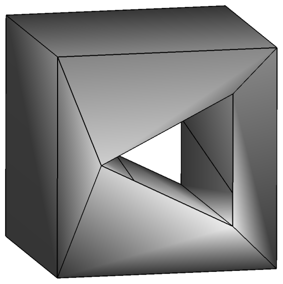

(β′)

Σχήμα 2.7: (α′) Σχηματισμός των εδρών της τεμνόμενης κυψέλης του σχήματος 2.6 που κείνται εντώς του ρέοντος ρευστού. (β′) Τριδιάστατη απεικόνιση της τεμνόμενης κυψέλης.

Η μέθοδος που αναπτύχθηκε κατά τη διάρκεια της διατριβής μπορεί να αντιμετωπίσει οποιαδήποτε περίπτωση τομής μεταξύ του πλέγματος και της γεωμετρίας ακόμα και σε περιπτώσεις που θεωρούνται ιδιάζουσες και συνήθως αποφεύγονται από άλλα λογισμικά λόγω της υψηλής πολυπλοκότητάς τους. Η τοπική πύκνωση του πλέγματος ακολουθείται συχνά προκειμένου να αποφευχθεί ο υπολογισμός τοπολογικά περίπλοκων τομών, κάτι που όμως αυξάνει το υπολογιστικό κόστος της επίλυσης της ροής και, συνεπώς, δεν υιοθετήθηκε. Παραδείγματα απαιτητικών περιπτώσεων που μπορεί να αντιμετωπίσει ο προτεινόμενος αλγόριθμος παρουσιάζονται στο σχήμα 2.8.

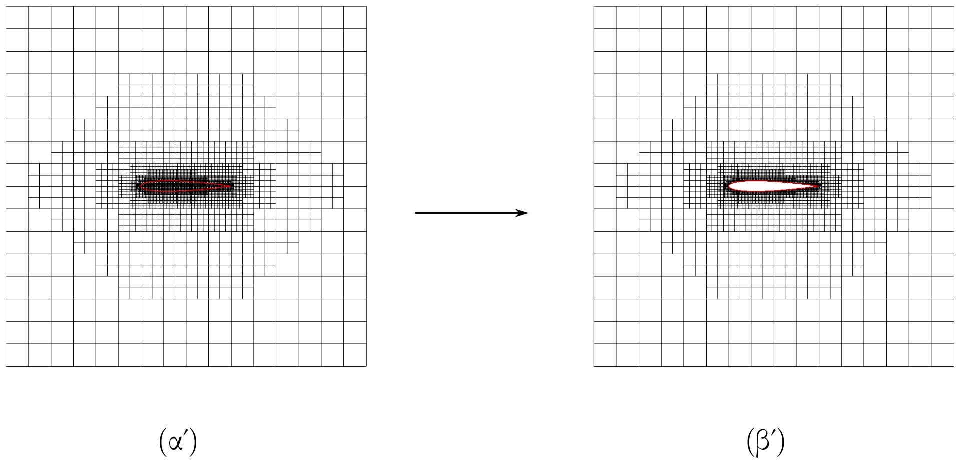

(α′)


(β′)


(γ′)

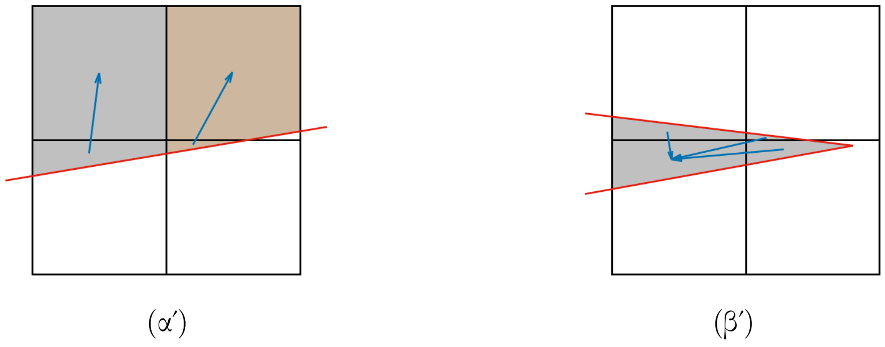
Σχήμα 2.8: Ιδιάζουσες περιπτώσεις τομής κυψελών από το στερεό όριο. (α′) Κυψέλη που τέμνεται από δύο διαφορετικές πλευρές (β′) Κυψέλη που έχει διαχωριστεί σε 3 επιμέρους τμήματα. (γ′) Διάτρηση κυψέλης από στερεό σώμα.

Η κατασκευή των τεμνόμενων κυψελών επιτρέπει την απόσχιση και αποβολή του στερεού μέρους του πλέγματος. Το σχήμα 2.9 παρουσιάζει το καρτεσιανό πλέγμα, όπως κατασκευάστηκε από τον αλγόριθμο της ενότητας 2.1 και το προκύπτον πλέγμα μετά την κατασκευή των τεμνόμενων κυψελών και την αποβολή των κυψελών που βρίσκονται εξολοκλήρου μέσα στο στερεό σώματος.



Σχήμα 2.9: (α΄) Πλέγμα γύρω απο αεροτομή όπως φαίνεται στο σχήμα 2.3β΄. (β΄) Προκύπτον πλέγμα μετά την αποκοπή του στερεού του τμήματος

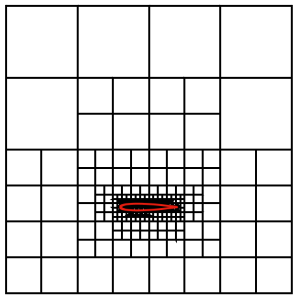
Όμως, το πλέγμα αυτό δεν είναι ακόμα κατάλληλο προς χρήση για την επίλυση της ροής, επειδή μικρά θραύσματα κυψελών συνορεύουν με αρκετά μεγαλύτερες κυψέλες προκαλώντας αριθμητική αστάθεια κατά την επίλυση των εξισώσεων ροής. Έχουν προταθεί πλείστες τεχνικές για την αντιμετώπιση αυτού του φαινομένου [49], [26]. Η μέθοδος που εφαρμόζεται στη διατριβή βασίζεται στην τεχνική της γεωμετρικής συγχώνευσης των μικρών κελιών με άλλα μεγαλύτερα [22] λόγω της αυξημένης ακρίβειας πρόλεξης της ροής που προσφέρει. Η συγχώνευση αυτή δημιουργεί νέους ενιαίους όγκους ελέγχου και οι μεταβλητές της ροής αποθηκεύονται στο βαρύκεντρο του συσσωματώματος. Το σχήμα 2.10α΄ παρουσιάζει δύο μικρές κυψέλες που συνενώνονται με τις γειτονικά τους, ενώ το σχήμα 2.10β΄ φανερώνει τη δυνατότητα του λογισμικού να προσαρτά παραπάνω της μιας κυψέλες σε έναν ενιαίο όγκο ελέγχου.

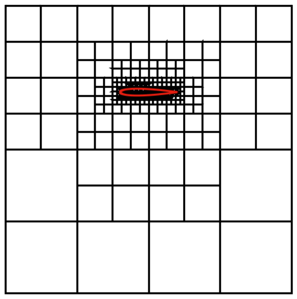


Σχήμα 2.10: (αʹ) Δυο μικρές κυψέλες συγχωνεύονται με δύο γείτονές τους όπως δείχνει το βέλος. Ο ομοιόμορφος χρωματισμός καταδεικνύει τα προκύπτοντα συσσωματώματα. (βʹ) Τέσσερις μικρές κυψέλες ενώνονται προκειμένου να σχηματίσουν έναν ενιαίο όγκου ελέγχου μεγαλύτερου μεγέθους.

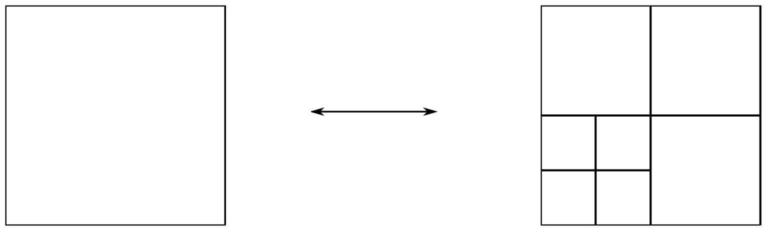
2.3 Γένεση Πλέγματος σε Χρονικά Μεταβαλλόμενα Χωρία

Η γένεση καρτεσιανού πλέγματος γύρω από κινούμενες γεωμετρίες παρουσιάζει ιδιαιτερότητες, οι οποίες δε συναντώνται σε οριόδετα πλέγματα και απαιτείται συνεχής προσαρμογή του πλέγματος στο στερεό όριο μετά από κάθε κίνησή του, σχήμα 2.11. Συγκεκριμένα, οι περιοχές κοντά στην προγενέστερη θέση της γεωμετρίας υποβάλλονται σε διαδικασία αραίωσης ενώ κελιά κοντά στη νέα θέση της διασπώνται αυξάνοντας την πύκνωση του πλέγματος. Έτσι, διατηρείται η ακρίβεια της διακριτοποίησης των ροϊκών ΜΔΕ κοντά στα κινούμενα στερεά τοιχώματα καθόλη τη διάρκεια της προσομοίωσης. Παρόλ' αυτά η μεταβολή της τοπολογίας του πλέγματος κάθε χρονική στιγμή επιβάλλει τη μεταφορά των μεγεθών του πεδίου ροής από το παλαιό πλέγμα στο νέο προκειμένου να διασφαλιστεί η ιστορική συνέχεια των μεταβλητών ροής. Εκμεταλλευόμενη την καρτεσιανή δομή των εν λόγω πλεγμάτων, η διαδικασία αυτή γίνεται εύκολα και γρήγορα καθώς περιλαμβάνει μόνο δύο περιπτώσεις παρεμβολής, 2.12. Στην πρώτη περίπτωση, μια κυψέλη της παλαιάς χρονικής στιγμής διασπάται σε επιμέρους κυψέλες, ενώ στη δεύτερη μια ομάδα κυψελών συγχωνεύονται σε μια.


(α΄)


(β΄)

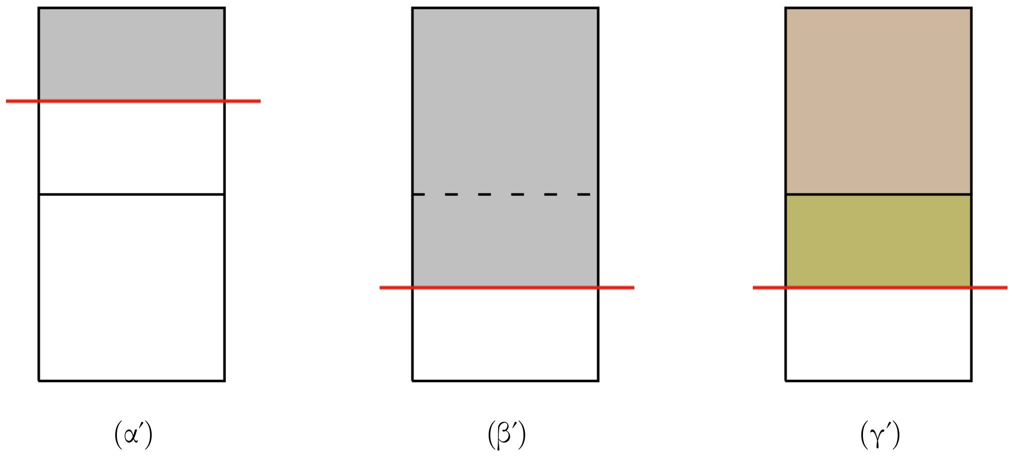
Σχήμα 2.11: Μια μεμονωμένη αεροτομή πραγματοποιεί μεταφορική κίνηση σε καρτεσιανό πλέγμα, το οποίο προσαρμόζεται σε κάθε χρονική στιγμή παρακολουθώντας την κίνησή της.


(α΄) (β΄)

Σχήμα 2.12: Περιοχή πλέγματος η οποία μεταβάλλεται μεταξύ δύο διαδοχικών χρονικών στιγμών. Το διπλό βέλος σηματοδοτεί τη δυνατότητα αραίωσης ή πύκνωσης του πλέγματος.

Η χρήση μόνο του εκτεθειμένου στη ροή τμήματος του πλέγματος προκαλεί δυσκολίες σε χρονικά μεταβαλλόμενα χωρία καθώς το κινούμενο στερεό σώμα καλύπτει ή αποκαλύπτει κυψέλες στο πέρασμά του. Ένα παράδειγμα φαίνεται στο σχήμα 2.13, όπου πριν τη μετατόπιση του στερεού ορίου, σχήμα 2.13α΄, μόνο η άνω κυψέλη συμμετέχει στην επίλυση των εξισώσεων ροής. Στη συνέχεια, το τοίχωμα κινείται προς τα κάτω και αποκαλύπτει ένα μέρος της κάτω κυψέλης, σχήμα 2.13γ΄, η οποία καθίσταται ιδιαίτερα προβληματική στην αντιμετώπισή της κατά τη διακριτοποίηση των ροϊκών ΜΔΕ. Συγκεκριμένα, η διακριτοποίηση του χρονικού όρου των εξισώσεων είναι αδύνατη αφού απαιτείται η χρήση των μεταβλητών της ροής την προγενέστερη χρονική στιγμή, όπου η κυψέλη καλυπτόταν από το στερεό. Η λύση που προτείνεται είναι η δημιουργία ενός ενδιάμεσου βήματος που εικονίζεται στο σχήμα 2.13β΄, όπου η νεογεννηθείσα κυψέλη συγχωνεύεται με τη γειτονική της, η οποία έχει μια συνεχή χρονική παρουσία μεταξύ των δύο χρονικών βημάτων. Έτσι, η διακριτοποίηση αντιμετωπίζει το συσσωμάτωμα ως τη νέα κατάσταση της κυψέλης του σχήματος 2.13α′. Μόλις ολοκληρωθεί η επίλυση της ροής για το τρέχον χρονικό βήμα, το συσσωμάτωμα διαχωρίζεται εκ νέου στις επιμέρους κυψέλες, σχήμα 2.13γ′, και η επίλυση της ροής συνεχίζεται στο επόμενο βήμα.

Εξίσου προβληματική είναι η ξαφνική κάλυψη μιας κυψέλης από το στερεό σώμα. Ακολουθώντας τα στιγμιότυπα του σχήματος 2.13 με αντίστροφη σειρά, η ανύψωση του τοιχώματος καλύπτει την κάτω κυψέλη της χρονικής στιγμής 2.13γ′. Έτσι, τη στιγμή 2.13α′ δε συμμετέχει στην επίλυση της ροής και η μάζα, ορμή και ενέργεια, που έχουν αποθηκευθεί σε αυτήν, χάνονται. Συνεπώς, οι καλυφθείσες κυψέλες λειτουργούν ως καταβόθρες αλλοιώνοντας το πεδίο ροής. Προκειμένου να αποφευχθεί αυτό, οι κυψέλες αυτές ενώνονται με γειτονικές τους σε ένα ενδιάμεσο βήμα, σχήμα 2.13β′, έτσι ώστε να τους αποδώσουν τις μεταβλητές της ροής που έχουν αποθηκεύσει προτού εξαφανιστούν.



Σχήμα 2.13: Από αριστερά προς τα δεξιά το στερεό όριο κινείται προς τα κάτω και αποκαλύπτει μια νέα κυψέλη. Η αντίθετη κίνηση του σώματος, από δεξιά προς τα αριστερά, καλύπτει μια κυψέλη. Τα χρωματισμένα χωρία αντιστοιχούν στο τμήμα του πλέγματος που βρέχεται από τη ροή και καταδεικνύουν τη σωστή συγχώνευση των καλυπτόμενων ή αποκαλυπτόμενων κυψελών.

Κεφάλαιο 3

Διακριτοποίηση των Εξισώσεων Navier-Stokes

3.1 Διακριτοποίηση των Εξισώσεων Συμπιεστής ροής

Η κίνησης των συμπιεστών ρευστών μοντελοποιείται μέσω των εξισώσεων Navier-Stokes και των εξισώσεων διατήρησης της μάζας και της ενέργειας. Αυτές αποτυπώνονται σε ένα καρτεσιανό σύστημα αξόνων (x_1, x_2, x_3) ως [28]

$$\frac{\partial U_i}{\partial t} + \frac{\partial f_{ik}^{inv}}{\partial x_k} - \frac{\partial f_{ik}^{vis}}{\partial x_k} = 0, \quad i = 1, \cdots, 5, \; k = 1, \cdots, 3$$

με

$$\vec{U} = \begin{bmatrix} \rho \\ \rho v_1 \\ \rho v_2 \\ \rho v_3 \\ \rho E \end{bmatrix}, \quad \vec{f}_k^{inv} = \begin{bmatrix} \rho v_k \\ \rho v_k v_1 + \delta_{1k} p \\ \rho v_k v_2 + \delta_{2k} p \\ \rho v_k v_3 + \delta_{3k} p \\ \rho v_k h_t \end{bmatrix}, \quad \vec{f}_k^{vis} = \begin{bmatrix} 0 \\ \tau_{1k} \\ \tau_{2k} \\ \tau_{3k} \\ v_j \tau_{jk} + q_k \end{bmatrix}$$

όπου σύμφωνα με τη σύμβαση κατά Eintein οι επαναλαμβανόμενοι δείκτες υποδηλώνουν άθροιση. Τα διανύσματα $\vec{U}$, $\vec{f}_k^{inv}$ και $\vec{f}_k^{vis}$ συμβολίζουν τις συντηρητικές ροϊκές μεταβλητές, τους όρους συναγωγής και τους όρους διάχυσης, αντίστοιχα. Το διάνυσμα των μη-συντηρητικών μεγεθών ορίζεται ως $\vec{V} = [\rho, \; v_1, \; v_2, \; v_3, \; p]$, όπου οι μεταβλητές

ρ, v_i και p, δηλώνουν την πυκνότητα, την ταχύτητα ανά κατεύθυνση και την πίεση. Ακόμα, E και h_t είναι η ενέργεια κατά βάρος και ολική ενθαλπία του ρευστού. Η θερμορoή q_m δίδεται από το νόμο του Fourier και ο τανιστής των τάσεων τ εκφράζεται ως $\tau_{ik} = \mu \left(\frac{\partial v_i}{\partial x_k} + \frac{\partial v_k}{\partial x_i} - \frac{2}{3}\delta_{ik}\frac{\partial v_m}{\partial x_m} \right)$ όπου μ είναι η δυναμική συνεκτικότητα του ρευστού και δ_{ik} η δ-συνάρτηση του Kronecker.

Η διακριτοποίηση των ανωτέρω εξισώσεων γίνεται μέσω της κεντροκυψελικής διατύπωσης των πεπερασμένων όγκων, όπου οι ροϊκές εξισώσεις ολοκληρώνονται σε όγκους ελέγχου που ταυτίζονται με τις κυψέλες του καρτεσιανού πλέγματος, κάτι που έχει ως συνέπεια την ανταλλαγή πληροφορίας μεταξύ γειτονικών κυψελών μέσω των διακριτοποιημένων διανυσμάτων ατριβούς ($\vec{f}_k^{inv,m}$) και συνεκτικής ($\vec{f}_k^{vis,m}$) ροής. Έτσι, για μόνιμες ροές, η διακριτή έκφραση των εξισώσεων είναι

$$\frac{\bar{U}_i^{n+1} - \bar{U}_i^n}{\Delta\tau}\Omega + \sum_{m=1}^{M} \left(\bar{f}_{ik}^{inv,m,n+1} - \bar{f}_{ik}^{vis,m,n+1} \right) n_k^m \Delta S^m = 0$$

όπου έχει προστεθεί ένας ψευδο-χρονικός όρος βήματος $\Delta\tau$ προκειμένου να διατηρηθεί ο υπερβολικός χαρακτήρας των ΜΔΕ [44]. Επιπλέον, οι μέσες τιμές $\bar{U}_i^n$ αντιστοιχούν στις μεταβλητές ροής που αποθηκεύονται στο βαρύκεντρο κάθε κυψέλης όγκου Ω με τον ακέραιο n να μετρά την τρέχουσα ψευδο-χρονική επανάληψη. Τέλος, ο ακέραιος m αριθμεί τις πλευρές κάθε κυψέλης με ΔS^m και $\vec{n}^m$ να αναπαριστούν την επιφάνεια και το μοναδιαίο κάθετο διάνυσμα σε κάθε πλευρά.

Το διάνυσμα υπολογίζεται μέσω της κατά Roe προσεγγιστικής επίλυσης του προβλήματος Riemann [38] και χρησιμοποιούνται οι περιοριστές κατά Barth & Jespersen [3] ή κατά Venkatakrishnan [45]. Ενδεικτική γεωμετρική απεικόνιση της προεκβολής που απαιτείται φαίνεται στο σχήμα 3.1 για γειτονικές κυψέλες ίδιου ή διαφορετικού μεγέθους, καθώς και για τεμνόμενες κυψέλες.

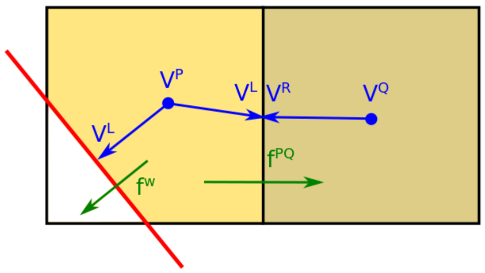
Σε περιπτώσεις κινούμενων στερεών σωμάτων, το σχήμα των τεμνόμενων κυψελών μεταβάλλεται με το χρόνο λόγω της μετατόπισης των στερεών τους εδρών, σχήμα 3.2. Η μεταβολή αυτή λαμβάνεται υπόψη μέσω της τεχνικής ALE (Arbitrary Lagrangian-Eulerian) [20], σύμφωνα με την οποία η ολοκληρωτική μορφή των εξισώσεων γράφεται

$$\frac{d}{dt}\int_{\Omega} U_i d\Omega - \int_{S} U_i v_k^g n_k dS + \int_{\Omega} \frac{\partial f_{ik}}{\partial x_k} d\Omega = 0$$

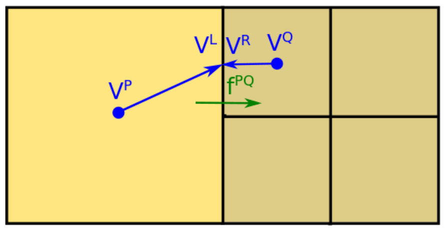
όπου v_k^g είναι η ταχύτητα της εκάστοτε έδρας. Η διακριτοποίηση της εξίσωσης οδηγεί στο σχήμα

$$\frac{\Omega^{n+1}U_i^{n+1,q+1} - \Omega^{n+\frac{1}{2}}U_i^{n+\frac{1}{2}}}{\Delta t} + \frac{\Omega^{n+1}U_i^{n+1,q+1} - \Omega^{n+1}U_i^{n+1,q}}{\Delta \tau} + \sum_{m=1}^{M} (f_{ik}^m n_k^m \Delta S^m)^{n+1,q+1} = 0$$

όπου οι μετρητές n και q αριθμούν το χρονικό και ψευδο-χρονικό βήμα αντίστοιχα. Επίσης, η χρονική στιγμή $n+1/2$ αντιστοιχεί στο πεδίο ροής της προηγούμενης χρονικής στιγμής προβεβλημένο στο τρέχον πλέγμα.



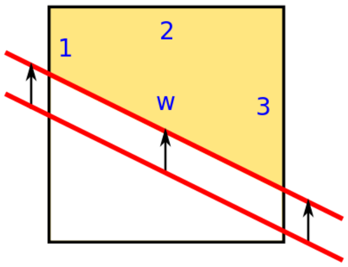
(α')



(β')

Σχήμα 3.1: Προεκβολή των μεγεθών $\vec{V}^P$ και $\vec{V}^Q$ από το βαρύκεντρο των κυψελών ίδιου ή διαφορετικού μεγέθους στο βαρύκεντρο των μεταξύ τους πλευρών, προκειμένου να υπολογιστούν οι τιμές των $\vec{V}^L$ και $\vec{V}^R$, που απαιτούνται από την έκφραση του διανύσματος ροής τόσο εσωτερικό ($\vec{f}^{PQ}$) και στο στερεό όριο ($\vec{f}^w$).

Η αριθμητική επίλυση του μη-μόνιμου προβλήματος γίνεται με την τεχνική του διπλού χρονικού βήματος [29], σύμφωνα με την οποία σε κάθε χρονική στιγμή λύνεται επαναληπτικά ένα ψευδο-χρονικό πρόβλημα. Η επίλυση του μη-γραμμικού συστήματος γίνεται με τη βοήθεια της μεθόδου Newton-Raphson. Έτσι, οι διακριτοποιημένες εξισώσεις συγκλίνουν σε κάθε χρονική επανάληψη προτού ο αλγόριθμος συνεχίσει στο επόμενο χρονικό βήμα, με αποτέλεσμα να επιτρέπεται επιλογή μεγαλύτερων χρονικών βημάτων χωρίς να προκαλείται αστάθεια στην αριθμητική επίλυση του συστήματος. Με αυτόν τον τρόπο επιτυγχάνεται η μείωση των χρονικών στιγμών και αποφεύγεται ο συχνός υπολογισμός της τομής του στερεού σώματος με τη γεωμετρία μειώνοντας το συνολικό χρόνο της προσομοίωσης.



Σχήμα 3.2: Μια τεμνόμενη κυψέλη παραμορφώνεται λόγω της κίνησης του στερεού ορίου ‘w’.

3.2 Διακριτοποίηση των Εξισώσεων Ασυμπίεστης Ροής

Η κίνηση ενός ρευστού σταθερής πυκνότητας μοντελοποιείται από τις εξισώσεις,

$$M_{ij}\frac{\partial V_j}{\partial t} + \frac{\partial f_{ik}^{inv}}{\partial x_k} - \frac{\partial f_{ik}^{vis}}{\partial x_k} = 0, \quad i = 1, \cdots, 4, \; k = 1, \cdots, 3$$

όπου

$$\vec{V} = \begin{bmatrix} p \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}, \quad \vec{f_k^{inv}} = \begin{bmatrix} v_k \\ v_k v_1 + \delta_{1k} p \\ v_k v_2 + \delta_{2k} p \\ v_k v_3 + \delta_{3k} p \end{bmatrix}, \quad \vec{f_k^{vis}} = \begin{bmatrix} 0 \\ \tau_{1k} \\ \tau_{2k} \\ \tau_{3k} \end{bmatrix}, \quad M = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Αντίθετα με τη συμπιεστή ροή, η μεταβλητή p προσδιορίζει την πίεση διαιρεμένη με την πυκνότητα. Προκειμένου να διατηρηθεί η ομοιομορφία του αναπτυχθέντος λογισμικού, η μέθοδος της τεχνητής συμπιεστότητας χρησιμοποιείται για να αποδώσει υπερβολικό χαρακτήρα στις ασυμπίεστες ΜΔΕ [8]. Η μέθοδος αυτή επιτρέπει τη δημιουργία ενός κοινού επιλύτη που καλύπτει όλες τις περιπτώσεις ροής με ελάχιστες διαφορές μεταξύ των σχημάτων διακριτοποίησης της συμπιεστής και ασυμπίεστης ροής. Έτσι, ο ψευδο-χρονικός όρος $\frac{1}{\beta^2}\frac{\partial p}{\partial \tau}$ προστίθεται στην εξίσωση συνέχειας. Η μεταβλητή β ονομάζεται τεχνητή συμπιεστότητα, επιλέγεται σταθερή για όλο το πεδίο, και η τιμή της επηρεάζει τόσο την ταχύτητα σύγκλισης όσο και την ακρίβεια του αποτελέσματος.

Κεφάλαιο 4

Η Συνεχής και Διακριτή Έκφραση των Συζυγών Εξισώσεων Ροής

Το κεφάλαιο αυτό παρουσιάζει τη μαθηματική διατύπωση της συζυγούς μεθόδου για 3Δ μη-μόνιμες συμπιεστές και ασυμπίεστες ροές για τον υπολογισμό της κλίσης της συνάρτησης στόχου F ως προς τις μεταβλητές σχεδιασμού $b_q,\ q = 1 \cdots N$ με υπολογιστικό κόστος ανεξάρτητο του N. Στα προβλήματα βελτιστοποίησης μορφής, οι μεταβλητές b_q ελέγχουν το σχήμα της υπό βελτιστοποίηση γεωμετρίας. Η γενική έκφραση της F που εξετάζεται στην εργασία αυτή είναι

$$F = \int_{T_F} \int_{\Omega_F} F_\Omega d\Omega dt + \int_{T_F} \int_{S_F} F_{S_k} n_k dS dt$$

όπου T_F είναι το χρονικό παράθυρο, στο οποίο η συνάρτηση στόχος παραμένει ενεργή, ενώ τα μεγέθη Ω_F και S_F αναπαριστούν τον όγκο ή επιφάνεια ορισμού της F.

4.1 Η Συνεχής Συζυγής Διατύπωση

Σύμφωνα με τη συνεχή συζυγή διατύπωση, η F επαυξάνεται κατά ένα μηδενικό ογκικό ολοκλήρωμα, οδηγώντας στον ορισμό της επαυξημένης συνάρτησης

$$L = F + \int_{T_R}\int_{\Omega} \Psi_i R_i d\Omega dt$$

όπου $\vec{R}=\vec{0}$ αναπαριστούν την αναλυτική έκφραση των ροϊκών ΜΔΕ, που επιλύονται για το διάστημα T_R, και $\vec{\Psi}$ είναι οι συζυγείς πεδιακές μεταβλητές. Η παραγώγιση της L οδηγεί σε μια έκφραση που περιλαμβάνει τους όρους $\partial U_i/\partial b_q$, των οποίων το κόστος υπολογισμού είναι ανάλογο του N. Έτσι, επιβάλλεται ο μηδενισμός των πολλαπλασιαστών τους ορίζοντας τις συζυγείς πεδιακές εξισώσεις. Η αντίστοιχη μαθηματική ανάλυση για συμπιεστές ροές καταλήγει

$$-\frac{\partial \Psi_i}{\partial t} - A_{jik}\frac{\partial \Psi_j}{\partial x_k} - T_i^{vis} + \frac{\partial F_\Omega}{\partial U_i} = 0$$

όπου

$$T_i^{vis} = \frac{\partial \tau_{kl}^A}{\partial x_k}\frac{\partial v_l}{\partial U_i} + \frac{\partial q_k^A}{\partial x_k}\frac{\partial T}{\partial U_i} - \frac{\partial \Psi_E}{\partial x_k}\tau_{lk}\frac{\partial v_l}{\partial U_i}$$

$$\tau_{kl}^A = \mu\left(h_{kl} + h_{lk} - \frac{2}{3}\delta_{kl}h_{mm}\right),$$
$$h_{kl} = \frac{\partial \Psi_{l+1}}{\partial x_k} + \frac{\partial \Psi_E}{\partial x_k}v_l, \quad q_k^A = k\frac{\partial \Psi_E}{\partial x_k}$$

Η διακριτοποίησή των συζυγών ΜΔΕ είναι αντίστοιχη με αυτήν των εξισώσεων ροής. Μια σημαντική ιδιομορφία του συζυγούς προβλήματος είναι η επίλυσή του αντίστροφα στο χρόνο. Αυτό συνεπάγεται την αποθήκευση εκ των προτέρων όλων των στιγμιοτύπων του πεδίου ροής, προκειμένου να χρησιμοποιηθούν κατά την επίλυση των συζυγών ΜΔΕ με αντίστροφη σειρά. Καθώς, οι υπολογιστικοί πόροι δεν επαρκούν πάντα για την αποθήκευση ενός τόσο μεγάλου όγκου πληροφορίας, στο πλαίσιο της διατριβής αναπτύχθηκαν λογισμικά συμπίεσης δεδομένων βασισμένα στις μεθόδους Singular Value Decomposition (SVD) [46] και Proper Generalized Decomposition (PGD) [7].

Τέλος, η αριθμητική επίλυση των ανωτέρω ΜΔΕ οδηγεί στον υπολογισμό του συζυγούς πεδίου, το οποίο χρησιμοποιείται για τον υπολογισμό της κλίσης της F μέσω της έκφρασης

$$\begin{aligned}
\frac{\delta F}{\delta b_q} &= \int_{T_R}\int_{S_w}\left(\Psi_i v_n^g \frac{\partial U_i}{\partial n} - \Psi_i \frac{\partial f_{ik}}{\partial n} n_k - \tau_{kl}^A n_k \frac{\partial v_l}{\partial n} + F_\Omega - F_{S_k} n_k H\right) v_n^s dSdt \\
&+ \int_{T_R}\int_{S_w}\left(F_{S_k} + p\Psi_{k+1} - \Psi_i f_{ik} - g_k \tau^n\right)\frac{\delta_s n_k}{\delta_s b_q} dSdt - \int_{T_R}\int_{S_w} g_k \tau^{t^r} \frac{\delta_s t_k^r}{\delta_s b_q} dSdt \\
&+ \int_{T_R}\int_{S_w}\left(\frac{\partial F_{S_k}}{\partial v_l^g} + \tau_{kl}^A - \Psi_E \tau_{kl}\right)\frac{\delta_s v_l^g}{\delta_s b_q} n_k dSdt \\
&+ \int_{T_R}\int_{S_w}\left(\Psi_i U_i + p\Psi_E\right)\frac{\delta_s v_n^g}{\delta_s b_q} dSdt
\end{aligned}$$

όπου $g_k = \Psi_{k+1} + v_k^g \Psi_E$ και $\vec{f}_k = \vec{f}_k^{inv} - \vec{f}_k^{vis}$.

Αντίστοιχα, προκύπτουν οι συζυγείς ΜΔΕ για τις ασυμπίεστες ροές και εξάγεται η έκφραση των παραγώγων ευαισθησίας

$$\begin{aligned}
\frac{\delta F}{\delta b_q} &= \int_{T_R}\int_{S_w}\left(M_{ij}\Psi_j v_n^g \frac{\partial V_i}{\partial n} - \Psi_i \frac{\partial f_{ik}}{\partial n} n_k - \tau_{kl}^A n_k \frac{\partial v_l}{\partial n} + F_\Omega - F_{S_k} n_k H\right) v_n^s dSdt \\
&+ \int_{T_R}\int_{S_w}\left(F_{S_k} + p\Psi_{k+1} - \Psi_i f_{ik} - \Psi_{k+1}\tau^n\right)\frac{\delta_s n_k}{\delta_s b_q} dSdt - \int_{T_R}\int_{S_w} \Psi_{k+1} \tau^{t^r} \frac{\delta_s t_k^r}{\delta_s b_q} dSdt \\
&+ \int_{T_R}\int_{S_w}\left(\frac{\partial F_{S_k}}{\partial v_l^g} + \tau_{kl}^A\right)\frac{\delta_s v_l^g}{\delta_s b_q} n_k dSdt + \int_{T_R}\int_{S_w}\left(M_{ij}\Psi_j V_i + \Psi_p\right)\frac{\delta_s v_n^g}{\delta_s b_q} dSdt
\end{aligned}$$

4.2 Η Διακριτή Συζυγής Διατύπωση

Αντίθετα με τη συνεχή διατύπωση, η διακριτή συζυγής μέθοδος λαμβάνει υπόψη τις διακριτοποιημένες εξισώσεις ροής $\vec{R}^C$ σε κάθε κελί C η οποία παρουσιάζεται με λεπτομέρεια στο πλήρες κείμενο της διατριβής στην αγγλική γλώσσα.

Κεφάλαιο 5

Εφαρμογή της Μεθόδου των Τεμνόμενων Κυψελών στη Ρευστοδυναμική Ανάλυση και Βελτιστοποίηση

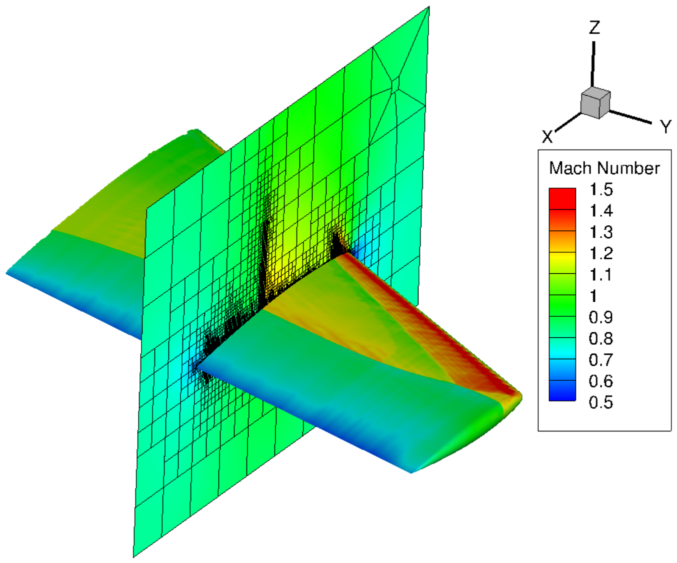
Στο κεφάλαιο αυτό παρουσιάζονται ενδεικτικά 2 εφαρμογές ρεσυτοδυναμικής ανάλυσης και 2 εφαρμογές βελτιστοποίησης σε πρακτικά προβλήματα. Η πρώτη εφαρμογή αφορά τη μελέτη της πτέρυγας ONERA M6, όπου γίνεται σύγκριση των αποτελεσμάτων του αναπτυχθέντος λογισμικού με πειραματικά δεδομένα και αποτελέσματα άλλων κωδίκων βασισμένων σε οριόδετα πλέγματα. Στη συνέχεια, μελετάται μια μηχανή κύλισης, όπου διαπιστώνεται η πλεονεκτική χρήση των καρτεσιανών πλεγμάτων. Το πρώτο πρόβλημα βελτιστοποίησης αφορά τη βαθμίδα μιας αντλίας μικτού τύπου εξόρυξης πετρελαίου. Τέλος, η συζυγής μέθοδος εφαρμόζεται στη βελτιστοποίηση υπό αβεβαιότητες μιας μικρής διαφραγματικής μη-βαλβιδοφόρου αντλίας.

Μια πληρέστερη πιστοποίηση του λογισμικού πρόλεξης της ροής και υπολογισμού των παραγώγων ευαισθησίας μέσω της συζυγούς μεθόδου παρουσιάζεται στο πλήρες κείμενο της διατριβής. Επίσης, παρουσιάζονται περισσότερες εφαρμογές της ΜΤΚ σε προβλήματα ανάλυσης και βελτιστοποίησης αναδεικνύοντας τα πλεονεκτήματα της μεθόδου έναντι άλλων τεχνικών.

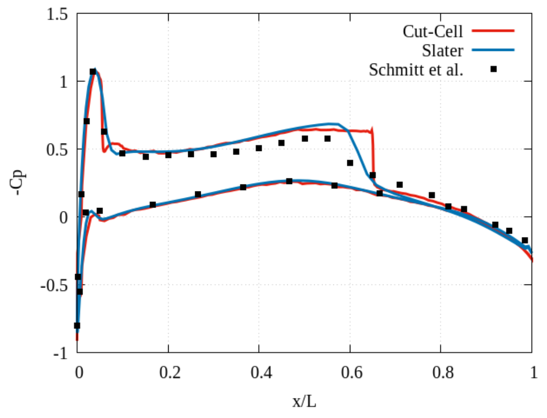
5.1 Διηχητική Ατριβής Ροή γύρω από την Πτέρυγα ONERA M6

Η πρόλεξη της ροής γύρω από την πτέρυγα ONERA M6 [15] χρησιμοποιείται συχνά για την πιστοποίηση λογισμικών αεροδυναμικής. Οι επ' άπειρο συνθήκες είναι $M_\infty = 0.84$ και $\alpha_\infty = 3.06°$. Εδώ, τα το αναπτυχθέν λογισμικό επιλύει της εξισώσεις της ατριβούς ροής σε ένα καρτεσιανό πλέγμα 1.4M κυψελών, το οποίο προσαρμόζεται στο κύμα κρούσης σχήματος 'λ' στην πλευρά υποπίεσης. Μια τομή του πεδίου ροής φαίνεται στο σχήμα 5.1.

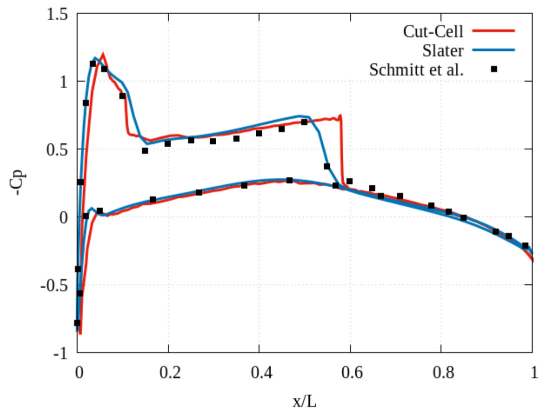
Επίσης, στο σχήμα 5.2 παρουσιάζονται διαγράμματα, όπου συγκρίνεται ο συντελεστής πίεσης με πειραματικά δεδομένα [39] σε 6 τομές κατά μήκος της πτέρυγας. Στα ίδια διαγράμματα φαίνονται και τα αποτελέσματα του λογισμικού CFL3D για την πρόλεξη της τυρβώδους ροής [41]. Η σύγκριση μεταξύ των αποτελεσμάτων αναδεικνύει την υψηλή ακρίβεια του αναπτυχθέντος λογισμικού. Συγκεκριμένα, το λογισμικό της ΜΤΚ έχει εντοπίσει σωστά τη θέση του κύματος και η διαφορά του C_p που υπολόγισε σε σχέση με τις πειραματικές μετρήσεις είναι μικρή και εφάμιλλη αυτής του πιστοποιημένου κώδικα CFL3D.



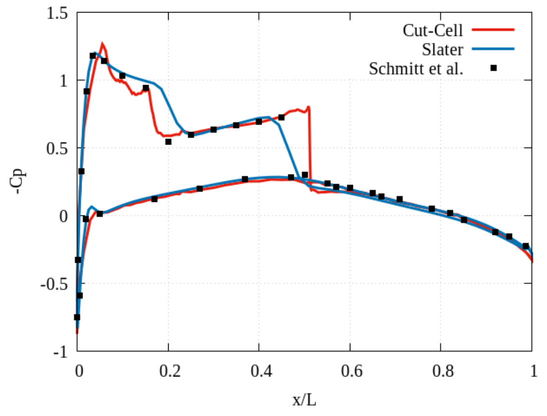
Σχήμα 5.1: Πεδίο του αριθμού Mach σε μια τομή κάθετη στην πτέρυγα. Το πλέγμα προσαρμόζεται στην περιοχή των δύο κυμάτων κρούσης καθώς και κοντά στο στερεό όριο.



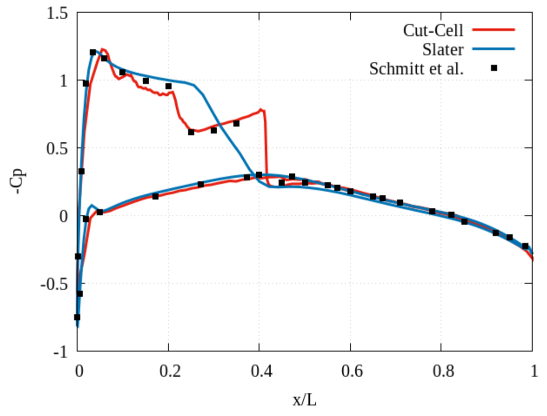
(α΄)



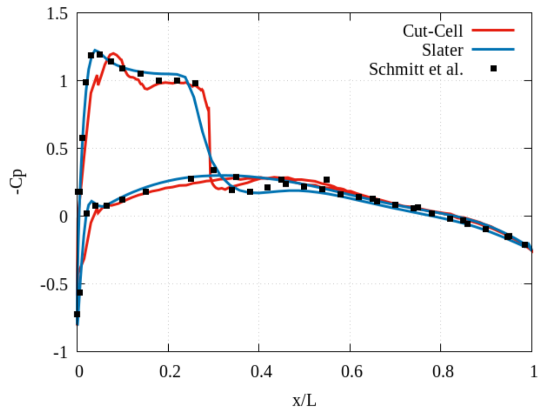
(β΄)



(γ΄)



(δ΄)



(ε΄)

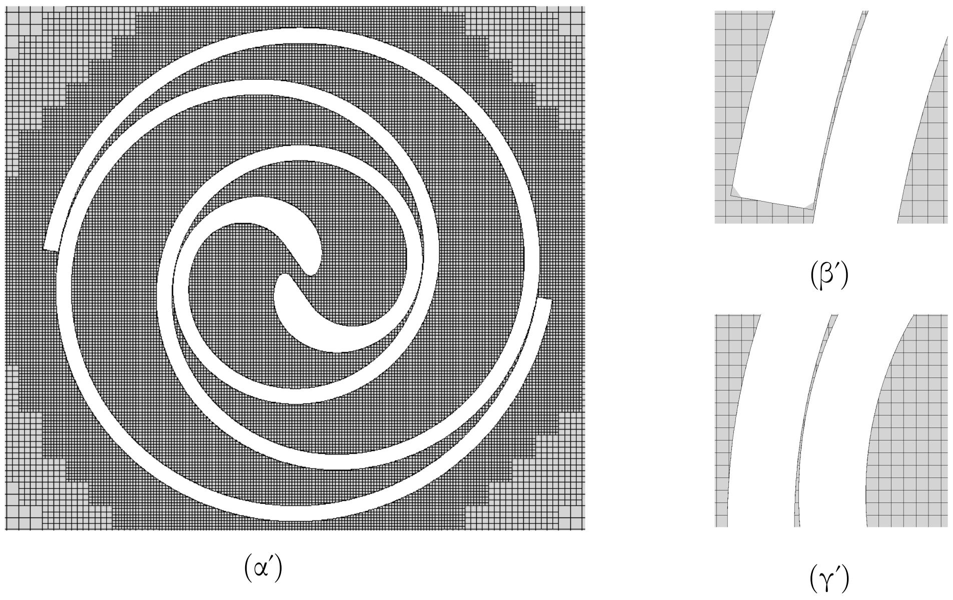
Σχήμα 5.2: Συντελεστής πίεσης σε 5 τομές κατά μήκος της πτέρυγας στις θέσεις 20%, 44%, 65%, 80%, 90% και 96%. Συγκρίνονται τα αποτελέσματα του οικείου λογισμικού (κόκκινο), του λογισμικού CFL3D (μπλε) και πειραματικές μετρήσεις (μαύρο).

5.2 Συμπιεστή Ροή εντός Μηχανής Κύλισης

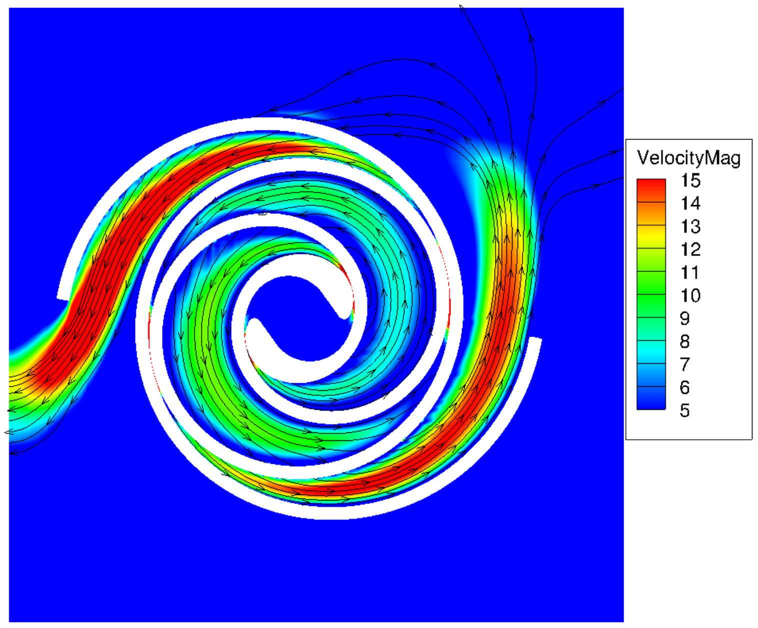
Η μηχανή κύλισης, η οποία προτάθηκε από τον Creux το 1905 [11], αποτελείται από δύο έλικες, η μια εκ των οποίων παραμένει σταθερή, ενώ η άλλη εκτελεί κυκλική κίνηση γύρω από την πρώτη. Η μηχανή μελετάται σε λειτουργία στροβίλου, όπου υψηλής πίεσης ρευστό εισέρχεται στο κέντρο της μηχανής και σπρώχνει την κινούμενη έλικα στην πορεία του προς την έξοδο. Όσο η μηχανή βρίσκεται σε κίνηση, τόσο μειώνεται η πίεση του ρευστού. Η ακριβής περιγραφή του σχήματος των ελίκων μπορεί να αναζητηθεί στο πλήρες κείμενο της διατριβής. Η μοντελοποίηση της ροής μέσα στη μηχανή, συμπεριλαμβανομένου του διακένου μεταξύ των δύο ελίκων, σπανίζει στη βιβλιογραφία λόγω των έντονα μεταβαλλόμενων θυλάκων, στα οποία αποσυμπιέζεται η ροή. Αντιθέτως, η χρήση καρτεσιανών πλεγμάτων επιτρέπει την εύκολη γένεση πλέγματος, όπως φαίνεται και στο σχήμα 5.3, ακόμα και εντός των μικρών διακένων αποφεύγοντας τη χρήση εργαλείων παραμόρφωσης πλέγματος.

Η στρωτή συμπιεστή ροή εντός της μηχανής κύλισης επιλύεται για έναν κύκλο λειτουργίας της. Η ταχύτητα κύλισης τίθεται ίση με 2000 rpm και η ολική πίεση και θερμοκρασία εισόδου είναι ίση με 40.37 bar και $90^\circ\ C$ αντίστοιχα. Το πεδίο της ταχύτητας και οι γραμμές ροής φαίνονται στο σχήμα 5.4. Επίσης, 10 στιγμιότυπα ισο-μοιρασμένα στην περίοδο λειτουργίας παρουσιάζονται στο σχήμα 5.5. Τέλος ο λόγος πίεσης και η παροχή μάζας υπολογίστηκαν 2.77 και 0.605 kg/s αντίστοιχα.

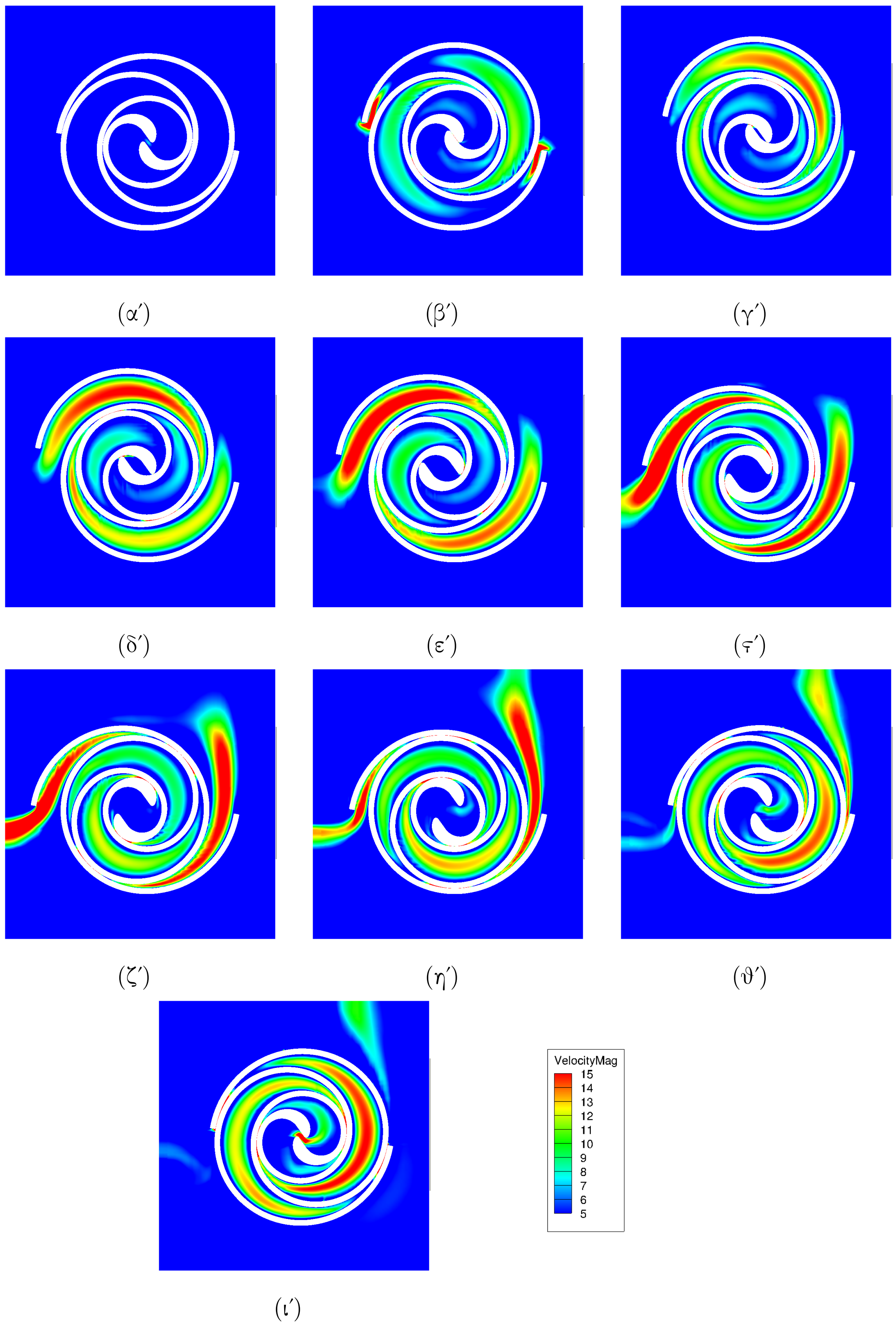
Τα αποτελέσματα της ενότητας αυτής αποτελούν μέρος του προγράμματος “ΣΥΝΕΡΓΑΣΙΑ 2011” και συγκεκριμένα του υποέργου “Ανάπτυξη ενός μικρής-κλίμακας χαμηλής-θερμοκρασίας υπερκρίσιμου οργανικού κύκλου Rankine με βελτιστοποιημένο εκτονωτή scroll και ατμοποιητή” που υλοποιήθηκε στο πλαίσιο του ΕΣΠΑ 2007-2013, και του Επιχειρησιακού προγράμματος “Ανταγωνιστικότητα & Επιχειρηματικότητα”, και συγχρηματοδοτείται από το Ευρωπαϊκό Ταμείο Περιφερειακής Ανάπτυξης της ΕΕ και από Εθνικούς Πόρους.



Σχήμα 5.3: (α′) Πλέγμα γύρω από τη μηχανή κύλισης στην αρχή της περιόδου λειτουργίας της. Λεπτομέρεια στην άκρη της έλικας (β′) και στο μικρό διάκενο μεταξύ των δύο ελίκων (γ′).



Σχήμα 5.4: Πεδίο του μέτρου της ταχύτητας και γραμμές της συμπιεστής στρωτής ροής σε ένα στιγμιότυπο λειτουργίας της μηχανής κύλισης.



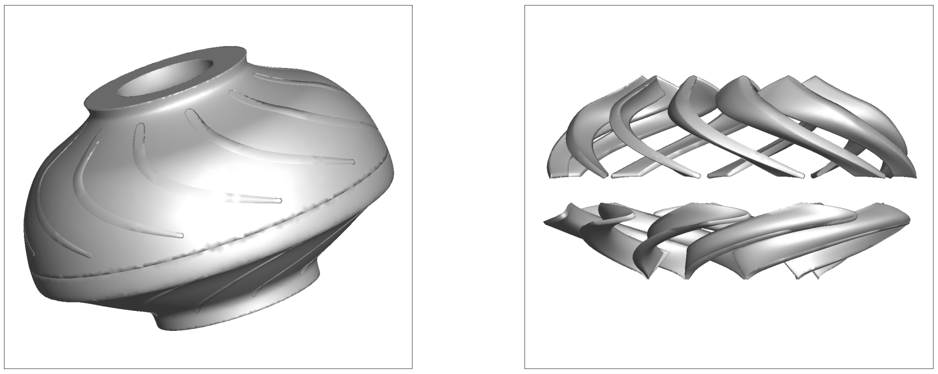
Σχήμα 5.5: Στιγμιότυπα του πεδίου του μέτρου της ταχύτητας ισο-μοιρασμένα σε μια περίοδο λειτουργίας της μηχανήω κύλισης.

5.3 Βελτιστοποίηση Βαθμίδας Αντλίας Εξώρυξης

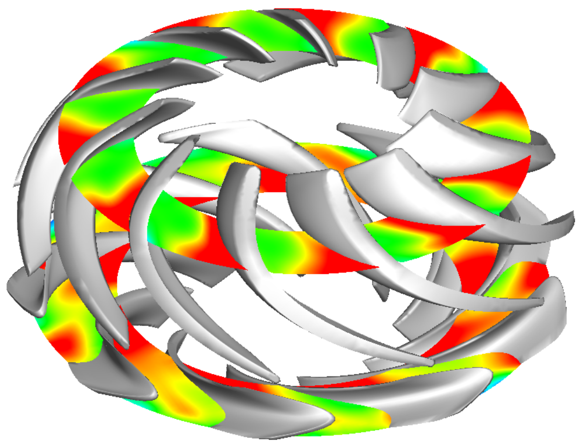
Οι ηλεκτρικές αντλίες εξόρυξης (Elactrical Submersible Pump) χρησιμοποιούνται ευρέως από τη βιομηχανία πετρελαίου καθώς πάνω από το 90% των κοιτασμάτων απαιτεί τεχνητή υποβοήθηση, προκειμένου να ανέλθει το ορυκτό στο επίπεδο της επιφάνειας [27]. Συνεπώς, η μελέτη τους καθίσταται εξαιρετικά σημαντική. Η διατριβή προτείνει τη ΜΤΚ ως ένα τρόπο αντιμετώπισης του προβλήματος αλληλεπίδρασης της κινούμενης και ακίνητης πτερύγωσης κάθε βαθμίδας. Η μελέτη που παρουσιάζεται σε αυτήν την ενότητα αποτελεί μέρος ενός έργου που χρηματοδοτήθηκε από την Schlumberger Cambridge Research Limited και έχει ως στόχο η βελτίωση της απόδοσης μιας βαθμίδας ESP μεικτής ροής, η οποία φαίνεται στο σχήμα 5.6, όπου η κατεύθυνση της ροής είναι από κάτω προς τα επάνω συναντώντας πρώτα την κινούμενη και μετά την ακίνητη πτερύγωση.

Χρησιμοποιήθηκε ένα ενιαίο καρτεσιανό πλέγμα τόσο για το στάτη όσο και για το δρομέα, το οποίο ακολουθεί τα κινούμενα πτερύγια εκτελώντας διαδοχικές αραιώσεις και πυκνώσεις. Το οικείο λογισμικό έλυσε τις μη-συνεκτικές εξισώσεις συμπιεστής ροής για τρεις περιόδους λειτουργίας της αντλίας, έτσι ώστε να αποκατασταθεί η περιοδικότητα της ροής. Στο σχήμα 5.7 φαίνονται δυο τομές του πεδίου σε ένα στιγμιότυπο της τρίτης περιόδου, όπου παρουσιάζεται το πεδίο του μέτρου της ταχύτητας γύρω από τα κινούμενα και τα ακίνητα πτερύγια.

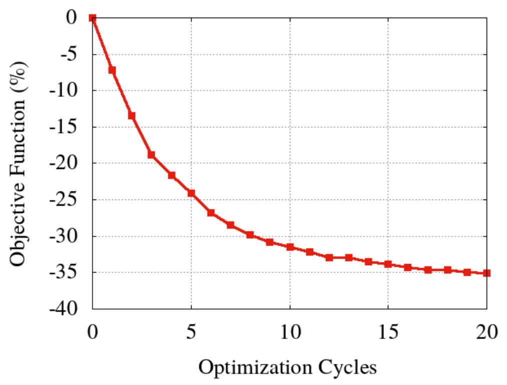
Στη συνέχεια, πραγματοποιήθηκε η βελτιστοποίηση μιας βαθμίδας της αντλίας με τη συζυγή μέθοδο. Ως μεταβλητές σχεδιασμού επιλέχθηκαν οι συντεταγμένες των σημείων που απαρτίζουν τα ακίνητα και κινούμενα πτερύγια. Στόχος της βελτιστοποίησης τέθηκε η μείωση της εφαπτομενικής ταχύτητας στην έξοδο του στάτη. Η πορεία της βελτιστοποίησης παρουσιάζεται στο σχήμα 5.8, όπου φαίνεται η ποσοστιαία μεταβολή της συνάρτησης στόχου σε κάθε κύκλο βελτιστοποίησης. Μετά το πέρας 20 κύκλων επιτεύχθηκε μείωσή της κατά 35%. Επίσης, το μέτρο της εφαπτομενικής ταχύτητας στην έξοδο της βαθμίδας πριν και μετά τη βελτιστοποίηση φαίνεται στο σχήμα 5.9, όπου διαπιστώνεται η μεγάλη μείωση του πεδίου τιμών της.


(α′) (β′)

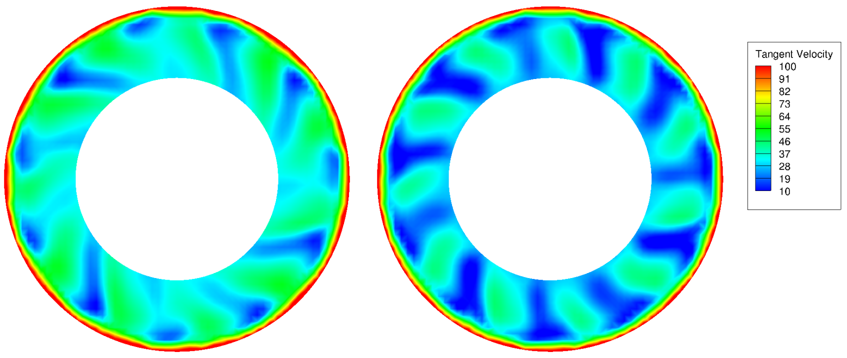
Σχήμα 5.6: (α′) Κέλυφος και (β) πτερύγια στάτη και δρομέα μιας βαθμίδας ESP.



Σχήμα 5.7: Δυο τομές του πεδίου συμπιεστής μη-συνεκτικής ροής στο ύψος του δρομέα και του στάτη. Παρουσιάζονται ισογραμμές του μέτρου της ταχύτητας.



Σχήμα 5.8: Πορεία βελτιστοποίησης της βαθμίδας με τη συζυγή μέθοδο. Μετά από 20 κύκλους βελτιστοποίησης η συνάρτηση στόχος μειώθηκε κατά 35% σε σχέση με την τιμή της στην αρχική γεωμετρία.



Σχήμα 5.9: Στιγμιαίο πεδίο του μέτρου της εφαπτομενικής ταχύτητας στην έξοδο της βαθμίδας της αρχικής (α΄) και της βελτιστοποιημένης (β΄) γεωμετρίας.

5.4 Βελτιστοποίηση μίας μικρής διαφραγματικής μη-βαλβιδοφόρου αντλίας

Οι διαφραγματικές αντλίες [42] αποτελούν υποκατηγορία των μηχανών θετικής μετατόπισης. Η λειτουργία τους βασίζεται στην περιοδική μετατόπιση του διαφράγματος, το οποίο τοποθετείται στη μια πλευρά ενός χωρίου με μια είσοδο και μια έξοδο. Η αντλία που μελετάται χρησιμοποιείται σε βιοϊατρικές εφαρμογές, όπου αποφεύγεται η χρήση κινούμενων μερών όπως οι βαλβίδες. Τη θέση τους καταλαμβάνουν δύο διαχύτες, όπως φαίνεται στο σχήμα 5.10. Έτσι, όταν το διάφραγμα μετατοπίζεται με τέτοιο τρόπο ώστε να αυξάνεται ο όγκος του κεντρικού χωρίου η ροή εισέρχεται στην αντλία κυρίως από τον διαχύτη εισόδου. Αντίθετα, όταν ο όγκος μειώνεται, η ροή προτιμά τον διαχύτη της εξόδου. Αναπόφευκτα, οι αντλίες αυτές αναρροφούν ένα μέρος της παροχής από την έξοδο. Σκοπός είναι η ελαχιστοποίηση της αναρρόφησης ρευστού από την έξοδο και η μεγιστοποίηση της παροχής που διοχετεύεται.

Οι μεταβλητές σχεδιασμού του προβλήματος ελέγχουν την κίνηση του διαφράγματος, επιτρέποντας εξαιρετικά μεγάλες μετατοπίσεις. Στο σχήμα 5.11 εικονίζεται η θέση ισορροπίας και η ελάχιστη κάτω θέση του διαφράγματος σε μια διαμήκη τομή της αντλίας. Έτσι, η χρήση της ΜΤΚ είναι πλεονεκτική, καθώς η ποιότητα του πλέγματος δεν επηρεάζεται από την έντονη κίνηση του στερεού τοιχώματος. Η περιοδικότητα του φαινομένου αποκαθίσταται μετά από 3 περιόδους. Το σχήμα 5.12 παρουσιάζει 4 χρονικά ισαπέχουσες χρονικές στιγμές κατά τη διάρκεια της τέταρτης περιόδου λειτουργίας.

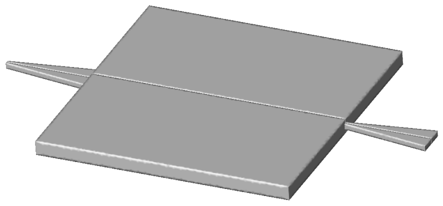
Κατά τη βελτιστοποίηση της αντλίας λήφθηκαν υπόψη κατασκευαστικές και λειτουργικές ατέλειες, οι οποίες μοντελοποιούνται ως αβεβαιότητες στις τιμές των μεταβλητών σχεδιασμού που ακολουθούν την κανονική στατιστική κατανομή. Αυτή η διακύμανση αποτυπώνεται στη μέτρηση των προαναφερθέντων μεγεθών ενδιαφέροντος αναρρόφησης (Q_{bf}) και παροχής (Q_{net}) μέσω του υπολογισμού της μέσης τιμής και τυπικής απόκλισης της απόκρισής τους. Έτσι, οι μεταβλητές σχεδιασμού διαμορφώνονται ως γραμμικός συνδυασμός των ανωτέρω στατιστικών μεγεθών,

$$\begin{aligned} F_1 =& w_{11}\mu_{Q_{bf}} + w_{12}\sigma_{Q_{bf}} \\ F_2 =& w_{21}\mu_{Q_{net}} + w_{22}\sigma_{Q_{net}} \end{aligned}$$

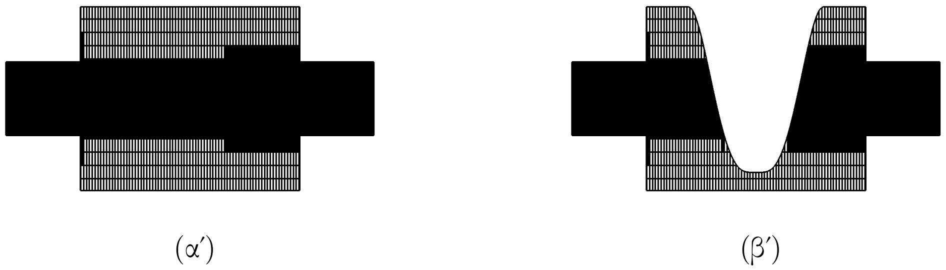
Εδώ επιλέγονται οι τιμές $w_{11} = +1$, $w_{12} = +1$, $w_{21} = +1$ και $w_{22} = -1$ καθώς επιθυμείται η βελτίωση της μέσης τιμής και η μείωση της τυπικής απόκλισης της κάθε συνάρτησης στόχου. Οι τιμές των στατιστικών μεγεθών υπολογίζονται μέσω της μεθόδου πολυωνυμικού χάους (Polynomial Chaos Expansion) [13], [48].

Τέλος, η βελτιστοποίηση πραγματοποιήθηκε μέσω της πλατφόρμας βελτιστοποίησης γενικής χρήσης EASY (Evolutional Algorithms SYstem) [1] υποβοηθούμενη από τη συζυγή μέθοδο. Το λογισμικό EASY βασίζεται σε εξελικτικούς αλγορίθμους προκειμένου να υπολογίσει το μέτωπο μη-κυριαρχούμενων λύσεων (Pareto front) [23]. Λόγω του μεγάλου αριθμού κλίσεων του λογισμικού επίλυσης της ροής που απαιτείται από τη μέθοδο PCE και τον εξελικτικό αλγόριθμο, χρησιμοποιούνται μεταπρότυπα [25]. Επίσης, σε κάθε γενιά οι καλύτερες λύσεις βελτιώνονται περαιτέρω με τη χρήση της κλίσης των συναρτήσεων στόχων [24], όπως φαίνεται και στο σχήμα 5.13β′. Το τελικό μέτωπο μη κυριαρχούμενων λύσεων εικονίζεται στο σχήμα 5.13α′. Επίσης, το σχήμα 5.14 παρουσιάζει τις τιμές των μεγεθών ενδιαφέροντος για τα δύο άκρα του μετώπου.

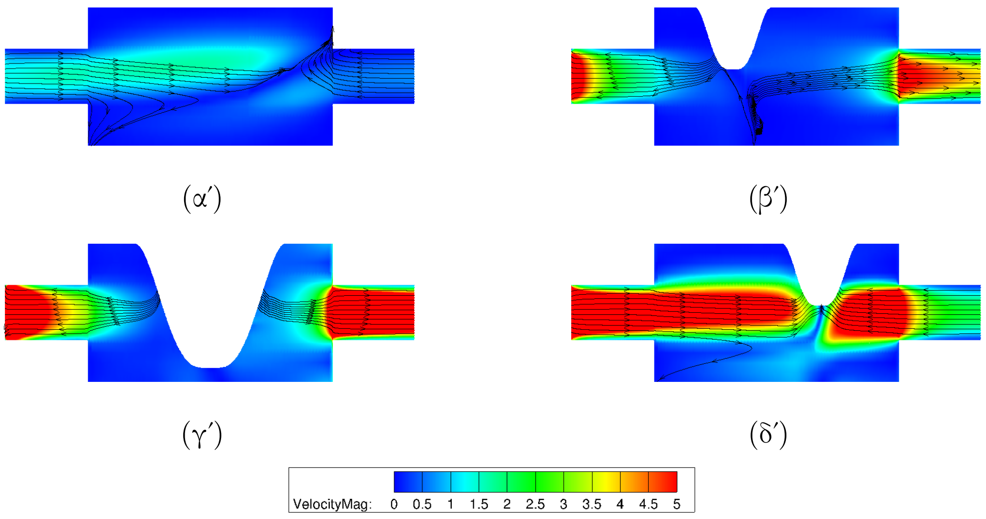
Τα αποτελέσματα αυτής της ενότητας είναι μέρος της εργασίας με τίτλο "Σχεδιασμός-Βελτιστοποίηση Διαφραγματικών Αντλιών παρουσία Λειτουργικών/Κατασκευαστικών Αβεβαιοτήτων, με τη Μέθοδο των Τεμνομένων Κυψελών και της Ανάπτυξης Πολυωνυμικού Χάους" και ανήκει σε έργο που συγχρηματοδοτείται από την Ελλάδα και την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) μέσω του Επιχειρησιακού Προγράμματος "Ανάπτυξη Ανθρώπινου Δυναμικού, Εκπαίδευση και Διά Βίου Μάθηση" που τιτλοφορείται ως "Υποστήριξη Ερευνητών με Έμφαση στους Νέους Ερευνητές".



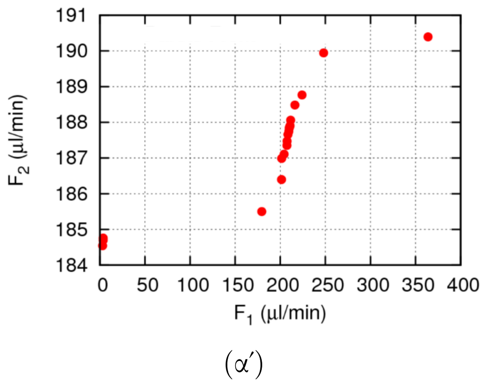
Σχήμα 5.10: Η διαφραγματική αντλία αποτελείται από ένα κεντρικό χωρίο, εξοπλισμένο με ένα κινούμενο διάφραγμα στην άνω επιφάνειά του και δύο διαχύτες. Η ροή διέρχεται από αριστερά προς τα δεξιά.

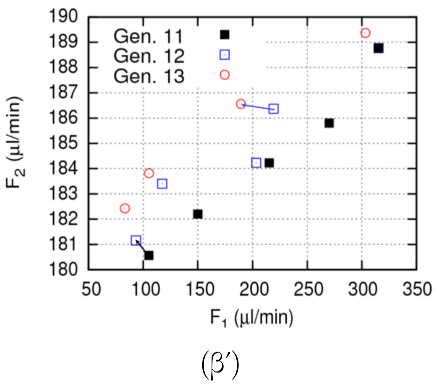


Σχήμα 5.11: Καρτεσιανό πλέγμα εντός της αντλίας όταν το διάφραγμα βρίσκεται (α′) στο σημείο ηρεμίας και (β′) στη θέση μέγιστης μετατόπισής. Άξονες σε διαφορετική κλίμακα.

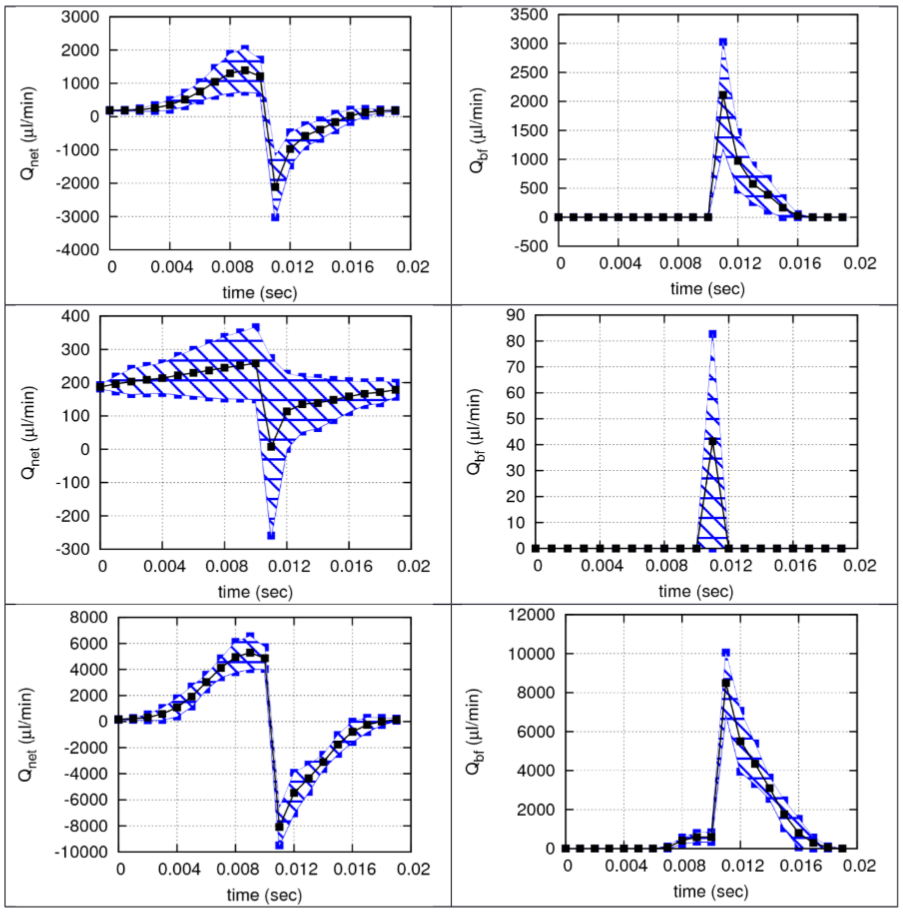


Σχήμα 5.12: Στρωτή ασυμπίεστη ροή εντός της αντλίας. Πεδίο μέτρου ταχύτητας σε 4 χρονικά ισαπέχοντα στιγμιότυπα σε μια περίοδο λειτουργείας της. Άξονες σε διαφορετική κλίμακα.





Σχήμα 5.13: (α′) Τελικό πεδίο μη-κυριαρχούμενων λύσεων (β′) Μέρος του μετώπου τριών συνεχόμενων γενεών. Το βέλος υποδεικνύει τη βελτίωση που επέφερε η χρήση των παραγώγων ευαισθησίας.



Σχήμα 5.14: Η χρονική μεταβολή των Q_{net} και Q_{bf} για μέγιστη F_2 (πάνω), ελάχιστη F_1 (μέσο) και αρχική (κάτω) λύση. Η μαύρη καμπύλη αντιστοιχεί στη μέση τιμή των μεγεθών και η μπλε περιοχή σηματοδοτεί ζώνη πάχους $\pm 3\sigma$.

Κεφάλαιο 6

Επίλογος

6.1 Ανακεφαλαίωση-Συμπεράσματα

Στόχος της διδακτορικής διατριβής ήταν η ανάπτυξη ενός ολοκληρωμένου συνόλου υπολογιστικών εργαλείων για τη ρευστοδυναμική ανάλυση και βελτιστοποίηση σε πρακτικές εφαρμογές. Για το σκοπό αυτό προτάθηκε η χρήση της ΜΤΚ, η οποία εξασφαλίζει την αυτόματη δημιουργία πλέγματος ανεξαρτήτως της πολυπλοκότητας του υπολογιστικού χωρίου, διατηρώντας παράλληλα την ακρίβεια επίλυσης των εκάστοτε ΜΔΕ. Έτσι, αναπτύχθηκε λογισμικό γένεσης καρτεσιανών πλεγμάτων, τα οποία τέμνονται από τα κινούμενα ή μη στερεά όρια σχηματίζοντας τεμνόμενες κυψέλες καθώς και λογισμικό πρόλεξης της συμπιεστής και ασυμπίεστης ροής. Το λογισμικό πιστοποιήθηκε σε μια σειρά από εφαρμογές εσωτερικής και εξωτερικής αεροδυναμικής αναδεικνύοντας τη δυνατότητα της ΜΤΚ να επιλύει τις εξισώσεις της μη-συνεκτικής και στρωτής ροής με υψηλή ακρίβεια εφάμιλλη των τεχνικών που βασίζονται σε οριόδετα πλέγματα. Στη συνέχεια, η διατριβή κατέδειξε τα πλεονεκτήματα της ΜΤΚ στη βελτιστοποίηση μορφής περίπλοκων γεωμετριών, καθώς μπορεί να ανταποκριθεί σε κάθε ενδιάμεση λύση που προκύπτει κατά τη διάρκεια της βελτιστοποίησης. Η συνεχής και διακριτή συζυγής μέθοδος χρησιμοποιήθηκαν για τον υπολογισμό των παραγώγων ευαισθησίας σε περιπτώσεις μόνιμης ή μη-μόνιμης συμπιεστής και ασυμπίεστης ροής. Το λογισμικό εφαρμόστηκε σε 3Δ βιομηχανικά προβλήματα βελτιστοποίησης με ή χωρίς αβεβαιότητες καταδεικνύοντας την ευελιξία και υψηλή απόδοσή του.

6.2 Στοιχεία Πρωτοτυπίας

- Προτάθηκε και δοκιμάστηκε επιτυχώς νέος αλγόριθμος για τη δημιουργία των τεμνόμενων κυψελών, ικανός να υπολογίσει οποιαδήποτε τομή του καρτεσιανού πλέγματος με τη στερεή γεωμετρία.

- Αναπτύχθηκε μια νέα μέθοδος αντιμετώπισης των κυψελών που καλύπτονται ή αποκαλύπτονται από το στερεό σώμα λόγω της κίνησής του πάνω από το καρτεσιανό πλέγμα. Αυτή δοκιμάστηκε σε περιπτώσεις έντονων μετατοπίσεων των στερεών ορίων και διαπιστώθηκε η διατήρηση της μάζας, ορμής και ενέργειας.

- Η εφαρμογή της συνεχούς συζυγούς μεθόδου σε καρτεσιανά πλέγματα τεμνόμενων κυψελών παρουσιάστηκε για πρώτη φορά στη βιβλιογραφία αποδεικνύοντας την καταλληλότητά της στη βελτιστοποίηση μορφής.

- Η διακριτή συζυγής μέθοδος εφαρμόστηκε στη ΜΤΚ για πρώτη φορά για την αντιμετώπιση προβλημάτων συνεκτικής και μη-μόνιμης ροής. Επίσης, δόθηκε έμφαση στη σωστή διαφόριση του αλγορίθμου κατασκευής των τεμνόμενων κυψελών καθώς και στον αλγόριθμο συγχώνευσης των καλυπτόμενων ή απόκαλυπτόμενων κυψελών σε προβλήματα βελτιστοποίησης κινούμενων σωμάτων.

Bibliography

[1] The EASY (Evolutionary Algorithms SYstem) software. http://velos0.ltt.mech.ntua.gr/EASY.

[2] W. Anderson and V. Venkatakrishnan. Aerodynamic Design Optimization on Unstructured Grids with a Continuous Adjoint Formulation. *Computers & Fluids*, 28(4):443–480, 1999.

[3] T. Barth and D. Jespersen. The Design and Application of Upwind Schemes on Unstructured Meshes. In *27th Aerospace Sciences Meeting*, 1989.

[4] S. Bayyuk, K. Powell, and B. van Leer. A Simulation Technique for 2-D Unsteady Inviscid Flows Around Arbitrarily Moving and Deforming Bodies of Arbitrary Geometry. July 1993.

[5] J. Benek, J. Steger, and F. Dougherty. A Chimera Grid Scheme. page 59–69, 1983.

[6] M. Berger and M. Aftosmis. Progress Towards a Cartesian Cut-Cell Method for Viscous Compressible Flow. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2012.

[7] F. Chinesta, R. Keunings, and A. Leygue. *The Proper Generalized Decomposition for Advanced Numerical Simulations, A Primer*. Springer International Publishing, Nantes, France, 2014.

[8] A. Chorin. A Numerical Method for Solving Incompressible Viscous Flow Problems. *Journal of Computational Physics*, 2(1):12–26, 1967.

[9] D. Clarke, M. Salas, and H. Hassan. Euler Calculations for Multielement Airfoils Using Cartesian Grids. *AIAA Journal*, 24(3):353–358, 1986.

[10] W. Coirier and K. Powell. Solution-Adaptive Cartesian Cell Approach for Viscous and Inviscid Flows. *AIAA Journal*, 34(5):938–945, 1996.

[11] L. Creux. Rotary Engine, U.S. Patent 801,182, October 1905.

[12] A. Dadone and B. Grossman. Efficient Fluid Dynamic Design Optimization Using Cartesian Grids. In *16th AIAA Computational Fluid Dynamics Conference*, 2003.

[13] M. Eldred and J. Burkardt. Comparison of Non-Intrusive Polynomial Chaos and Stochastic Collocation Methods for Uncertainty Quantification. *47th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, January 2009.

[14] R. Fletcher and C. M. Reeves. Function Minimization by Conjugate Gradients. *The Computer Journal*, 7(2):149–154, January 1964.

[15] North Atlantic Treaty Organization. Advisory Group for Aerospace Research and Development. Fluid Dynamics Panel. Working Group 07. *Test Cases for Inviscid Flow Field Methods: Report of Fluid Dynamics Panel Working Group 07*. AGARD advisory report. AGARD, 1985.

[16] R. Gaffney, H. Hassan, and M. Salas. Euler Calculations for Wings Using Cartesian Grids. *AIAA Paper 87-0356*, 1987.

[17] M. Giles, M. Duta, J. Muller, and N. Pierce. Algorithm Developments for Discrete Adjoint Methods. *AIAA Journal*, 41(2):198–205, 2003.

[18] D. Hartmann, M. Meinke, and W. Schröder. A Strictly Conservative Cartesian Cut-Cell Method for Compressible Viscous Flows on Adaptive Grids. *Computer Methods in Applied Mechanics and Engineering*, 200(9):1038–1052, 2011.

[19] C. Hinterberger and M. Olesen. Automatic Geometry Optimization of Exhaust Systems Based on Sensitivities Computed by a Continuous Adjoint CFD Method in OpenFOAM. April 2010.

[20] C Hirt, A. Amsden, and J. Cook. An Arbitrary Lagrangian-Eulerian Computing Method for all Flow Speeds. *Journal of Computational Physics*, 14(3):227–253, 1974.

[21] H. Ji, F. Lien, and E. Yee. A New Adaptive Mesh Refinement Data Structure with an Application to Detonation. *Journal of Computational Physics*, 229(23):8981–8993, 2010.

[22] H. Ji, F. Lien, and E. Yee. Numerical Simulation of Detonation Using an Adaptive Cartesian Cut-Cell Method Combined with a Cell-Merging Technique. *Computers & Fluids*, 39:1041–1057, June 2010.

[23] D. Kapsoulis. *Low-Cost Metamodel-Assisted Evolutionary Algorithms with Application in Shape Optimization in Fluid Dynamics*. PhD thesis, National Technical University of Athens, 2019.

[24] D. Kapsoulis, K. Tsiakas, X. Trompoukis, V. Asouti, and K. Giannakoglou. A PCA-assisted Hybrid Algorithm Combining EAs and Adjoint Methods for CFD-based Optimization. *Applied Soft Computing*, 73:520–529, 2018.

[25] M. Karakasis and K. Giannakoglou. On the use of metamodel-assisted, multi-objective evolutionary algorithms. *Engineering Optimization*, 38(8):941–957, 2006.

[26] M. Kirkpatrick, S. Armfield, and J. 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*, 184(1):1–36, 2003.

[27] E. Ladopoulos. Four-dimensional Petroleum Exploration & Non-linear ESP Artificial Lift by Multiple Pumps for Petroleum Well Development. *Universal Journal of Hydraulics*, 3:1–14, 01 2015.

[28] L. Landau and E. Lifshitz. *Fluid Mechanics*, volume 6 of *Course of Theoretical Physics*. Pergamon Press, 1987.

[29] C. Merkle. Time-Accurate Unsteady Incompressible Flow Algorithms Based on Artificial Compressibility. In *8th Computational Fluid Dynamics Conference*, 1987.

[30] R. Mittal and G. Iaccarino. Immersed Boundary Methods. *Annual Review of Fluid Mechanics*, 37(1):239–261, 2005.

[31] S. Murman, M. Aftosmis, M. Berger, and D. Kwak. Implicit Approaches for Moving Boundaries in a 3-D Cartesian Method. In *41st Aerospace Sciences Meeting and Exhibit*, February 2003.

[32] M. Nemec and M. Aftosmis. Adjoint Sensitivity Computations for an Embedded-Boundary Cartesian Mesh Method and CAD Geometry. volume 227, pages 2724–2742, 2008.

[33] E. Papoutsis-Kiachagias. *Adjoint Methods for Turbulent Flows, Applied to Shape or Topology Optimization and Robust Design*. PhD thesis, National Technical University of Athens, 2013.

[34] O. Pironneau. *Optimal Shape Design for Elliptic Systems*. Springer, Berlin, Heidelberg, 1982.

[35] J. Purvis and J. Burkhalter. Prediction of Critical Mach Number for Store Configurations. *AIAA Journal*, 17(11):1170–1177, 1979.

[36] J. Quirk. *An Adaptive Grid Algorithm for Computational Shock Hydrodynamics*. PhD thesis, Cranfield University, 1991.

[37] J. Quirk. An Alternative to Unstructured Grids for Computing Gas Dynamic Flows around Arbitrarily Complex Two-Dimensional Bodies. *Computers & Fluids*, 23(1):125–142, 1994.

[38] P. Roe. Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes. *Journal of Computational Physics*, 43(2):357–372, 1981.

[39] V. Schmitt and F. Charpin. Pressure Distributions on the ONERA M6 Wing at Transonic Mach Numbers. Report of the Fluid Dynamics Panel Working Group 04, AGARD AR 138, May 1979.

[40] L. Schneiders, C. Günther, M. Meinke, and W. Schröder. An Efficient Conservative Cut-Cell Method for Rigid Bodies Interacting with Viscous Compressible Flows. *Journal of Computational Physics*, 311:62–86, 2016.

[41] J. Slater. https://www.grc.nasa.gov/WWW/wind/valid/m6wing/m6wing01/m6wing01.html.

[42] E. Stemme and Stemme G. A Valveless Diffuser/Nozzle-Based Fluid Pump. *Sensors and Actuators A: Physical*, 39(2):159–167, 1993.

[43] I. Sutherland and G. Hodgman. Reentrant Polygon Clipping. *Commun. ACM*, 17(1):32–42, January 1974.

[44] E. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*. January 2009.

[45] V. Venkatakrishnan. On the Accuracy of Limiters and Convergence to Steady State Solutions. In *31st Aerospace Sciences Meeting*, 1993.

[46] C. Vezyris, E. Papoutsis-Kiachagias, and K. Giannakoglou. On the Incremental Singular Value Decomposition Method to Support Unsteady Adjoint-Based Optimization. *International Journal for Numerical Methods in Fluids*, 91(7):315–331, 2019.

[47] B. Wedan and J. South. A Method for Solving the Transonic Full-Potential Equation for General Configurations. 1983.

[48] D. Xiu and G. Karniadakis. The Wiener–Askey Polynomial Chaos for Stochastic Differential Equations. *SIAM Journal on Scientific Computing*, 24(2):619–644, 2002.

[49] 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*, 156(2):209–240, 1999.