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

h-Refinement of Tetrahedral Element Meshes. Software Programming and CFD Applications

Diploma Thesis

Dimitrios Chondroudakis

Advisor: Kyriakos C. Giannakoglou, Professor NTUA

Athens, 2025

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor, Professor Kyriakos C. Giannakoglou for his constant guidance and support throughout the entire course of my diploma thesis. His experience and approach to problem-solving as well as his willingness to help me at any time have truly been invaluable. I would also like to thank him for being the most inspiring figure during my studies and for making me, through his enlightening lectures, view engineering from a different perspective.

Special thanks are also due to Dr. Xenofon Trompoukis for his inspiring guidance and assistance on any technical issue I faced during the entire course of this thesis. I would also like to thank every member of the PCOpt Unit that I met and worked with for their willingness to help me at any time.

Lastly, I want to thank my family for their constant support throughout the entire course of my studies and for always believing in me. I am thankful for all the friends I made along the way and all the unforgettable memories we created.

National Technical University of Athens

School of Mechanical Engineering

Fluids Section

Parallel CFD & Optimization Unit

h-Refinement of Tetrahedral Element Meshes. Software Programming and CFD Applications

Diploma Thesis

Dimitrios Chondroudakis

Advisor: Kyriakos C. Giannakoglou, Professor NTUA

Athens, 2025

Abstract

Over the past few decades, Computational Fluid Dynamics (CFD) has experienced a significant growth across various fields, in both industrial and research applications. This rise is mainly driven by the continuous increase in computational power of modern systems. With this, the demand for more complex and large-scaled simulations has also risen. These simulations' increased computational cost emerge from the need of very fine meshes. To address this problem, mesh adaptation methods have emerged. These methods aim to solve this problem by dynamically refining the mesh in regions where high solution accuracy is necessary. This diploma thesis explores the theory and practical application of mesh adaptation, particularly the h-refinement, when applied to tetrahedral element meshes. The principles governing the refinement of tetrahedral meshes are established, along with the criteria used to identify regions requiring higher resolution. An algorithm that incorporates these principles with an emphasis on efficient data and memory management is detailed.

In order to apply the proposed algorithm to actual CFD aplications, a dedicated software tool is developed in the C++ programming language. This software is designed to integrate with PUMA, the in-house GPU accelerated flow analysis software of the PCOpt Unit/NTUA, which is used for all CFD simulations performed in this thesis.

Using this software, the presented refinement algorithm is applied to two common CFD benchmark problems in order to validate and showcase its efficiency. The first case involves an internal aerodynamic problem, specifically the flow inside a channel with a small bump. Two variants of this problem are examined; one with a

symmetrical bump and one with an added peak on its surface in order to introduce 3D features to the flow. The second case involves an external aerodynamic problem, particularly the flow around a sphere. This study also includes two variants with different freestream Mach numbers and is inspired by a similar study that examines the transient version of the flow without mesh refinement. The results of these simulations are presented and compared with the findings of then reference study. In every simulation that was performed, the flow is transonic and is followed by the formation of a shock wave. The algorithm concentrated the refinement of the given mesh on the region of the shock.

Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Μηχανολόγων Μηχανικών

Τομέας Ρευστών

Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης

η-Εμπλουτισμός Πλεγμάτων Τετραεδρικών Στοιχείων. Προγραμματισμός λογισμικού και εφαρμογές στην Υπολογιστική Ρευστοδυναμική

Διπλωματική Εργασία

Δημήτριος Χονδρουδάκης

Επιβλέπων: Κυριάκος Χ. Γιαννάκογλου, Καθηγητής ΕΜΠ

Αθήνα, 2025

Περίληψη

Τα τελευταία χρόνια, η Υπολογιστική Ρευστοδυναμική (ΥΡΔ) έχει γνωρίσει σημαντική ανάπτυξη σε διάφορους τομείς, τόσο σε βιομηχανικές εφαρμογές όσο και στον τομέα της έρευνας. Η ανάπτυξη αυτή οφείλεται κυριώς στη συνεχή εξέλιξη των μοντέρνων υπολογιστικών συστημάτων. Η ανάπτυξη αυτή συνοδεύεται και από αύξηση της ζήτησης σε πιο σύνθετες και μεγάλες προσομοιώσεις. Τα προβλήματα αυτά συνοδεύονται από επιπλέον αυξημένο υπολογιστικό κόστος λόγω της ανάγκης πλεγμάτων υψηλής ποιότητας για την πραγματοποίησή τους. Για την αντιμετώπιση του προβλήματος αυτού αναπτύχθηκαν διάφορες μέθοδοι προσαρμογής πλεγμάτων. Οι μέθοδοι αυτοί, στοχεύουν στην πύκνωνση του πλέγματος μόνο σε περιοχές όπου απαιτείται υψηλή ακρίβεια λύσης. Αυτή η διπλωματική εργασία ασχολείται με θεωρία και εφαρμογές της προσαρμογής πλεγμάτων στην υπό εξέλιξη λύση, συγκεκριμένα του h-εμπλουτισμού, όπως αυτός εφαρμόζεται σε πλέγματα τετραεδρικών στοιχείων. Διατυπώνονται οι βασικές αρχές του εμπλουτισμού στα τετράεδρα μαζί με τα κριτήρια ανίχνευσης των περιοχών του πλέγματος στις οποίες εφαρμόζονται. Επίσης διατυπώνεται ένας αλγόριθμος που εφαρμόζει την θεωρία αυτή, με έμφαση στην αποδοτική διαχείριση μνήμης και δεδομένων.

Προκειμένου ο αλγόριθμός αυτός να δοκιμαστεί σε εφαρμογές ΥΡΔ, προγραμματίστηκε ένα λογισμικό στη γλώσσα προγραμματισμού C++. Το λογισμικό αυτό σχεδιάστηκε προκειμένου να συνεργάζεται εξωτερικά με το λογισμικό επίλυσης ροών PUMA της ΜΠΥΡΔ&Β του ΕΜΠ, το οποίο χρησιμοποιείται και σε όλες τις προσομοιώσεις που

πραγματοποιήθηκαν σε αυτή την εργασία.

Χρησιμοποιώντας το λογισμικό που προγραμματίστηκε, εφαρμόζεται ο αλγόριθμος προσαρμογής σε δύο χαρακτηριστικά προβλήματα αναφοράς προκειμένου να αποτιμηθεί η λειτουργία του και η αποδοτικότητά του. Η πρώτη περίπτωση που εξετάζεται αφορά ένα πρόβλημα εσωτερικής αεροδυναμικής, συγκεκριμένα τη ροή εντός ενός καναλιού με ένα εμπόδιο. Εξετάζονται δύο παραλλαγές του προβλήματος: μια με ένα συμμετρικό εμπόδιο και μια με μια κορυφή στην επιφάνεια του εμποδίου με στόχο τη εισαγωγή 3Δ χαρακτηριστικών στη ροή. Η δεύτερη περίπτωση αφορά ένα πρόβλημα εξωτερικής αεροδυναμικής και συγκεκριμένα τη ροή γύρω από μια σφαίρα. Η μελέτη αυτή περιλαμβάνει επίσης δύο παραλλαγές, με διαφορετικούς αριθμούς Mach της επ΄ άπειρο ροής και βασίζεται σε μια αντίστοιχη μελέτη της μη-μόνιμης εκδοχής του προβλήματος. Τα αποτελέσματα αυτών των προσομοιώσεων παρουσιάζονται και συγκρίνονται με αυτή της προϋπάρχουσας μελέτης. Σε όλες τις προσομοιώσεις που πραγματοποιήθηκαν, η ροή είναι διηχητική και συνοδεύεται από τον σχηματισμό ενός κρουστικού κύματος στο οποίο ο αλγόριθμος εστίασε τον εμπλουτισμό του εκάστοτε πλέγματος.

Abbreviations

C_D	Drag Coefficient
CFD	Computational Fluid Dynamics
C_P	Pressure Coefficient
GPU	Graphics Processing Unit
NTUA	National Technical University of Athens
PCOpt	Parallel CFD and Optimization Unit
PUMA	Parallel Unstructured Multirow and Adjoint

ΕΜΠ	Εθνικό Μετσόβιο Πολυτεχνείο
ΜΠΥΡΔ&Β	Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης
ΥΡΔ	Υπολογιστική Ρευστοδυναμική

Contents

Chapter 1

Introduction

Over the past few decades, Computational Fluid Dynamics (CFD) has had a significant contribution to scientific research and engineering applications. The purpose of CFD is to compute the flow variables (pressure, velocity, density, e.t.c) by solving the equations which govern fluid flow. These equations are derived from basic principles of fluid mechanics, mainly conservation laws. These equations don't have general analytical solutions for most practical problems, therefore the use of numerical methods is necessary. This used to be a problem in the past, but due to the great increase in power of modern computational systems and advancements made in numerical methods, CFD has seen a great prosperity. CFD was initially used mainly for aerospace applications, but has since expanded to a broad range of fields such as the automotive industry, environmental modeling and biomedical engineering. Because the equations are being solved numerically through discretization schemes, the use of a mesh is essential.

A mesh is the basis of CFD and meshing is a fundamental process that plays a significant role in the accuracy and reliability of a simulation's results. Therefore, a well-constructed mesh is needed not only to ensure a correct convergence of the numerical method used, but also to capture all the important flow phenomena that take place in the physical problem. Thus, an important task in every CFD simulation is the generation and use of an appropriate mesh. This mesh needs to have a high enough resolution in areas where important flow phenomena (such as boundary layers, vortices or shock waves) take place in order to get accurate results. However, in practice it is often not possible to know beforehand the exact location where these phenomena appear and therefore high mesh resolution is needed. Solution-based mesh adaptation is a popular method that aims to solve this problem by enhancing the resolution and quality of the mesh, specifically in such areas of interest.

Mesh adaptation is a process that is performed regularly, in line with the iterations of the numerical method used to solve the flow equations. It utilizes the progress of

the solution and predicts which areas of the mesh need an increase or decrease in resolution. The goal is to enhance the mesh in a way that balances computational cost and solution accuracy without any knowledge of the final flow field prior to the simulation, thus eliminating the danger of potential adjustments, in case of a poorly constructed initial mesh.

1.1 Mesh Types

The purpose of a mesh is to divide the geometry of a problem into discrete elements in order to apply a numerical method. For CFD applications this translates to solving the flow equations. There are two main types of meshes:

1. **Structured meshes**
2. **Unstructured meshes**

Structured meshes utilize implicit connectivity whose structure allows for easy identification of their elements and nodes. Structured meshes are made up of quadrilaterals (in 2D) or hexahedrons (in 3D). Their main advantages include: better numerical convergence, a high degree of quality and control as well as reduced memory requirements. The latter is because each element (or node) can be described by the use of only three (in 3D) indices (i, j, k) and all their connectivity relations derive from specific, notable, patterns formed by those indices.

A typical example of a structured mesh is shown in figure 1.1.

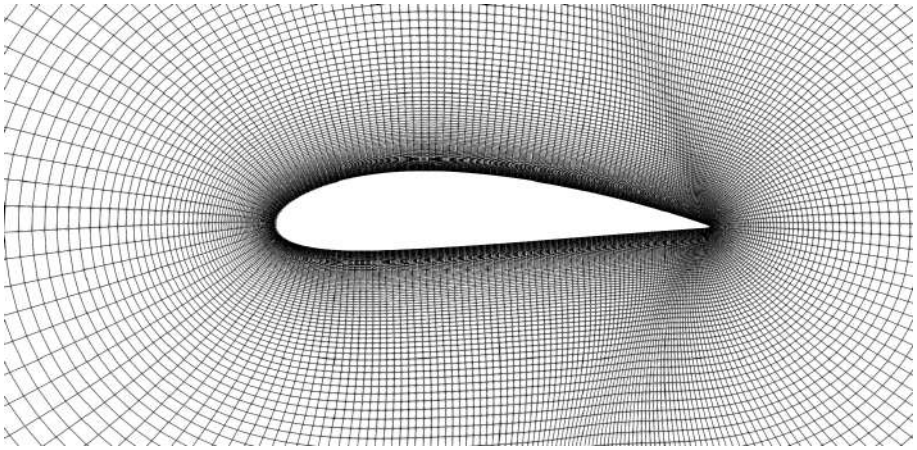


Figure 1.1: *A typical 2D structured mesh around an airfoil.*

Unstructured meshes, on the other hand, lack implicit connectivity. Instead they utilize a more complex topological data structure to represent the relations between their elements. Therefore the memory requirements for an unstructured mesh are substantially larger, compared to those for a structured one. Their key advantage,

however, lies in their high flexibility and adaptability to complex geometries, making them highly versatile. That is because they are made of irregularly shaped elements, namely triangles (in 2D space) or tetrahedra (in 3D space).

A mesh that combines structured regions with unstructured ones is referred to as a hybrid mesh. It is also worth noting that in, three dimensional problems, two new types of elements are introduced, the pyramid and the prism. Hybrid meshes combine the advantages of both structured and unstructured meshes and have seen a great rise in popularity over the past few years.

An example of a typical unstructured and hybrid mesh is shown in figures 1.2 and 1.3 respectively.

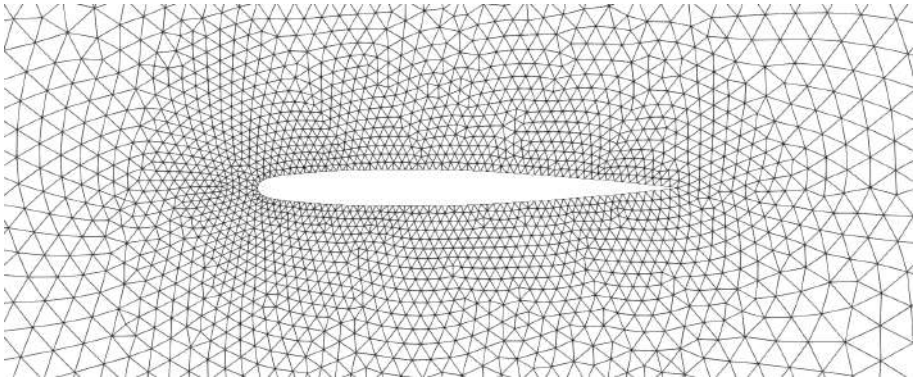


Figure 1.2: *A typical 2D unstructured mesh around an airfoil.*

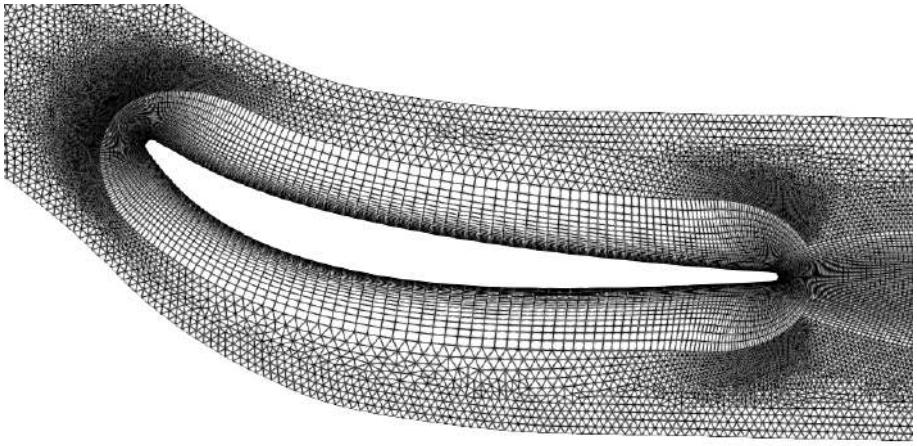


Figure 1.3: *A typical 2D hybrid mesh around a compressor stator blade.*

One extra piece of information that is worth noting is related to the solving software each type of mesh utilizes. In a traditional structured mesh solver, the equations are first transformed, then discretized and solved in a different domain, often referred to as the "computational domain" ([17]), where specific conditions are met. Meanwhile

an unstructured mesh solver discretizes and solves the equations directly in the physical domain. An unstructured mesh solver can also work with a structured mesh, as long as its topological information is in line with the format it uses, making unstructured solvers more generalized. On the other hand an exclusively structured mesh solver is unable to properly utilize an unstructured mesh.

1.2 Mesh Adaptation

Modern engineering problems require large scaled CFD simulations around complex geometries, leading to the use of dense meshes in order to capture all the important aspects of a physical phenomenon. This can create a significant increase in computational demands. Mesh adaptation provides a solution to that problem by adjusting the mesh density where deemed necessary. This way, it aims to optimize and balance computational efficiency with solution accuracy.

Mesh adaptation consists of the refinement and coarsening of a mesh, by adding new or removing old elements in specific areas of the domain, respectively. This procedure takes place as the convergence of the numerical method used to solve the flow equations is progressing and is performed more than a single time. Each adaptation procedure is referred to as a "cycle". The first adaptation cycle begins after the equations have converged to a point where a first, sufficiently accurate, picture of the flow field is obtained. Refinement adds new elements to areas where high solution accuracy is needed and the original mesh is unable to produce. Respectively, coarsening ensures the mesh will not be overloaded with unnecessary elements and keeps the computational and storage demands within reasonable levels, by removing excessive elements. Due to it's nature, mesh refinement is mainly applied to unstructured meshes. Apart from their use in complex geometries, the key advantage of refining an unstructured mesh, is the ability to make changes to it, only in the specified regions of interest. On the other hand, on a structured mesh, a small intervention affects the entire mesh, by adding a large amount of elements across the entire domain, that needlessly increase the computational load of the simulation. The areas in which the mesh needs to be modified are identified with the use of a mathematical approach. In CFD applications, these regions typically correspond to areas where important physical phenomena appear, such as shock waves, boundary layers, vortices, stagnation points e.t.c.

Numerous reasons as to why mesh adaptation is becoming increasingly important can be exposed. The most significant one is that a lot of features of the flow field are unknown until the problem is solved or at least a first view of it is available. This means that it might not be possible to know beforehand the part of the domain where an important phenomenon, such as a shock wave, will be formed. In fact, in practice it is often difficult, even for experienced users, to anticipate exactly what mesh resolution is required in different regions of the domain and therefore

generate an appropriately fine mesh. Hence, as a more clear view of the flow field is obtained, the initial mesh might need to undergo through many evaluations and adjustments, until one that provides the desired accuracy is produced. Mesh adaptation reduces the dependence on initial mesh quality and at the same time removes the tedious task of repeatedly adjusting the mesh, thus minimizing the need for manual intervention. Using mesh adaptation, the CFD simulation can be initiated with a relatively coarse mesh in order to obtain a first view of the flow field, and then, through the adaptation procedure, get a modified mesh that provides more accurate results without introducing an unreasonable computational demand. However, starting with a very coarse mesh can be an ineffective strategy, in some cases. Two main categories of Mesh Refinement methods exist [11]:

1. **The Feature-Based Mesh Refinement**
2. **The Adjoint-Based Mesh Refinement**

Feature-based mesh refinement, as the name implies, focuses on identifying where features of the flow such as shock waves, vortices and stagnation points appear within the domain in order to enhance the mesh. This method involves a mathematical function that takes as input one or more flow variables as well as parameters regarding mesh size. By applying this function, the mesh is evaluated and parts of the domain that require refinement are identified. The selection of flow variables along with the analytical formula of the function are highly dependent on the nature of the problem. Different combinations may produce better results for different types of problems.

Adjoint-based mesh refinement, establishes a mathematical relation between aerodynamic forces, flow variables and the local residual. Using adjoint theory, it detects which regions of the mesh have a greater influence on the local residual error, based on a target value. The target value is incorporated into an objective function and an adjoint optimization problem is then formulated and solved.

More information regarding adjoint based mesh refinement methods, can be found in [9, 11, 20, 13].

One additional point worth mentioning is that adjoint based methods also require an adjoint equation solver, thus increasing the total computational cost of the simulation. On the other hand feature-based mesh refinement is based only on simple evaluation functions and doesn't add any costly numerical method to the overall simulation.

The diploma thesis will focus specifically on the h-refinement of unstructured meshes made of only tetrahedral elements, using feature-based refinement. Further detailed insight regarding the method used will be presented in the following chapters. Further details on adaptation algorithms that consider coarsening, for both 2D and 3D meshes, can be found in [7, 8, 16].

1.3 Thesis Outline

This diploma thesis focuses on the implementation, programming and evaluation of a feature-based mesh adaptation algorithm for the h-refinement of 3D unstructured meshes with only tetrahedral elements. The algorithm is programmed in the C++ programming language and is set to run in line with PUMA [2], the in-house GPU accelerated flow analysis software, developed by the PCOpt/NTUA. The open-source software Paraview [1] is used for the processing and visualization of the results.

The structure of the thesis is outlined as follows:

- **Chapter 2:** A thorough presentation of mesh refinement on tetrahedral element meshes. The fundamental principles of mesh adaptation on tetrahedra are outlined. The types of data used in an adaptation algorithm, along with their storage requirements are discussed. The mathematical expressions for evaluating a mesh in a feature-based adaptation are introduced. The Mesh Adaptation algorithm used is described in detail.
- **Chapter 3:** Application of the mesh refinement algorithm to an inviscid flow of a compressible fluid inside a channel with a bump. Two variants of the problem are considered: one where the bump is symmetrical and one with an asymmetry introduced on its surface in order to induce 3D features to the flow. The adaptation algorithm is employed in both cases and the results are presented and compared.
- **Chapter 4:** Application of the mesh refinement algorithm to an inviscid flow of a compressible fluid around a sphere. This problem is studied for two different freestream Mach numbers, based on a similar study on a transient variant of the problem. The refinement algorithm is employed in both cases and the results are presented and compered to the reference data of that study.
- **Chapter 5:** An overview of the present diploma thesis, followed by conclusions drawn from the studies and suggestions for future work.

Chapter 2

Refinement of Tetrahedral Element Meshes

2.1 Introduction

As mentioned in the Introduction, this diploma thesis focuses on the refinement of 3D meshes with only tetrahedral elements, often referred to as "tetra meshes". In this chapter, the basic principles of mesh adaptation on tetrahedral element meshes will be established. This theory was extracted by expanding on fundamental principles of 2D triangular mesh refinement, as they are comprehensively described in [7, 16, 21]. It also incorporates already established theories specific to tetrahedral element meshes, as outlined in [3, 8, 11].

2.2 Refinement Principles on Tetra Meshes

Mesh Adaptation, in the context of this diploma thesis, involves only the refinement of the mesh which is executed periodically, after a specified amount of iterations of the numerical solution scheme used to solve the flow equations. Refinement entails the subdivision of larger elements into smaller ones in order to increase the mesh resolution. The elements of the initial mesh are often referred to as F1 elements (a term borrowed from biology to denote a first generation). The fundamental concept of mesh refinement, as outlined, for tetra meshes, is illustrated in figure 2.1.

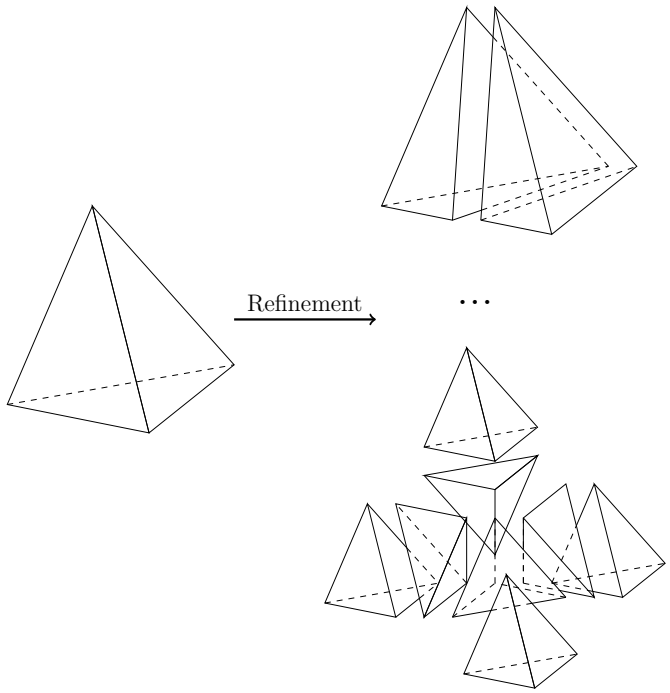


Figure 2.1: *Visual representation of mesh refinement.*

There are several approaches to mesh refinement, each one based on different fundamental principles, resulting in distinct methods. For instance, in this diploma thesis, the primary principle is that no more than three nodes are allowed on the boundary of an active element (an active element is one that is taking part in the solution of the flow equations; this term will further be clarified in the following chapters). This principle, as well as an instance where it is not upheld, is presented in figure 2.2.

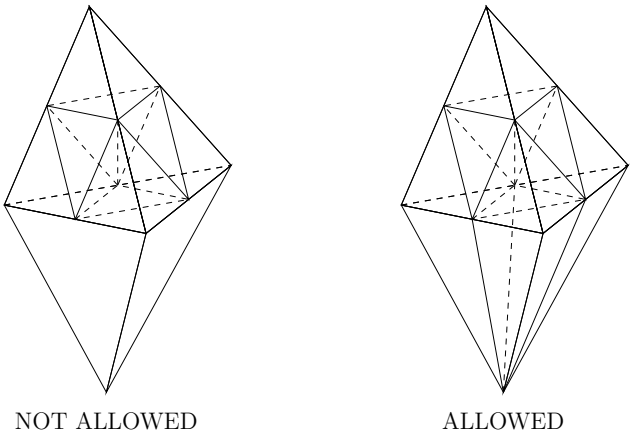


Figure 2.2: *Comparison between a allowed and a non-allowed tetrahedron subdivision. The non-allowed subdivision case corresponds to a different type of adaptation scheme that would also require modifications to the solver software.*

Therefore, it is clear that every mesh refinement algorithm must adhere to certain rules. The set of rules that will be adopted in the present thesis is outlined as follows:

- **Rule 1:** Elements of the original F1 mesh are not allowed to merge in order to formulate new, larger, elements.
  This implies that the initial nodes and edges, which where generated using a mesh generation algorithm cannot be relocated, removed or repurposed into different combinations in order to form different elements with any mesh generation technique. This constraint is illustrated in figure 2.3.

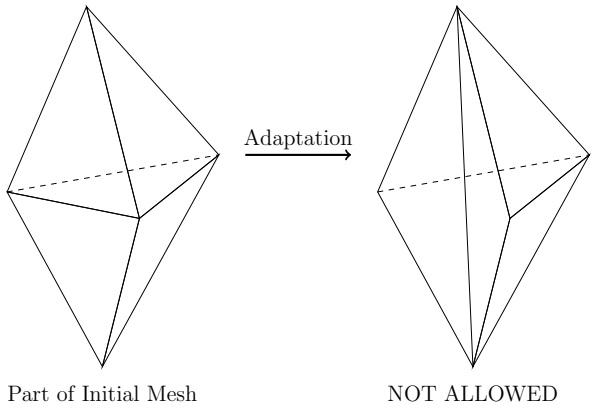


Figure 2.3: *Example of a violation of Rule 1.*

- **Rule 2:** A tetrahedron is allowed to be subdivided in 4 distinct ways.
  The edges of a tetrahedron that are subdivided are determined using mathematical criteria (which will be detailed thoroughly in 2.4). Until subdivision occurs, these edges remain "marked" and these markings define the subdivision patterns of a tetrahedron. A key rule derived from the refinement of 2D triangular element meshes [16, 21] and is expanded to 3D, is that a triangle (a face of a tetrahedron) can have either 1 or all 3 of its edges marked for refinement. Based on this principle, the first 3 unique configurations can be extracted. These subdivision cases, identified by their selected code names, are illustrated in figures 2.4- 2.6.

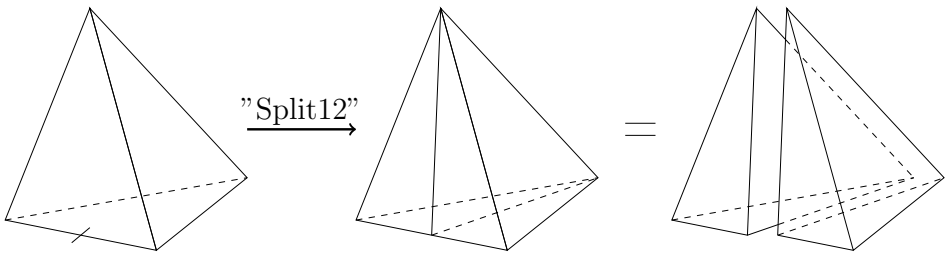


Figure 2.4: *First allowed subdivision of a tetrahedron into 2 tetrahedra, referred to as "Split12".*

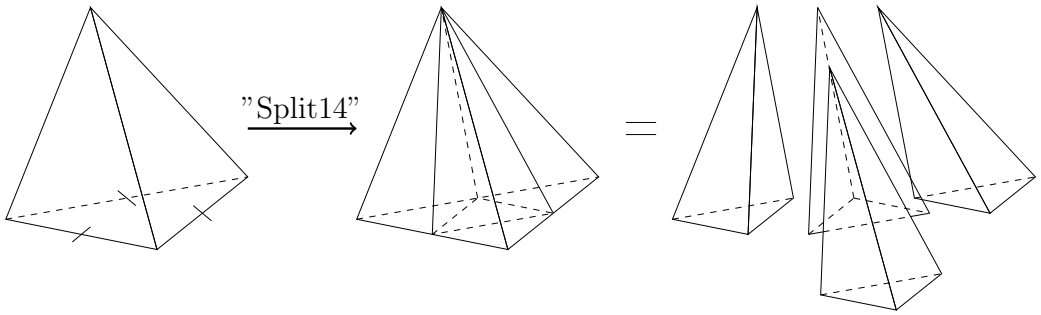


Figure 2.5: *Second allowed subdivision of a tetrahedron into 4 tetrahedra, referred to as "Split14".*

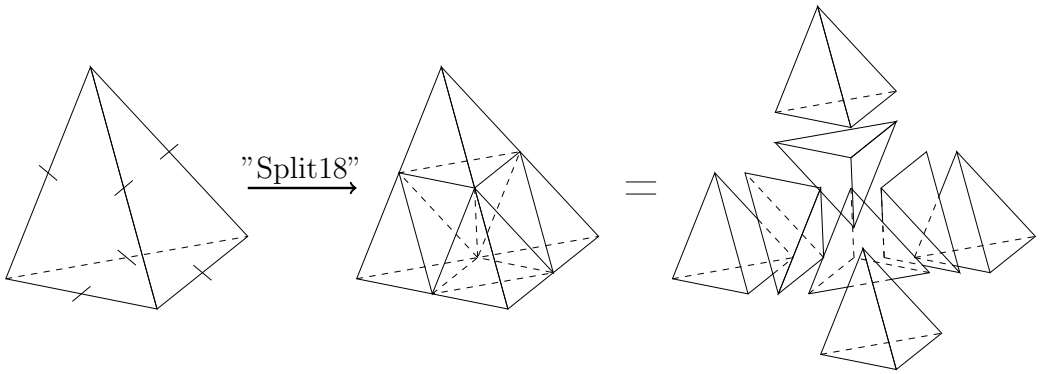


Figure 2.6: *Third allowed subdivision of a tetrahedron into 8 tetrahedra, referred to as "Split18".*

However, due to the 3D nature of the tetrahedron, there is also a case where two edges which are not on the same face are marked for refinement. In such a case, the original tetrahedron must be subdivided using the "Split12" subdivision type (shown in figure 2.4) twice, resulting in the fourth subdivision form, shown in figure 2.7.

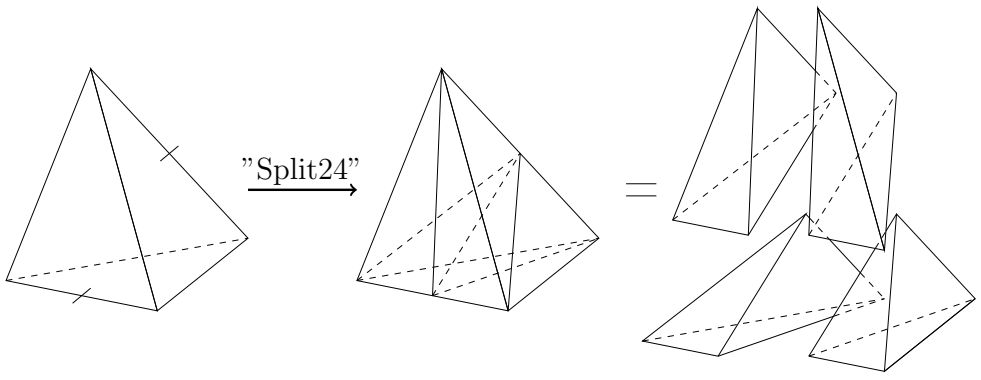


Figure 2.7: *Fourth allowed subdivision of a tetrahedron into 4 tetrahedra, referred to as "Split24".*

Finally, a non-allowed subdivision of a tetrahedron is presented:

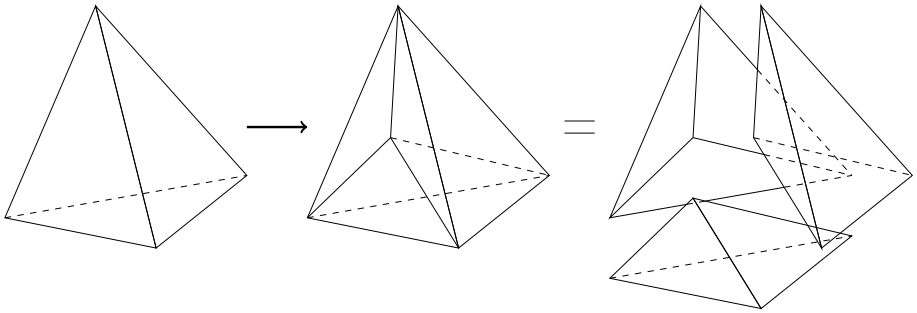


Figure 2.8: *Example of a non-allowed tetrahedron subdivision that violates Rule 2.*

- **Rule 3:** A tetrahedron that originates from the subdivision of a previous generation element into two or four is not allowed to undergo further subdivision. Instead, the parent tetrahedron must be re-activated and then split, according the modes showcased in figures 2.4 - 2.7 , before it can be subdivided again. This procedure, applied to various different cases, is illustrated in figure 2.9.

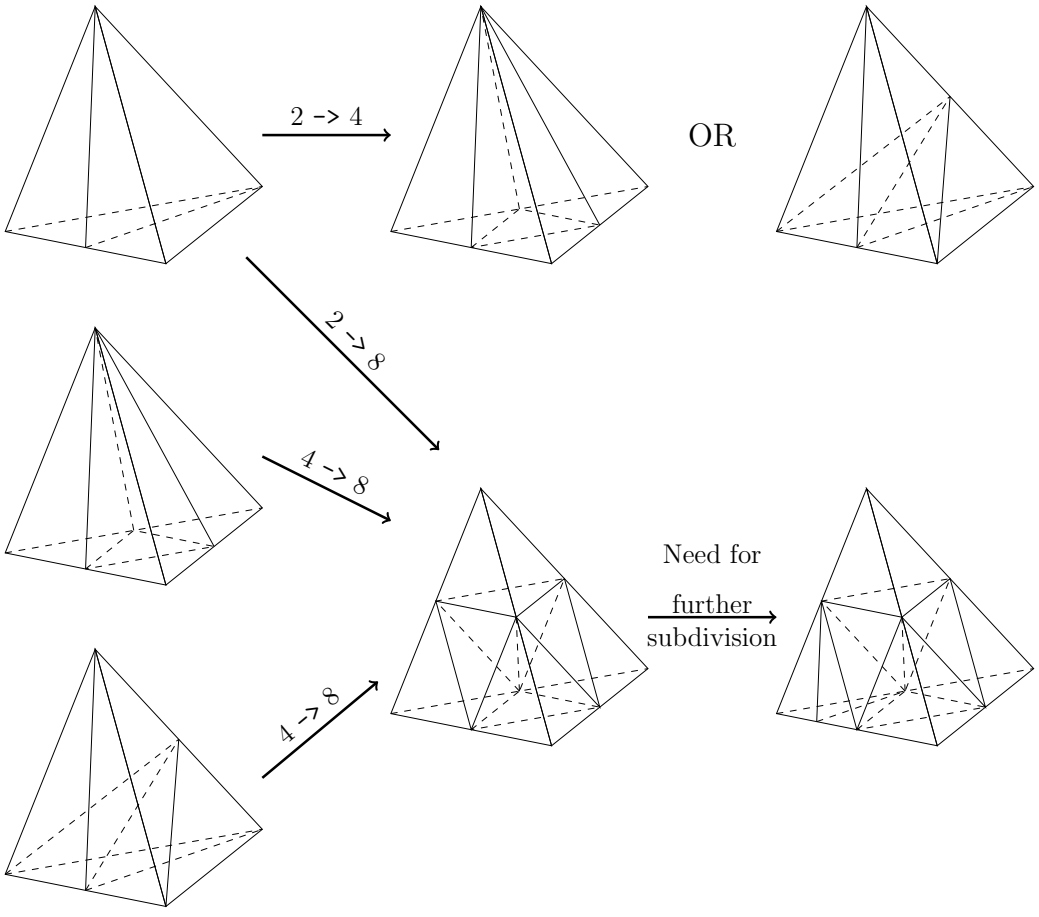


Figure 2.9: *A visual illustration of Rule 3.*

This restriction is imposed in order to prevent the creation of elements of very high aspect ratio due to a continuous "halving" of elements, which can lead to numerical errors. The generation of these "undesirable" elements is shown in figure 2.10.

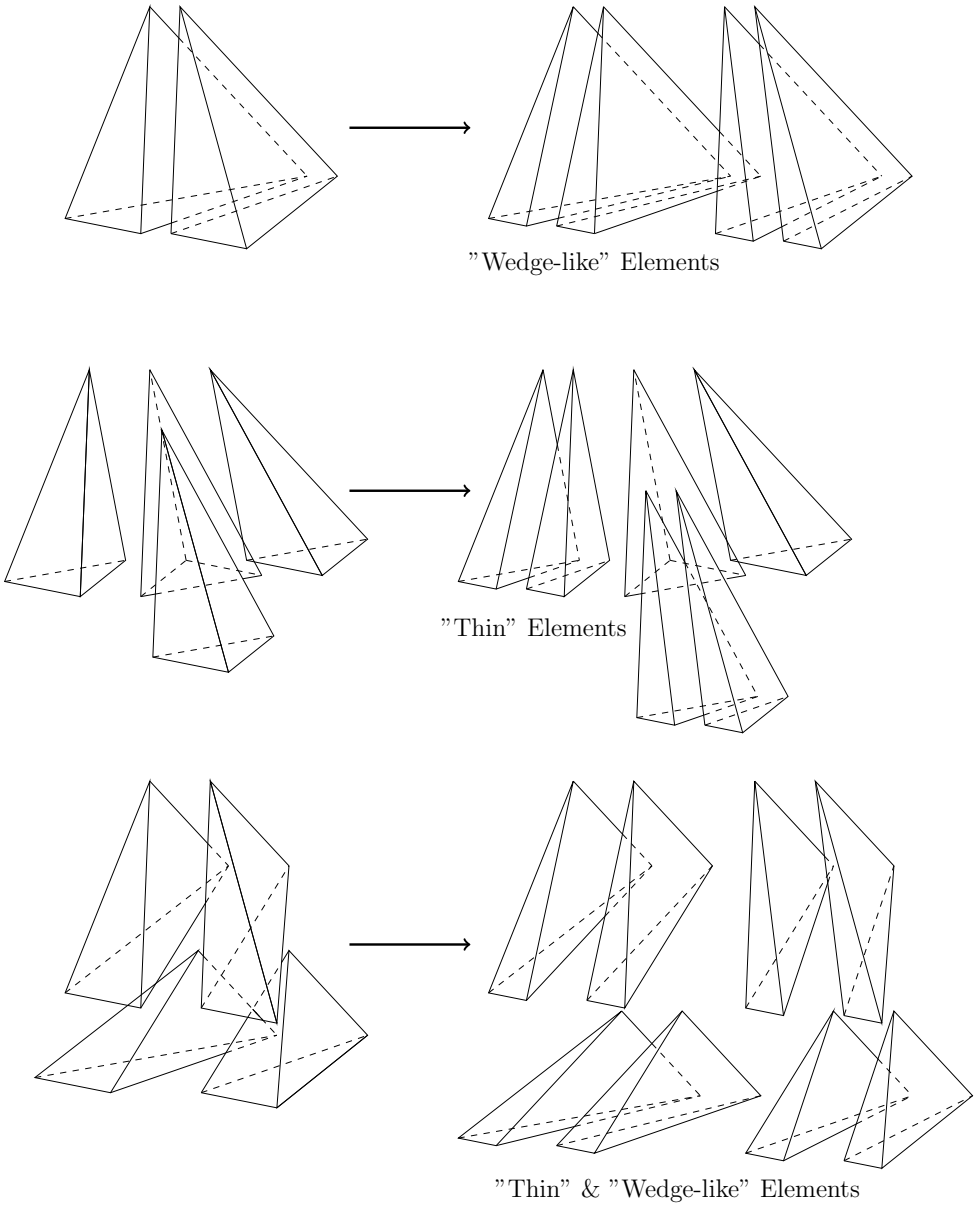


Figure 2.10: *Elements generated from violation of Rule 3.*

A detailed analysis of tetrahedral element quality metrics can be found in [10].

- **Rule 4:** A tetrahedron originating from the subdivision of a previous generation element into 8 tetrahedra ("Split18") is allowed to further be subdivided, without any of the restrictions imposed by Rule 3.

- **Rule 5:** Tetrahedra are not allowed to divide endlessly.
  This constraint is imposed in order to prevent the creation of very small elements which can cause convergence issues and unnecessary memory overload. For that reason, a minimum allowed edge length is defined by the user.

- **Rule 6:** When a tetrahedron is subdivided, it may also trigger the subdivision of its neighboring elements, in order to preserve the topology of a typical unstructured mesh. An example is shown in figure 2.11.

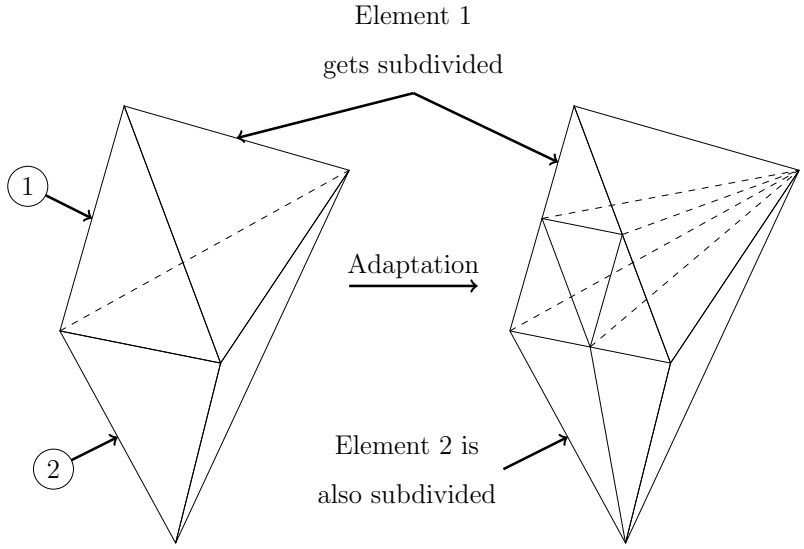


Figure 2.11: *Example of the application of Rule 6.*

- **Rule 7:** In periodic meshes (commonly found in internal aerodynamics problems, such as the analysis of a compressor stage), any modifications made to one periodic boundary must be replicated on its corresponding pair. This is because periodic boundaries, by definition, imply that with appropriate translation and/ or rotation, the periodic boundaries will eventually coincide. A simplified example is shown in figure 2.12.

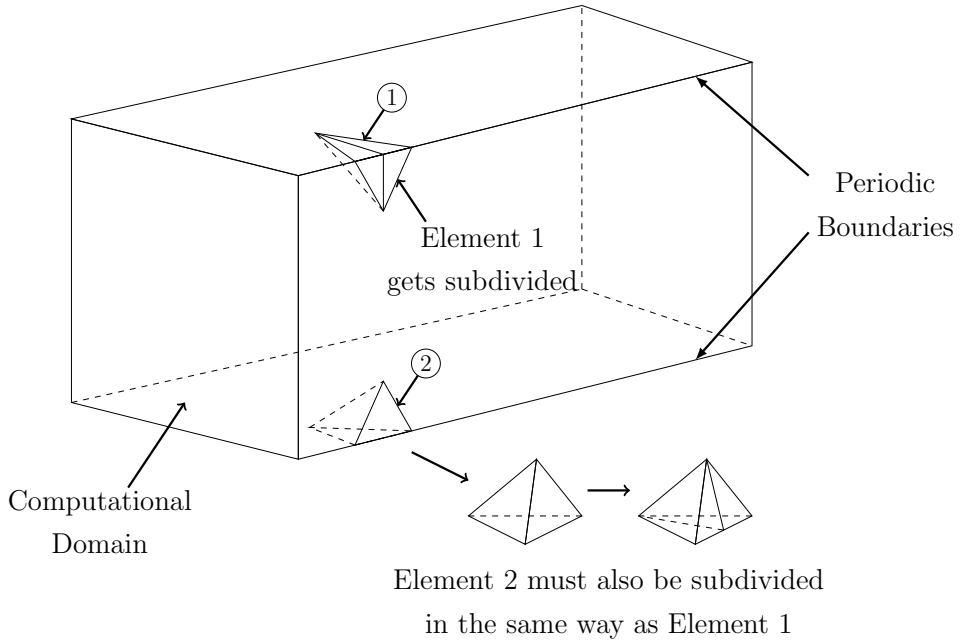


Figure 2.12: *Example of the application of Rule 7 on a simplified computational periodic domain.*

2.3 Refinement Data Structure

The selection of an appropriate data structure for a mesh refinement algorithm is a complex decision. This arises not only due to considerations related to the implementation of the algorithm itself but also from resource constraints. Therefore, a very important task prior to the development of the algorithm is the selection of an efficient data structure capable of both storing and providing easy access to all the necessary information.

Due to the nature of unstructured meshes, their processing and their refinement, is inherently complex. An unstructured mesh relies on a complex topological data structure that contains the various relations between its elements in order to function properly. With the introduction of adaptation, additional information regarding each adaptation cycle, as well as new interrelations between elements is also necessary.

In the context of mesh refinement, the elements of the mesh can be classified into one of the following three categories:

- **Active Elements:** are the elements of the latest generation and are taken into account during the current iteration of the numerical scheme used to solve the flow equations. These can either be elements of the initial mesh (F1 mesh) or descendants generated in any refinement cycle. These elements may be

marked for refinement during the current refinement cycle.

- **Inactive Elements:** are the elements that are not taken into account during the current iteration of the numerical scheme used to solve the flow equations. These elements have been replaced by newer ones due to their subdivision during an adaptation cycle. Their data must remained stored, as they may become active again in a future refinement cycle, particularly when Rule 3 is applied.

- **Neutralized Elements:** are elements that are marked for deletion in order to ensure compliance with Rule 3. These elements are no longer necessary and their data must be discarded. However, their identification numbers must be stored in a dedicated list for future use by newly generated elements. An example is illustrated in figure 2.13.

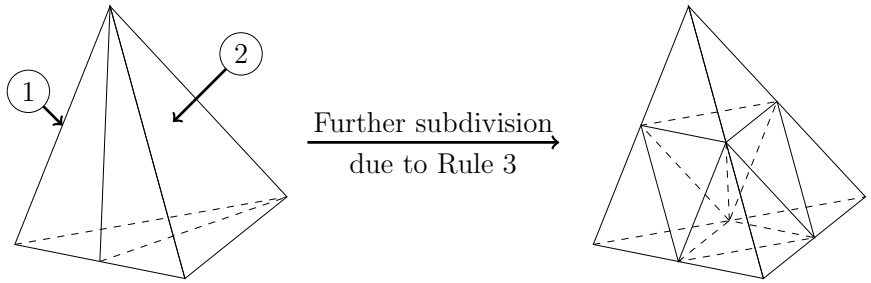


Figure 2.13: *Neutralization of elements 1 and 2 due to parent element refinement.*

There are also two important terms that characterize an element in relation to its adaptation history. The term ancestor (or parent) corresponds to an element that has been subdivided into smaller elements during the refinement process. These smaller elements are referred to as descendants (or children) of the original ancestor element. Therefore, an inactive element is typically an ancestor of two or more active elements while an active element can either be a descendant or a F1 mesh element.

The same classifications described above, with identical features, also apply to edges.

In order to formulate a mesh refinement algorithm, two main types of data need to be stored:

- **Topological Data:** Contains all the necessary information in order to describe an unstructured mesh. It stores information such as vertex and element connectivity, node coordinates and boundary conditions. A topological data structure is required regardless of whether a mesh refinement scheme is employed or not, as it is essential for the numerical solution of the flow equations.

  For a tetra mesh refinement algorithm, an edge-based topological data structure is proposed (similar to the one described in [3]). This type of edge data

structure facilitates efficient refinement and contains the right amount of information while also maintaining reasonable memory requirements. This structure is organized as follows:

Each vertex is associated with a unique identification number (node ID) and its X, Y and Z coordinates are stored, as well as a flag that indicates whether it belongs to a boundary of the domain or not, along with its corresponding boundary condition. Each edge is described by a unique identification number (edge ID) as well as the two nodes that define it.

Lastly, regarding the elements, each tetrahedron is made up of 4 vertices, 6 edges and 4 faces. In this data structure, each tetrahedron is represented by its unique identification number (element ID) and its 6 edges, rather than its 4 vertices. To ensure consistency within this proposed type of element data structure, an agreement on the local numbering of each element's nodes, edges and faces must be adopted. The numbering scheme that was used aligns with the internal element indexing of the PUMA CFD solver, developed by the PCOpt Unit of NTUA. The local numbering on a single tetrahedron is showcased in figure 2.14.

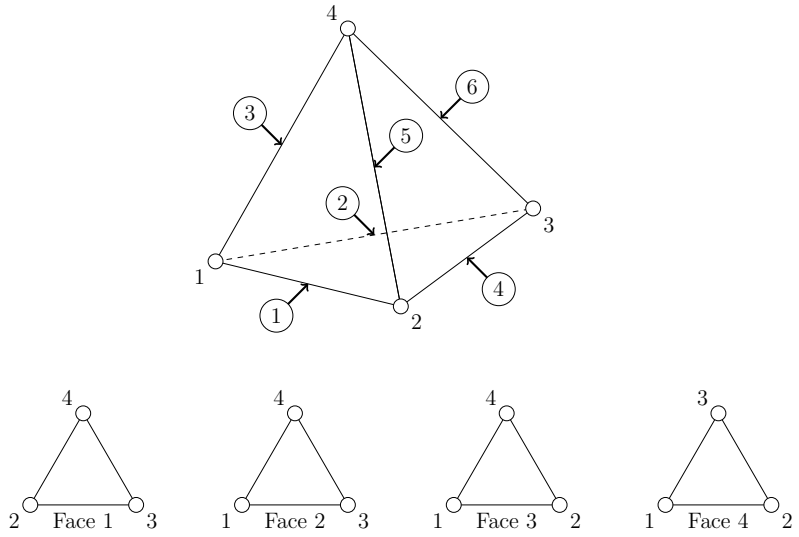


Figure 2.14: *Local indexing of a tetrahedron's vertices, edges and faces.*

- **Genealogical Data:** Contains all the essential information in order to perform a mesh adaptation on a given mesh. It contains generational relations between mesh edges and elements (such as ancestors and descendants). In the absence of mesh refinement, this information becomes unnecessary and the solution of the flow equations relies solely on the topological data structure of the mesh, as previously. For a tetra mesh refinement algorithm, a genealogical data structure based on mesh edges and elements is proposed. Specifically the

data structure described in [16], for triangular meshes is expanded for tetra meshes and is organized as follows:

For each edge, 4 integers are required to fully describe its genealogical information. The first 2 integers denote the identification numbers of the 2 descendants originating from the current edge. This means that if the edge is active, these two numbers are equal to zero.

The 3rd integer represents the ID of the ancestor of the edge. If it is equal to zero then this edge is either a F1 mesh edge or an internal edge generated during the refinement stage.

The last integer, corresponds to the ID of the mid-node generated on that edge. The term mid-node refers to a new node generated between an existing edge's nodes during its subdivision, at the refinement stage. The prefix "mid" is used because it is placed directly at the middle of the edge (unless the edge belongs to a boundary). If that last integer is equal to zero, then the edge is active. This four-integer, per edge, genealogical data structure is illustrated in the form of an array, in figure 2.15.

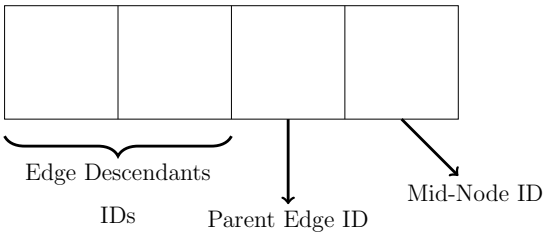


Figure 2.15: *Array representation of a single edge's genealogical data.*

For each element, its genealogical information is described using 10 integers. The first integer represents the number of descendants this elements has. It can either be equal to 2, 4 or 8, which indicates that the element is inactive. If it is equal to zero then the element is active, while a negative value indicates that this is a neutralized element and its data should be discarded. The next 8 integers are used to store the IDs of the descendant elements that originate from that element. If the element is active, these integers are all equal to zero (along with the first). The last integer is corresponds to the ID of the parent element that was subdivided to generate the current element. If this number is zero, then the element is part of the F1 mesh. This 10-integer, per element, genealogical data structure is showcased in the form of an array, in figure 2.16.

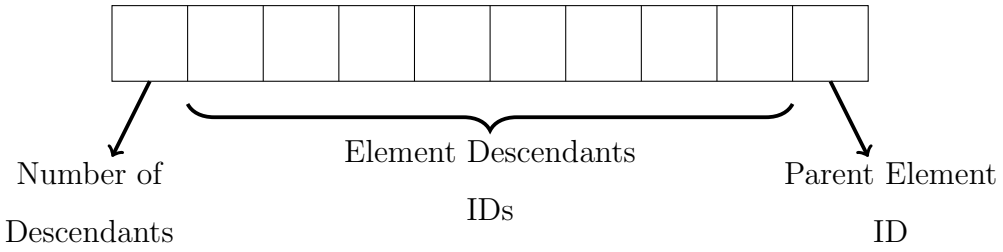


Figure 2.16: *Array representation of a single element's genealogical data.*

It is important to highlight some modifications applied to the element genealogical data structure as part of the programmed software. Since only refinement is allowed, subdivided elements cannot get reactivated due to coarsening (only due to application of Rule 3, during refinement). Therefore, when an element is subdivided into 8, as dictated in Rule 4, its descendants can never become active again. Consequently, it can be immediately neutralized and its descendants are handled as if they were F1 mesh elements (pseudo-F1 mesh elements). This modification aims to reduce memory requirements of the genealogical data structure. Using this approach, the number of integers stored per element is reduced from 10 to 6, since only inactive elements with up to 4 children will need to be stored. This updated 6-integer, per element, genealogical data structure is showcased in the form of an array, in figure 2.17.

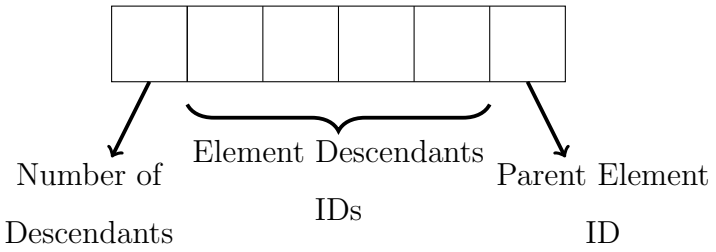


Figure 2.17: *Array representation of a single element's genealogical data when only refinement is considered.*

This, for example, in a programming language such as C++ where an integer occupies 4 bytes of memory, the total memory requirement for each element is reduced from 4 bytes x 10 integers (40 bytes) to 4 bytes x 6 integers (24 bytes), resulting in a 40% decrease in memory usage per element.

However, if de-refinement is also considered, this approach is no longer feasible due to the nature of the algorithm itself, unless an alternative data structure is introduced. A slightly different approach on this is discussed in [8], which could serve as a foundation for further future development.

2.4 Refinement Criteria

The main objective of a mesh refinement method is to obtain the most accurate approximation of the exact numerical solution of the flow equations (or any system of partial differential equations, within the broader scope of computational mechanics) while minimizing the number of mesh elements. To ensure a successful refinement, it is crucial to accurately and consistently identify the areas here the mesh needs to be refined. Such areas are identified using edge-based criteria, which, in finite volume schemes, typically assume a vertex-centered approach. There are two basic discretization schemes in finite volume methods; the cell-centered and the vertex-centered. In a cell-centered approach, the finite volumes coincide with the mesh elements, while in a vertex centered approach the finite volumes are formed around each node of the mesh. In both approaches the flow variables are computed at the center of their respective finite volumes. It is worth noting, however, that with appropriate modifications, the same principles can be applied to a cell-centered finite volume scheme, as described in [12]. These criteria are based on features of the flow (such as pressure, velocity etc.) and are applied to every active edge of the mesh in order to determine whether it requires refinement or not. To ensure that the refinement algorithm accurately marks the mesh edges that need to be refined, three key parameters must be considered:

1. The selection of a flow feature, which is referred to as a "Sensor", capable of providing sufficient information regarding the areas of the domain where significant physical phenomena occur. For instance, in a shock wave dominated problem, the Mach number could serve as a sensor, indicating regions where shock waves form.

2. The use of an appropriate analytical expression, which is referred to as an "Decision Function". This function takes the preselected sensor as input and is applied to every active edge of the mesh. Its purpose is to provide a numerical value that indicates the sensitivity of each edge. It typically relies on the rate of change of the sensor along each edge, therefore a common approach is to base it on the gradient of the sensor or some form of differential operator.

3. The selection of an appropriate threshold for the indexes provided by the decision function. This threshold, also referred to as "Cutoff Value", is used to determine whether an edge should be marked for refinement or not. The Cutoff Value must be carefully chosen in order to avoid repeatedly marking the same edges (constantly forcing them to activate and re-activate), which could lead to performance issues and inaccuracies.

Decision functions and Cutoff Values will be further discussed in section 2.5.

It is worth noting that multiple sensors and decision functions can be employed simultaneously. This approach is typically used in order to detect different flow

phenomena occurring in different regions of the computational domain.

2.5 Decision functions & Cutoff Values

As previously mentioned, selecting the appropriate sensor and decision function is crucial for the success of the refinement process. This chapter will examine the various types of decision functions, highlighting their advantages and limitations. Additionally, methods for determining their threshold value for refinement will be discussed. In this context, sensors are denoted by the letter Φ, while decision functions are represented by the letter f.

A decision function is an analytical expression that takes the selected sensor as input and is applied to every active edge of the mesh. It provides a numerical index that is used to determine whether an edge should be subdivided or not. Decision functions are primarily categorized based on whether they use the magnitude of the sensor's change along the edge, or its relative change with respect to the edge length, similar to a derivative. However, other types of decision functions, focusing on the quality metrics of tetrahedra or the estimated residual error of the flow equations, are also discussed in the literature, such as in [8]. To illustrate the concept of decision functions, a 1D case will be presented, as shown in figure 2.18.

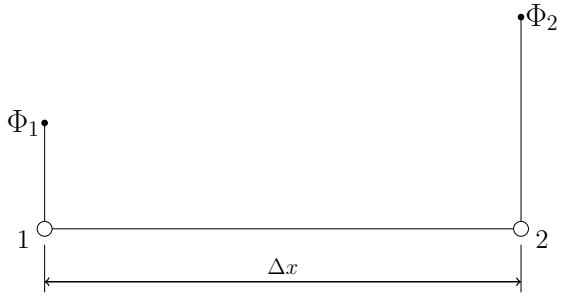


Figure 2.18: *Example of a 1D adaptation problem on an edge with nodes 1 and 2.*

The simplest decision function expression, that is based on the change of the sensor along the edge, is:

$$f = |\Phi_1 - \Phi_2| \tag{2.1}$$

Alternatively, a simple expression that uses sensor's rate of change, relative to the edge length, is:

$$f = \left|\frac{d\Phi}{dx}\right| \sim \left|\frac{\Phi_1 - \Phi_2}{\Delta x}\right| \tag{2.2}$$

Other similar expressions can be found in the literature [14, 16], such as the following:

$$f = 1 + \left| \frac{d\Phi}{dx} \right| \tag{2.3}$$

$$f = \sqrt{1 + \left| \frac{d\Phi}{dx} \right|^2} \tag{2.4}$$

$$f = \left| \frac{d^2\Phi}{dx^2} \right| \tag{2.5}$$

Each of the previously presented decision functions (Equations 2.1- 2.5), yields different results and is suited for various types of problems. In some cases, a simpler decision function, such as the one in Equation 2.1, may produce a better refined mesh than one that might seem more mathematically sophisticated, like the ones in Equation 2.3 or 2.4. If the selected sensor quantity has a linear distribution along an edge, a derivative-based decision function will produce the same values, regardless of the number of times the edge has been subdivided. However, if the sensor does not follow a linear distribution, a decision function based on its gradient will produce varying values as the edge is subdivided. This could result in continuous subdivision of the edge, or the suspension after the first one. An example of a continuous subdivision is shown in figure 2.19.

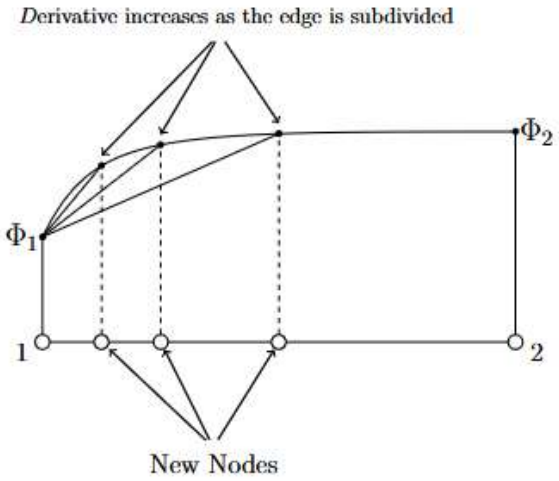


Figure 2.19: *Example of a non-linear sensor distribution along the edge 1-2, using a gradient-based decision function.*

However, the behaviors described may be desirable, depending on the specific problem. It also possible for a decision function to be a combination of various decision

functions, like the ones previously shown in Equations 2.1- 2.5, for example:

$$f = \left|\frac{d\Phi}{dx}\right| + \beta \cdot \left|\frac{d^2\Phi}{dx^2}\right| \tag{2.6}$$

Equation 2.6 is a combination of equations 2.2 and 2.5, with β being a constant predefined by the user. It is important to note that, in practice, it is common to normalize decision functions. For example:

$$f = \left|\frac{\Phi_1 - \Phi_2}{\Phi_1 + \Phi_2}\right| \tag{2.7}$$

Normalizing helps to detect phenomena where the changes in the sensor are very minor and at the same time, a higher mesh resolution is required. It also provides a common scale for determining the threshold values, which will be discussed further in the following section regarding these cutoff values. However, when normalizing in this manner, it is possible for the denominator of an expression to become zero. In such cases, a small quantity, denoted as ϵ, is added to prevent division by zero.

$$f = \frac{|\Phi_1 - \Phi_2|}{|\Phi_1 + \Phi_2| + \epsilon} \tag{2.8}$$

If multiple sensors are employed, a suggested decision function proposed in [11], is the following:

$$f = \nabla q = (q_1 - q_2)(d_{12})^p \tag{2.9}$$

where subscripts 1 and 2 refer to the edge nodes, d_{12} is the edge length and the index p is the distance influence factor. q is the weighted and normalized sum of the various Sensors used. In a generalized notation, for node i, q is defined as:

$$q_i = \sum_{j=1}^{N_{sens}} w_j \cdot \left|\frac{\Phi_i^j}{\Phi_{REF}^j}\right| \tag{2.10}$$

where index i denotes the number of the edge node, the subscript j denotes the Sensor, w_j is the weight of the sensor j. In case only one sensor is selected, its weight is set to 1 and all the others are equal to zero. N_{sens} is the number of sensors used, Φ_{REF}^j is a reference value, used to normalize the sensor j. It can be a characteristic value of the flow field. For example, in the analysis of a wing, if sensor j represents the velocity, the reference value would be equal to the far-field velocity, V_∞.

Lastly, in certain problems, the user may want to promote the splitting of longer or shorter edges. In such cases, a weight function, $g(l)$, is employed. This function takes the edge length, l, as input and is multiplied with the decision function. The new, weighted, value of the decision function, f^*, is then given by:

$$f^* = f \cdot g(l) \tag{2.11}$$

After the appropriate sensor and decision function are selected, the next step is to define the corresponding threshold, commonly referred to as cutoff value. The refinement cutoff value is denoted as f_1. Edges where the decision function exceeds f_1 are marked for refinement. It can easily be extracted that a small value of f_1 will result in a large number of edges being marked for refinement, potentially leading to an excessive number of elements, causing memory overload. A common approach for determining the refinement threshold is to base it on statistical quantities of the decision function. For example:

$$f_1 = \alpha \cdot f_{mean} + \beta \cdot f_{dev} \tag{2.12}$$

where f_{mean} is the mean value of the decision function on all the active edges it was calculated, f_{dev} is the standard deviation. α and β are multiplication factors that range between 0 to 1 and 0 to 3, respectively.

2.6 Refinement Algorithm

The refinement process begins by scanning each and every active edge of the mesh and calculating the value of the decision function on it. Based on this value, the edge is flagged with an index that indicates whether it should be refined or coarsened. This classification of the edges relies on the two cutoff values discussed in section 2.5. During this marking phase, Rule 5 is applied to prevent subdivision into excessively small elements. Edges shorter than a specified threshold are not allowed to be marked for refinement. This threshold is generally defined as a percentage of the minimum edge length of the initial F1 mesh, typically ranging between 10% and 20%.

After the marking of the edges is complete, the refinement process is initiated. The refinement algorithm employed in this work consists of the following four steps:

- **Step 1:** Upon completing the initial marking of edges, certain elements of the mesh may form patterns of marked edges that do not comply with one of the four subdivision cases of a tetrahedron, as established by Rule 2. In such cases, additional edges must be marked, in order to form the allowed patterns. For instance, if only two edges of an element are marked for refinement, two

possible cases arise: If they are not on the same face, this leads to the "Split24" subdivision (as shown in figure 2.7). If they are on the same face, the third unmarked edge on that face must also get marked to lead to the "Split14" subdivision (shown in figure 2.5). This distinction is illustrated in figure 2.20.

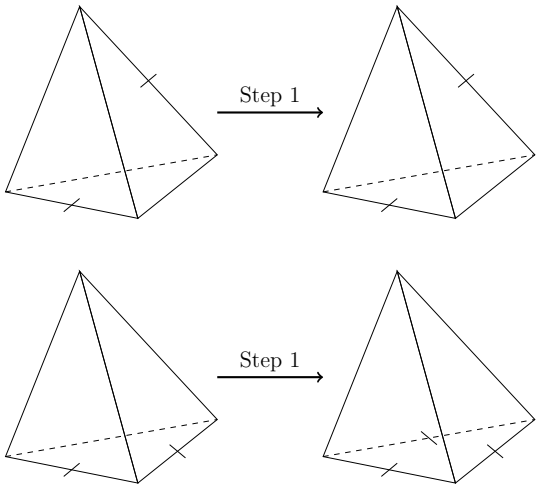


Figure 2.20: *Example of Step 1 of the refinement algorithm for a tetrahedron with 2 marked edges.*

Similarly, if three edges of a tetrahedron are marked, a check of whether they belong on the same face or not must be performed. If they are on the same face, no additional edges are marked (as it already complies with the "Split14" subdivision). If they are not, then all the edges of the element must be marked in order to comply with the "Split18" subdivision. This is demonstrated in figure 2.21.

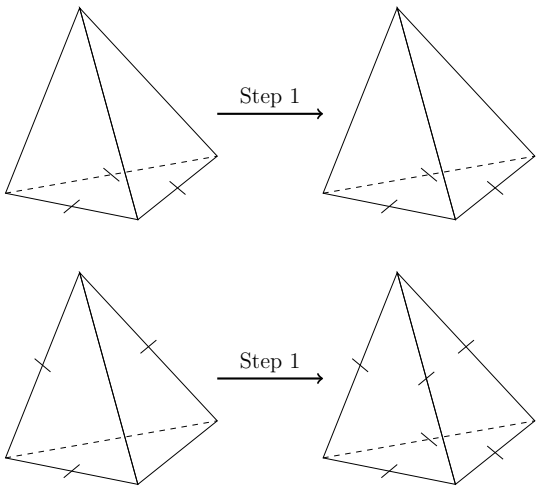


Figure 2.21: *Example of Step 1 of the refinement algorithm for a tetrahedron with 3 marked edges.*

Lastly, if four or five edges of an element are marked, all the remaining, unmarked, edges must also be marked, in order to lead to the "Split18" subdivision, as shown in figure 2.22.

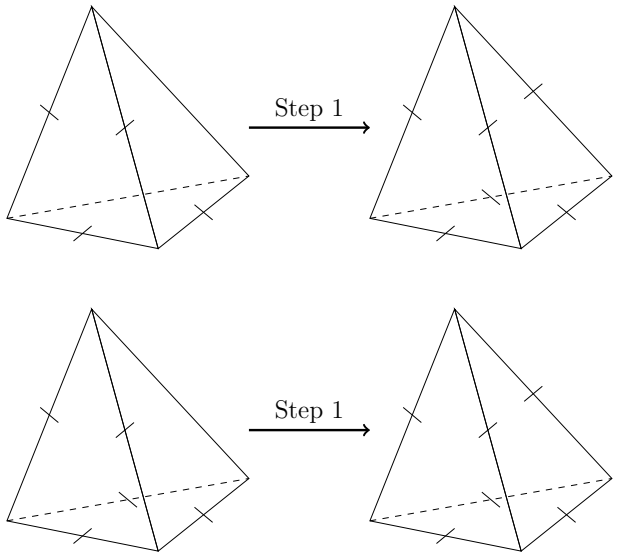


Figure 2.22: *Example of Step 1 of the refinement algorithm for tetrahedra with 4 or 5 marked edges.*

- **Step 2:** According to Rule 3, the offspring elements of a tetrahedron that has been subdivided into two or four (as a result of one of the "Split12", "Split14" or "Split24" subdivisions) are not allowed to be further subdivided. If such an element is detected with at least one marked edge, its parent element is reactivated and, then, its edges are marked according to the allowed patterns, as described in Step 1. This process is carried out in three stages:

  i) First, the offspring elements are neutralized (as described in section 2.3) and their IDs are added to a list for future use by new elements. However, their mid-nodes are not deleted, as they are required in order to form the new elements.

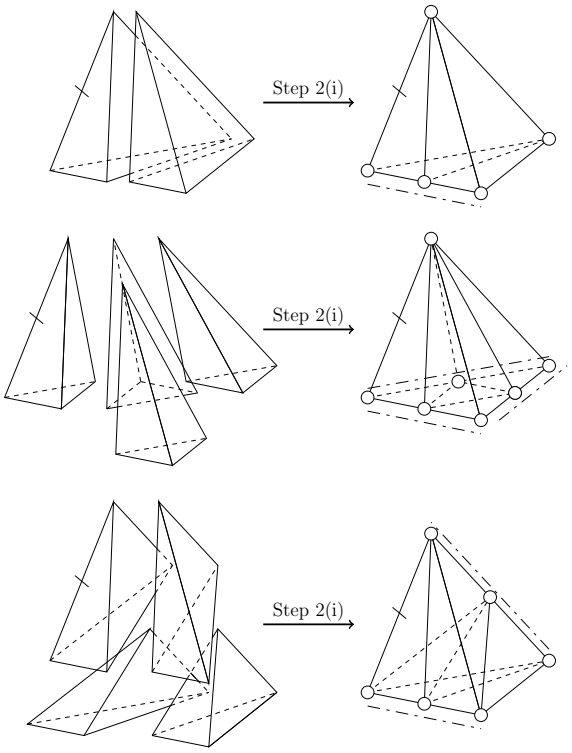


Figure 2.23: *Example of Step 2(i) of the refinement algorithm: Neutralization of descendant elements and re-activation of the parent element.*

ii) Next, all unnecessary edges are removed. These are the edges that split the faces of the parent element into two or four. The latter type is always preserved, figure 2.24, while the former are deleted.

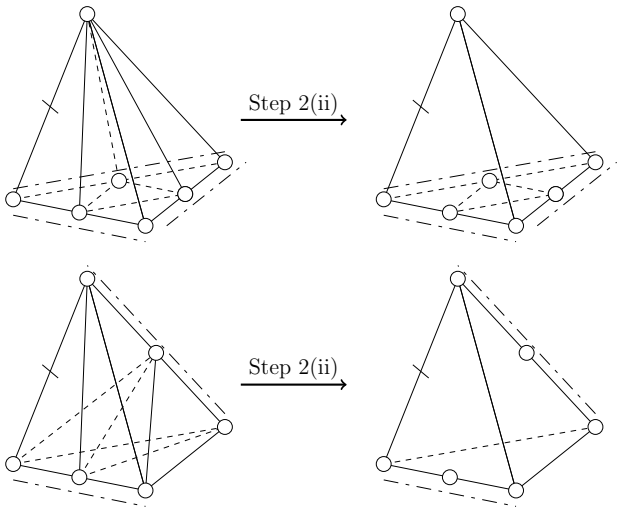


Figure 2.24: *Example of Step 2(i) of the refinement algorithm: Deletion of unnecessary edges.*

However, if an element that was previously subdivided according to "Split12" is reactivated, some of these types of edges may still be needed to form the new elements. This occurs when the parent element is heading towards a "Split14" of "Split24" subdivision. In such cases, one or both of these edges remain intact and are not deleted, as shown in figure 2.25.

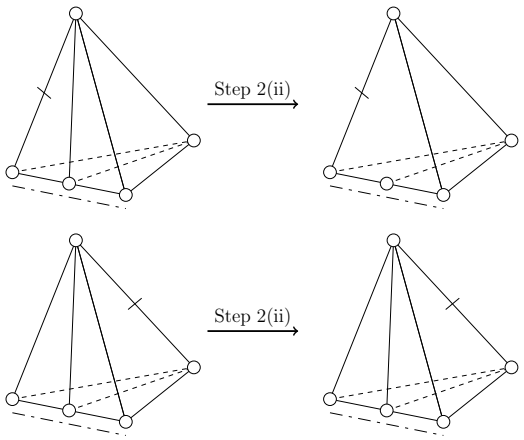


Figure 2.25: *Example of of Step 2(ii) of the refinement algorithm where some edges are not deleted.*

iii) Lastly, the inactive edges of the parent element are marked. Additional edges may be marked using the method described in Step 1 to ensure that one of the four allowed subdivision patterns of a tetrahedron is formed.

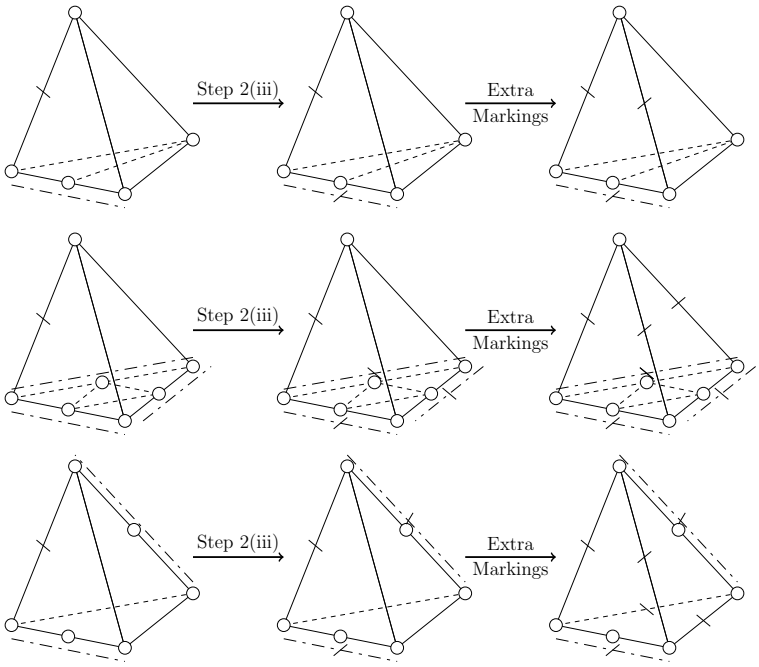


Figure 2.26: *Example of Step 2(iii) of the refinement algorithm: Extra markings of element edges in order to form one of the four allowed subdivision patterns.*

It is clear that Steps 1 and 2 of the refinement algorithm alter the state of marked edges within the mesh. Therefore, these two steps must be executed iteratively within a nested loop until no additional edges get marked to ensure that only the allowed subdivision patterns exist. This is done because, as made clear in Step 1, a marked edge can trigger the marking of other edges not marked during the initial marking stage, using the decision function. This nested loop logic is described in the flow chart of figure 2.27.

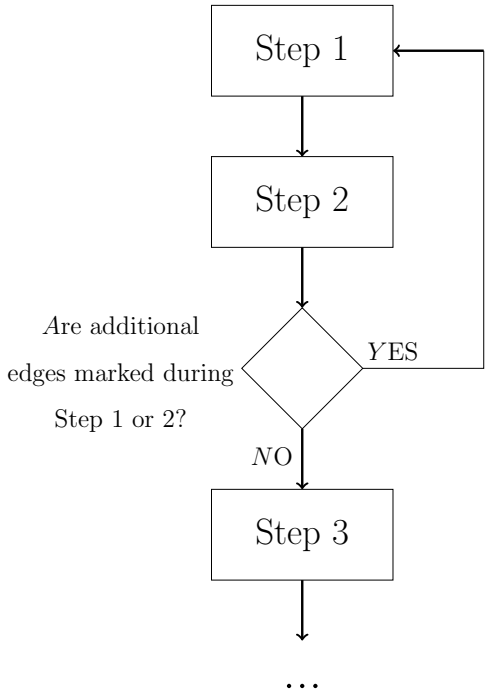


Figure 2.27: *Flow chart of the continuous repetitions of Steps 1 & 2 of the refinement algorithm.*

- **Step 3:** After marking for refinement of the mesh edges is complete, the marked edges are subdivided. Each marked edge is split into two and a new node is generated, as shown in figure 2.28.

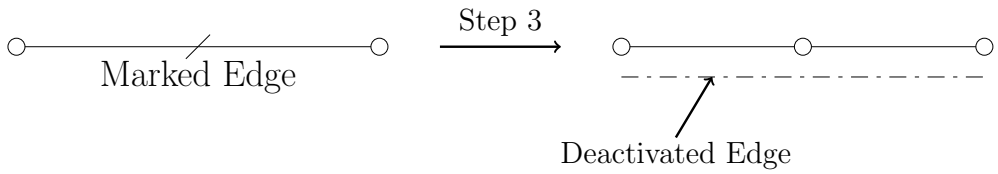


Figure 2.28: *Example of Step 3 of the refinement algorithm: Subdivision of marked edges & generation of new nodes.*

The new node is positioned at the midpoint of the edge, unless it belongs to a boundary of the domain. In this case its coordinates are determined through

interpolation in order to preserve the original domain geometry and maintain consistency with the exact solution of the specific problem. For instance, in the case of a wing, if a new node is added along the surface, its placement must comply with its corresponding geometry‘[13]. If this is not ensured, it will distort the original geometry of the wing, potentially compromising the accuracy of the computed aerodynamic results, including lift and drag. An example, on a 2D problem, is showcased in figure 2.29.

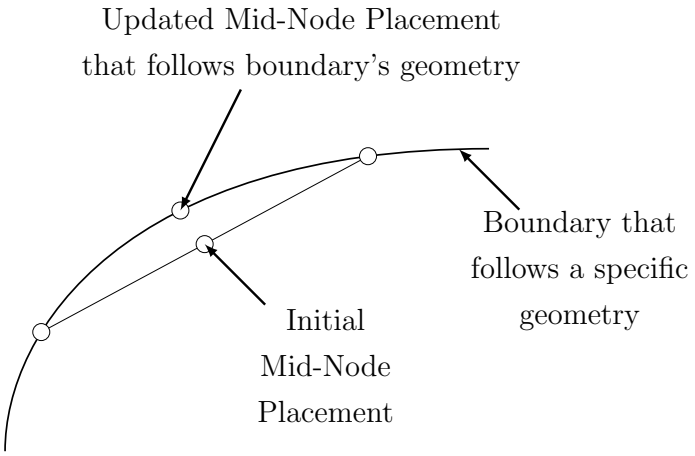


Figure 2.29: *Correction of a boundary node's coordinates in order to comply with the boundary's real geometry.*

After this is complete, the flow variables are computed on the new nodes via numerical interpolation in order to be used when the solution of the flow equations will progress after the refinement is complete. An example of a simple, linear, interpolation in a 1D problem is illustrated in figure 2.30.

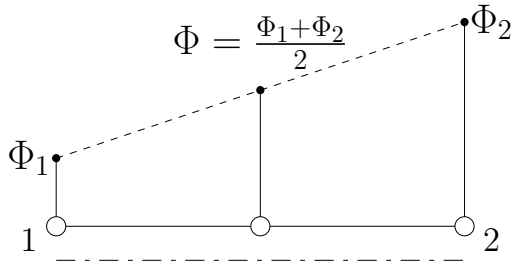


Figure 2.30: *Linear interpolation of the flow quantity "Φ" at a newly generated mid-node of an edge.*

- **Step 4:** The final step of the refinement process is the creation of the new

elements to replace their ancestors. Their topological and genealogical information is determined and, at the same time, their parent elements are deactivated and their genealogical data is updated. This element generation process is illustrated in figure 2.31.

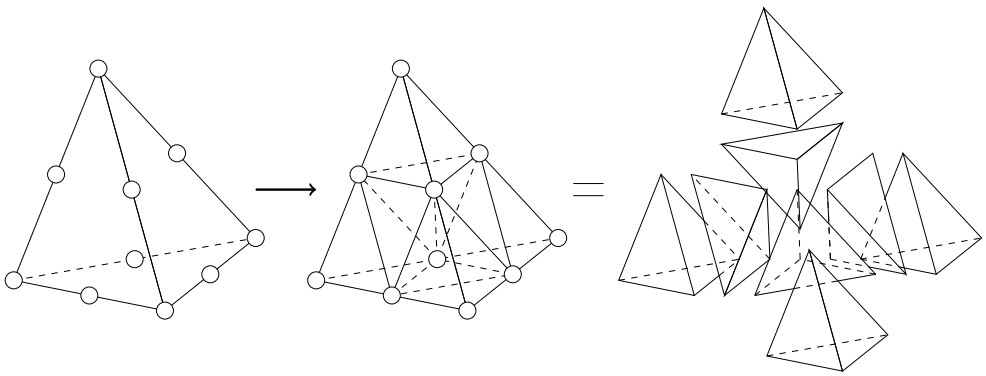


Figure 2.31: *Example of Step 4 of the refinement algorithm: Generation of new elements.*

At this stage, the solution of the flow equations resumes with the new, adapted mesh. The solution restarts using the same flow field the adaptation was initiated with, including the interpolated flow variables at the newly generated nodes, as described in Step 3. Once the solver iterations reach a predefined limit, a new refinement cycle begins.

Chapter 3

Mesh Refinement in a Channel with a Bump

Having presented the theoretical background of the programmed mesh refinement algorithm, a series of tests must follow for its assessment. The primary objective of this chapter is to evaluate the effectiveness of the algorithm while ensuring that the refined meshes capture all critical flow features, without introducing an unnecessary computational overload. The CFD simulations are all conducted using PUMA [2]. The results obtained from these studies will serve as verification of the algorithm's performance in simple problems, offering a valuable insight into its strengths as well as identifying potential weaknesses or limitations.

This chapter focuses on the application of the developed mesh refinement algorithm to a flow inside a straight channel with a small bump on one of its walls. As the high-speed flow reaches the bump, a shock wave is expected to form. A shock wave is an important flow feature and requires a high mesh resolution in that area. However, predetermining ts precise location is difficult, making this case ideal for the application of mesh refinement. Since the flow is transonic, the Mach number is selected as the sensor, for all cases. This problem is based on a common benchmark case that has been used widely, mainly in 2D applications, to study inviscid compressible flows [6]. In this context, two variants of this problem will be examined: one with a symmetrical bump and one with an asymmetrical.

3.1 Case A: Channel with a symmetrical bump

3.1.1 Case Description

The first variant of the case involves a symmetrical bump inside the channel. The geometry of the channel is depicted in figure 3.1. The bump's curvature is given by an analytical polynomial expression in order to allow refinement by fully respecting the original shape.

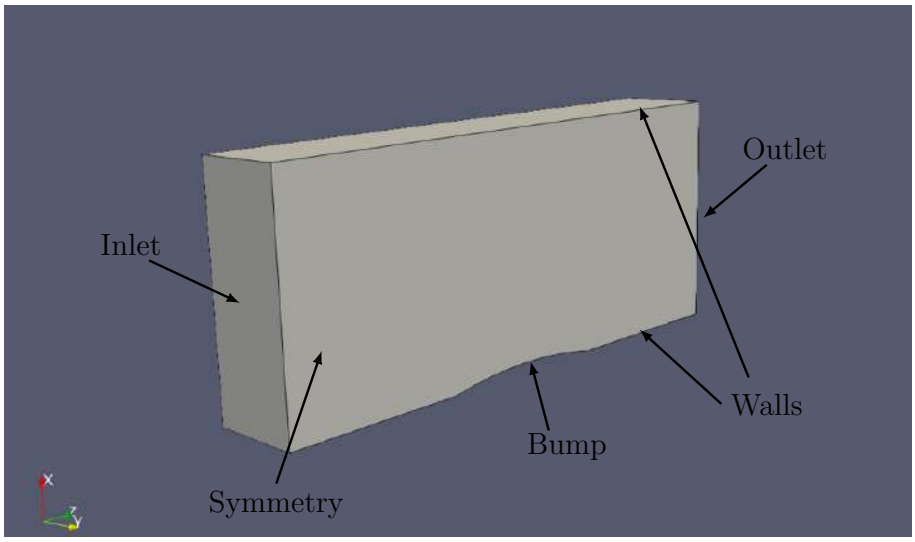


Figure 3.1: *Geometry of the channel with a bump.*

The problem involves an inviscid flow of a compressible fluid with an inlet Mach number of 0.8. The boundary conditions for the simulation (which are also noted in figure 3.1) are set as follows:

- **Inlet Condition**
  At the inlet, fixed values for the total pressure, P_t, total temperature, T_t, and flow angle are prescribed as boundary conditions. These parameters are all uniformly distributed along the inlet boundary.
- **Outlet Condition**
  At the outlet, a fixed value for the static pressure, P, is specified, which is also uniformly distributed along the outlet boundary.
- **Wall Condition**
  Since the flow is inviscid, the walls are modeled as slip walls, where the normal velocity component is equal to zero.
- **Symmetry Condition**
  A symmetry boundary condition is imposed at lateral sides of the channel.

The initial mesh generated for this case consists of 14324 nodes and 68830 tetrahedra and is illustrated in figure 3.2.

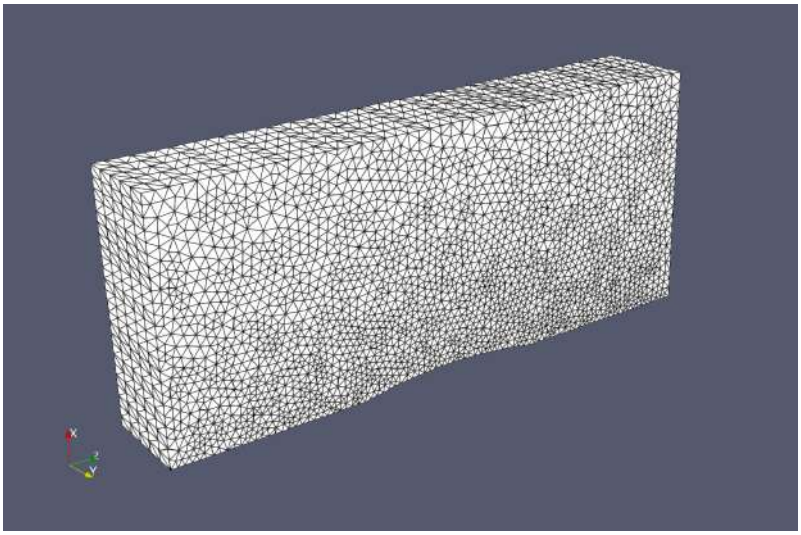


Figure 3.2: *Channel Case A: Initial mesh of the channel.*

3.1.2 Results

For this simulation, the maximum number of solver iterations is set to 5000, with 4 refinement cycles in total. Each refinement cycle is performed after every 1000 iterations of the solver. The evaluation function used is given by equation 2.1, it is not normalized and no weight functions are employed.

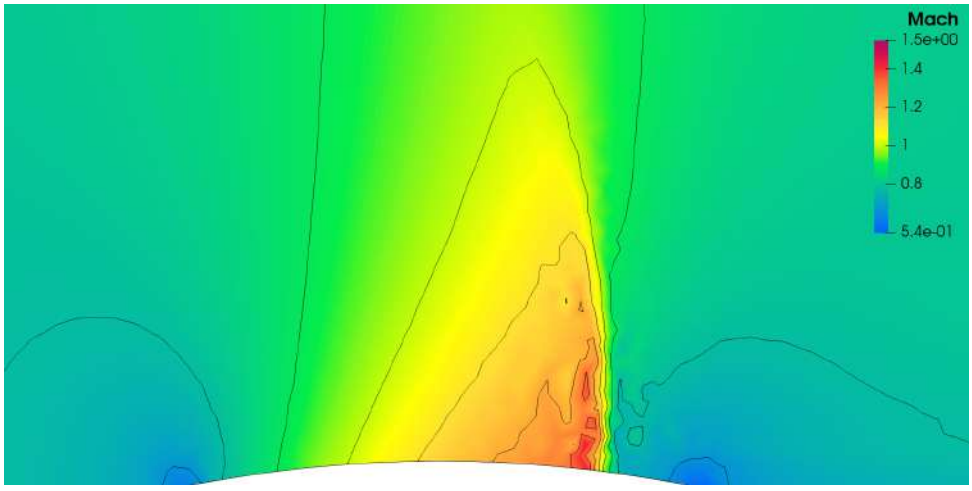


Figure 3.3: *Channel Case A: Mach number field computed on the initial mesh, after the first 1000 iterations of the solver. A shock wave formation can be spotted near the end of the bump, however due to poor initial mesh resolution, the shock wave appears artificially thick and is not captured with proper accuracy.*

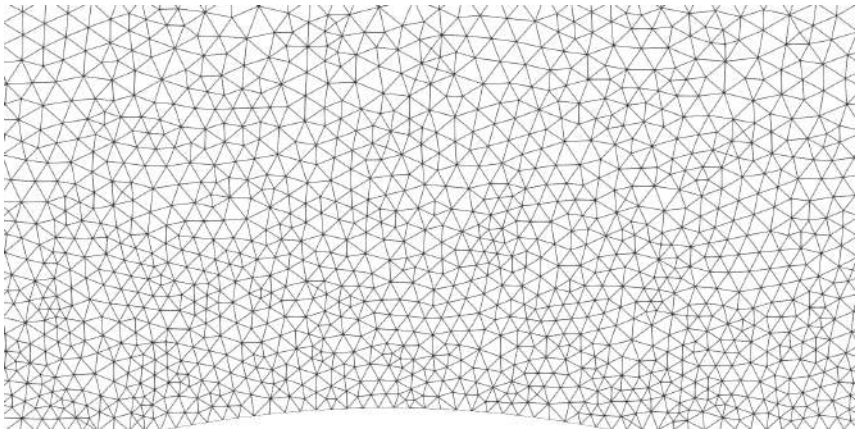


Figure 3.4: *Channel Case A: View of the initial mesh of the channel near the bump, on a cross-section along the symmetry plane.*

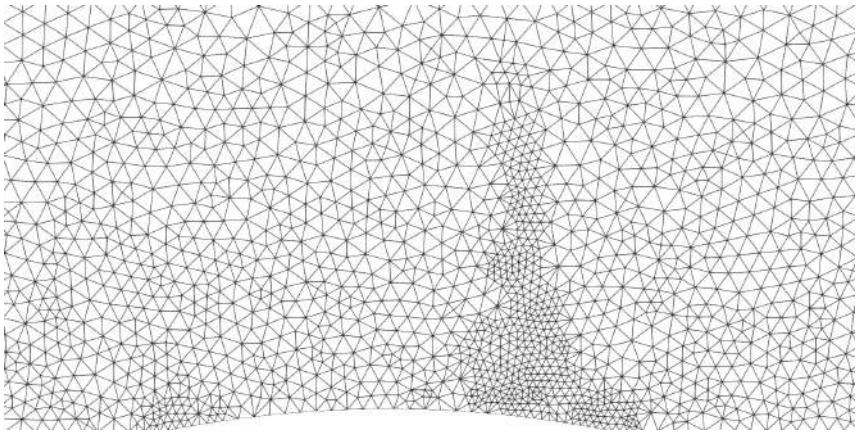


Figure 3.5: *Channel Case A: View of the mesh of the channel near the bump, on a cross-section along the symmetry plane, after the 1st refinement cycle.*

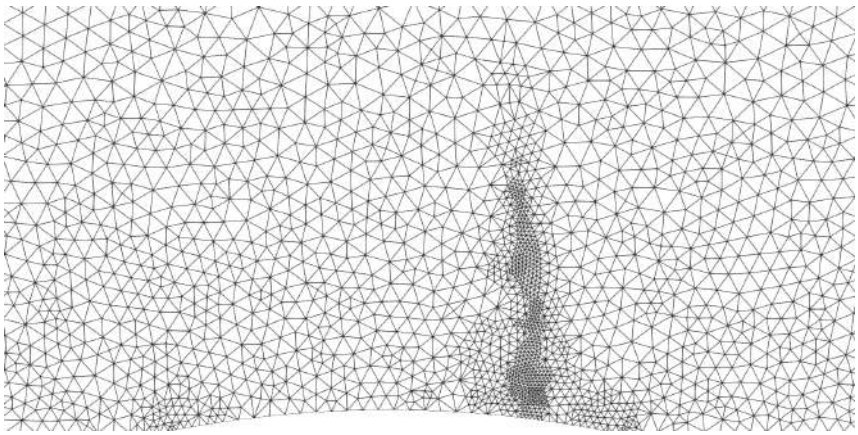


Figure 3.6: *Channel Case A: View of the mesh of the channel near the bump, on a cross-section along the symmetry plane, after the 2nd refinement cycle.*

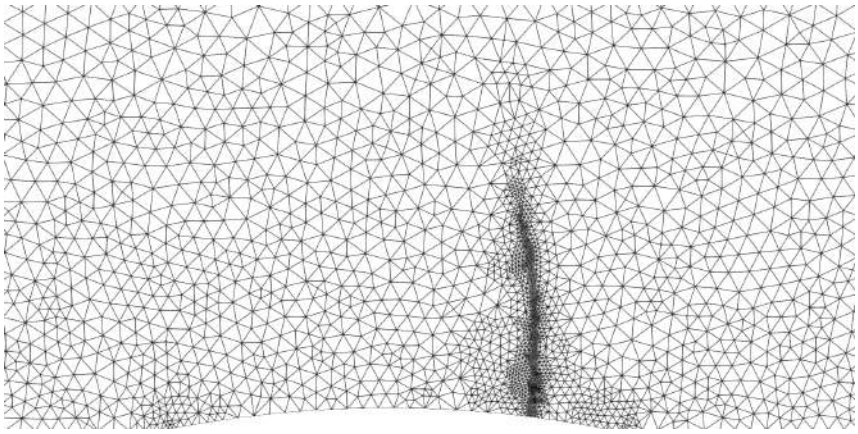


Figure 3.7: *Channel Case A: View of the mesh of the channel near the bump, on a cross-section along the symmetry plane, after the 3rd refinement cycle.*

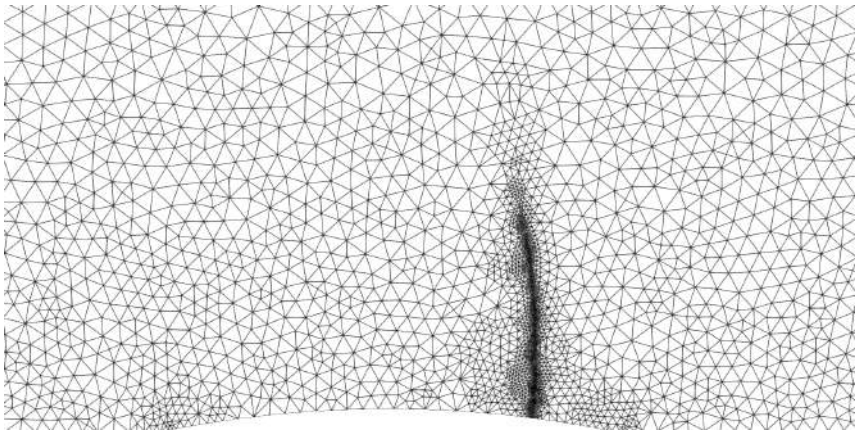


Figure 3.8: *Channel Case A: View of the mesh of the channel near the bump, on a cross-section along the symmetry plane, after the 4th refinement cycle.*

Figures 3.4- 3.8 illustrate the progression of the mesh refinement over the course of the entire simulation. With each refinement cycle, the mesh becomes increasingly more dense in the region where the shock wave is formed. Additionally, a smaller, but noticeable refinement occurs, at the first refinement cycle, near the start and end of the bump where the flow velocity is decreases.

The final mesh, after the fourth refinement cycle, demonstrates a significant concentration of elements at the shock wave's position. Notably, this final cycle also generated the highest number of elements, all concentrated at the shock wave's plane, where higher accuracy and resolution are required, with no additional refinement observed elsewhere in the domain.

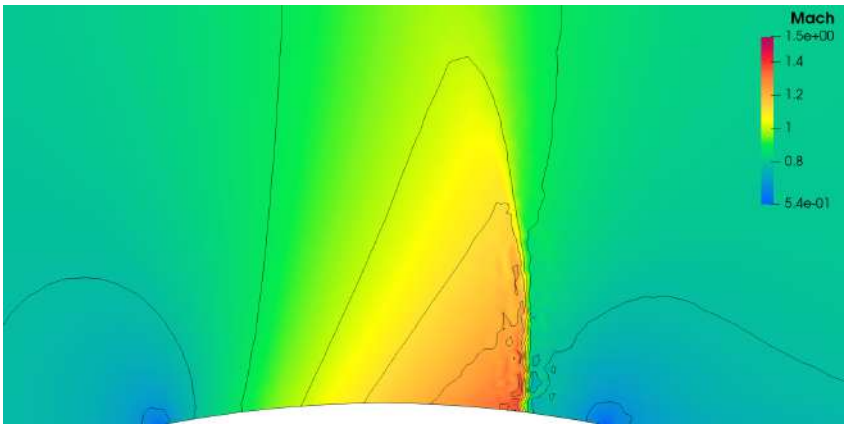


Figure 3.9: *Channel Case A: Mach number field near the bump, on a cross-section along the symmetry plane, computed using the mesh from the 1st refinement cycle.*

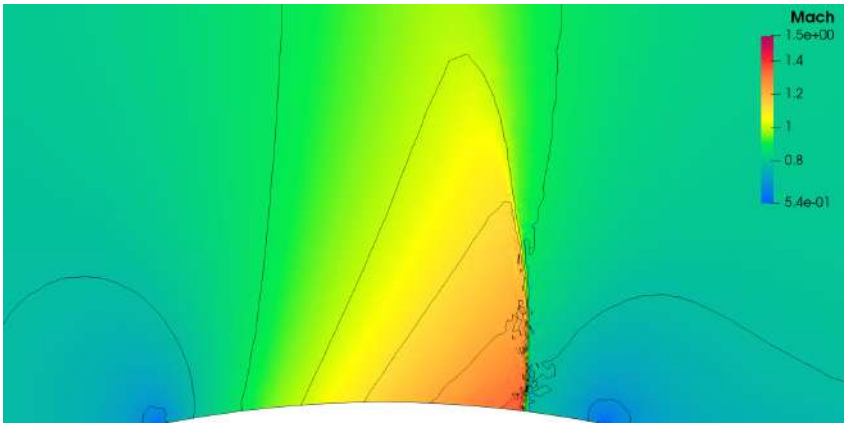


Figure 3.10: *Channel Case A: Mach number field near the bump, on a cross-section along the symmetry plane, computed using the mesh from the 2nd refinement cycle.*

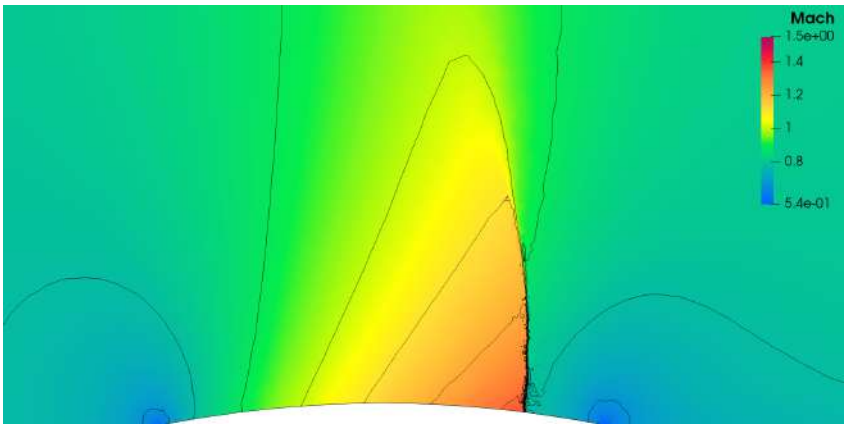


Figure 3.11: *Channel Case A: Mach number field near the bump, on a cross-section along the symmetry plane, prior to the 4th refinement cycle (computed using the mesh from the 3rd refinement cycle).*

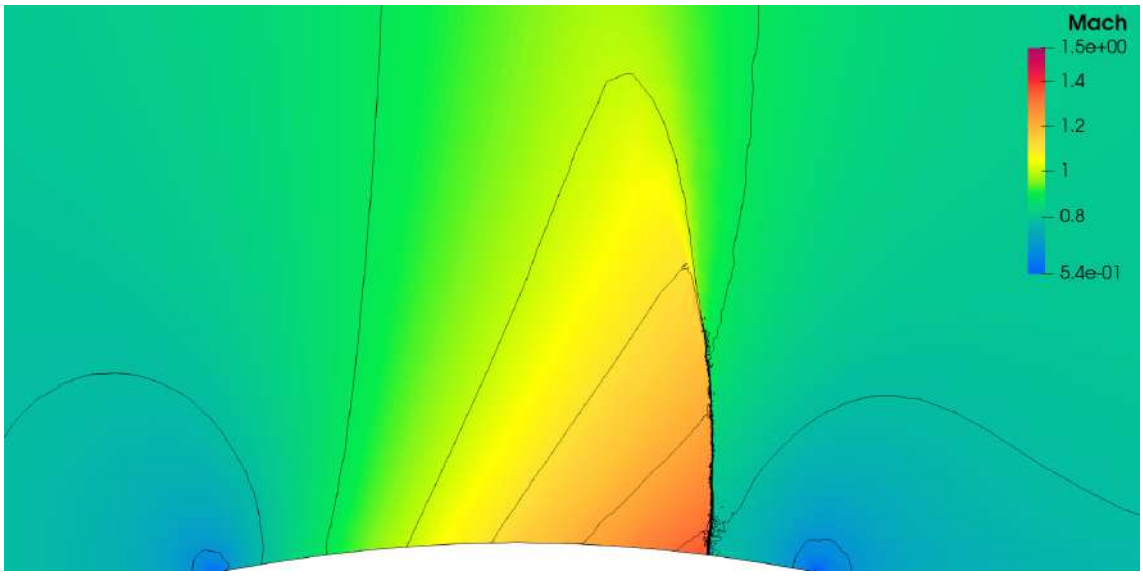


Figure 3.12: *Channel Case A: Mach number field near the bump, on a cross-section along the symmetry plane, after the 4th and final refinement cycle.*

Figures 3.9- 3.12 illustrate the progression of the Mach Number field across the refinement cycles. Initially, as shown in figure 3.3, the shock wave appears unrealistically thick with very low resolution, as a result of poor mesh quality. As the mesh refinement cycles progress, the high Mach Number gradients in the shock wave region lead to the increasing refinement of that area. Consequently, a noticeable reduction in the shock's thickness is observed from one refinement cycle to the next. By the final refinement cycle, the shock wave's thickness is significantly reduced, approaching a straight line.

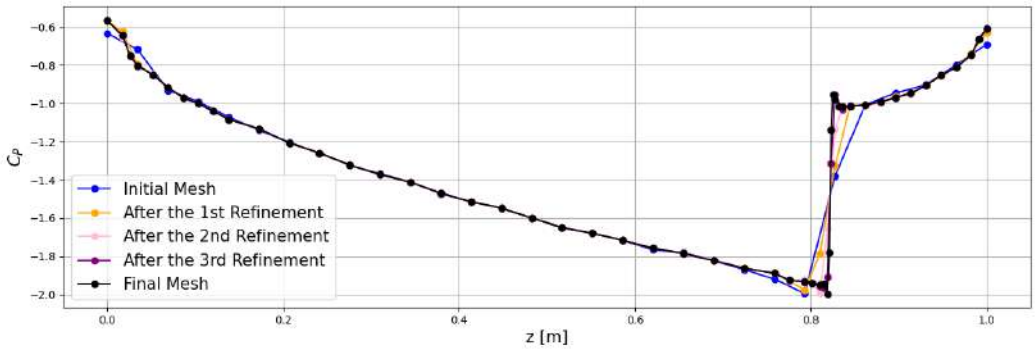


Figure 3.13: *Channel Case A: Distribution of the pressure coefficient, C_P, across the length of the bump. The precise location of the shock wave can be identified by the sudden increase of C_P. As the mesh is refined, the C_P curve at this point becomes steeper, almost vertical, effectively capturing the shock's large gradient.*

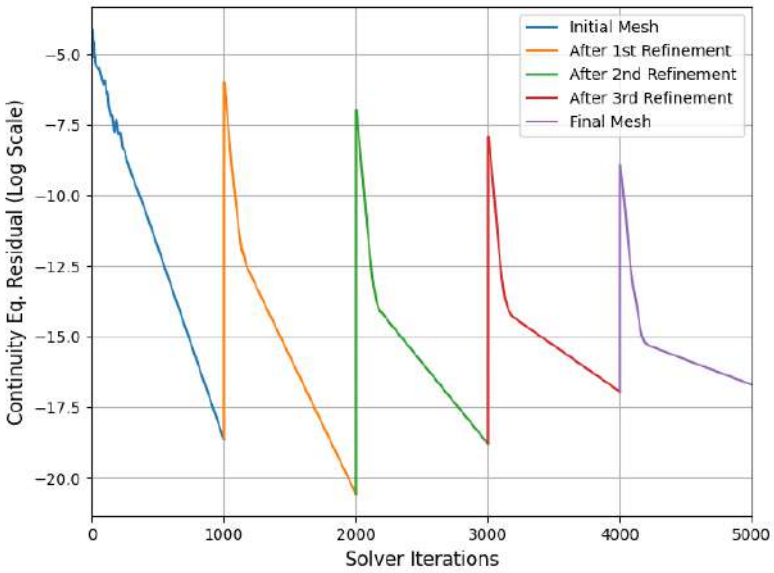


Figure 3.14: *Channel Case A: Convergence of the residual of the continuity equation, in logarithmic scale, across the entire simulation.*

Figure 3.14 shows the convergence of the residuals of the continuity equation, in logarithmic scale, across the entire simulation. When the flow equations continue to solve after a refinement cycle, a sudden increase in the residuals in observed. While an refined mesh simulation always begins on the same field the previous mesh stopped, the increased mesh resolution and accuracy of the computed results lead to this increase in the residuals. Lastly, the rate of residual convergence seems to decrease with each refinement cycle, as higher resolution and more accurate results require more iterations for convergence.

The number of nodes and tetrahedra of each one of the meshes shown in figures 3.4-3.8 is presented in table 3.1.

Mesh	**Number of nodes**	**Number of tetrahedra**
Initial Mesh	14324	68830
After the 1st Refinement	18604	91505
After the 2nd Refinement	26743	136873
After the 3rd Refinement	49972	269923
Final Mesh	122465	687557

Table 3.1: *Channel Case A: Number of nodes and tetrahedra of each mesh of the simulation.*

3.2 Case B: Channel with an asymmetrical bump

3.2.1 Case Description

The second variant of the case involves a asymmetrical bump inside the channel. While the geometry of the channel remains the same as in Case A, the surface of the bump is modified with the addition of a very small peak. This modification in added in order to introduce a 3D aspect to the flow, as the symmetry of Case A, effectively, resembled a 2D problem, with uniform characteristics along each cross-section. The purpose of Case B is to test the refinement algorithm on a genuine 3D flow field, with variants along all axes. The peak is positioned at the highest point of the bump, slightly offset from its center, as shown in figure 3.15. Both the bump and the peak are described using analytical equations. Thus, the refinement at the walls is performed by respecting its shape.

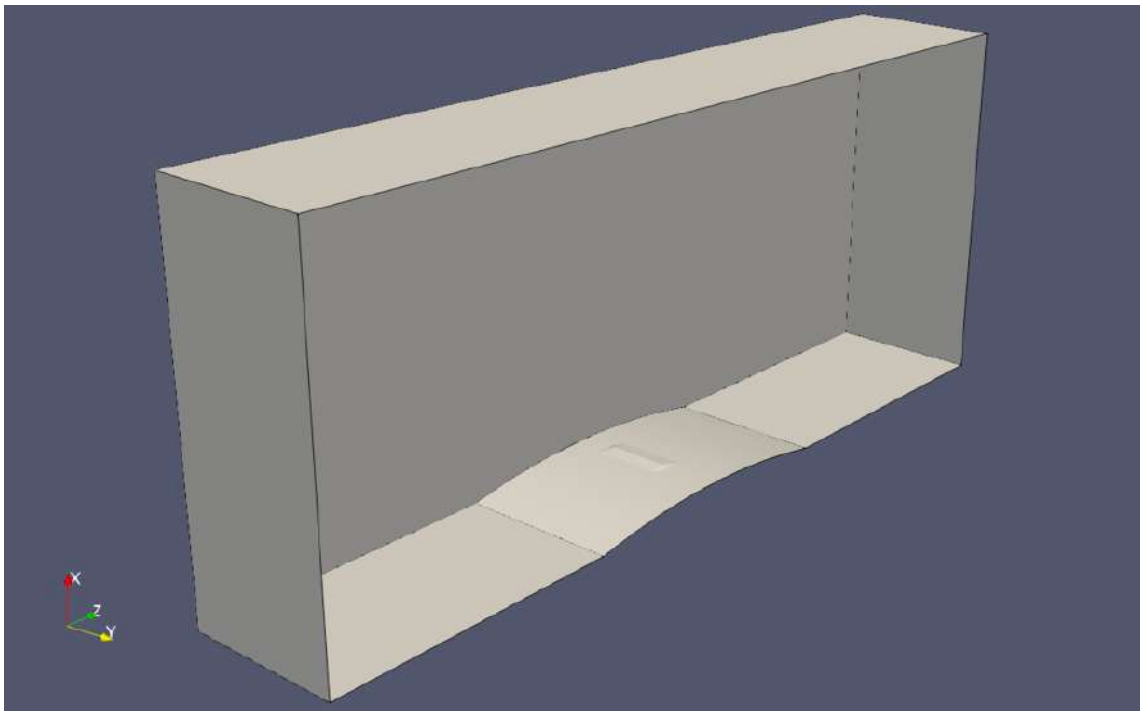


Figure 3.15: *Channel Case B: View of the modified bump.*

The flow conditions are the same as in Case A, specifically an inviscid flow of a compressible fluid with an inlet Mach number of 0.8. Similarly, the boundary conditions remain the same as in Case A, as illustrated in figure 3.1.

The initial mesh generated for this case consists of 52783 nodes and 289590 tetrahedra. This increased number of nodes and elements on the initial mesh, compared to Case A, intends to accurately capture the modified geometry of the bump, as well

as the 3D characteristics of the flow. To achieve this, a finer discretization is applied across the y-direction, as shown in figure 3.16.

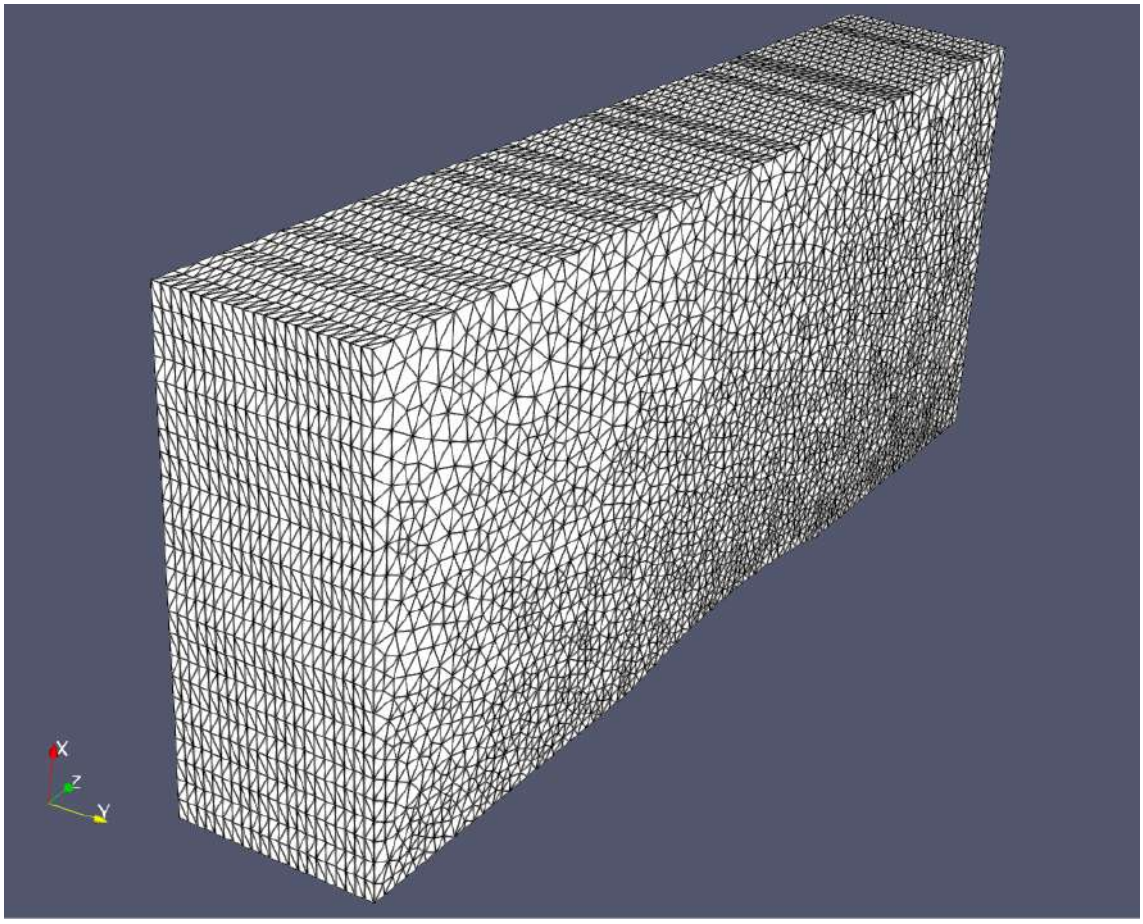


Figure 3.16: *Channel Case B: Initial mesh of the channel.*

3.2.2 Results

For this simulation, the maximum number of the flow solver iterations is set to 5000, with 4 refinement cycles in total, each one performed after every 1000 iterations of the solver, similar to Case A. The same evaluation function as Case A is also used, given by equation 2.1.

To verify, and showcase, the 3D features of the flow field and the refinement, the results are presented across two different cross-sections of the channel: one along the symmetry plane (as in Case A), and one along a cross-section that includes the modified bump with the added peak.

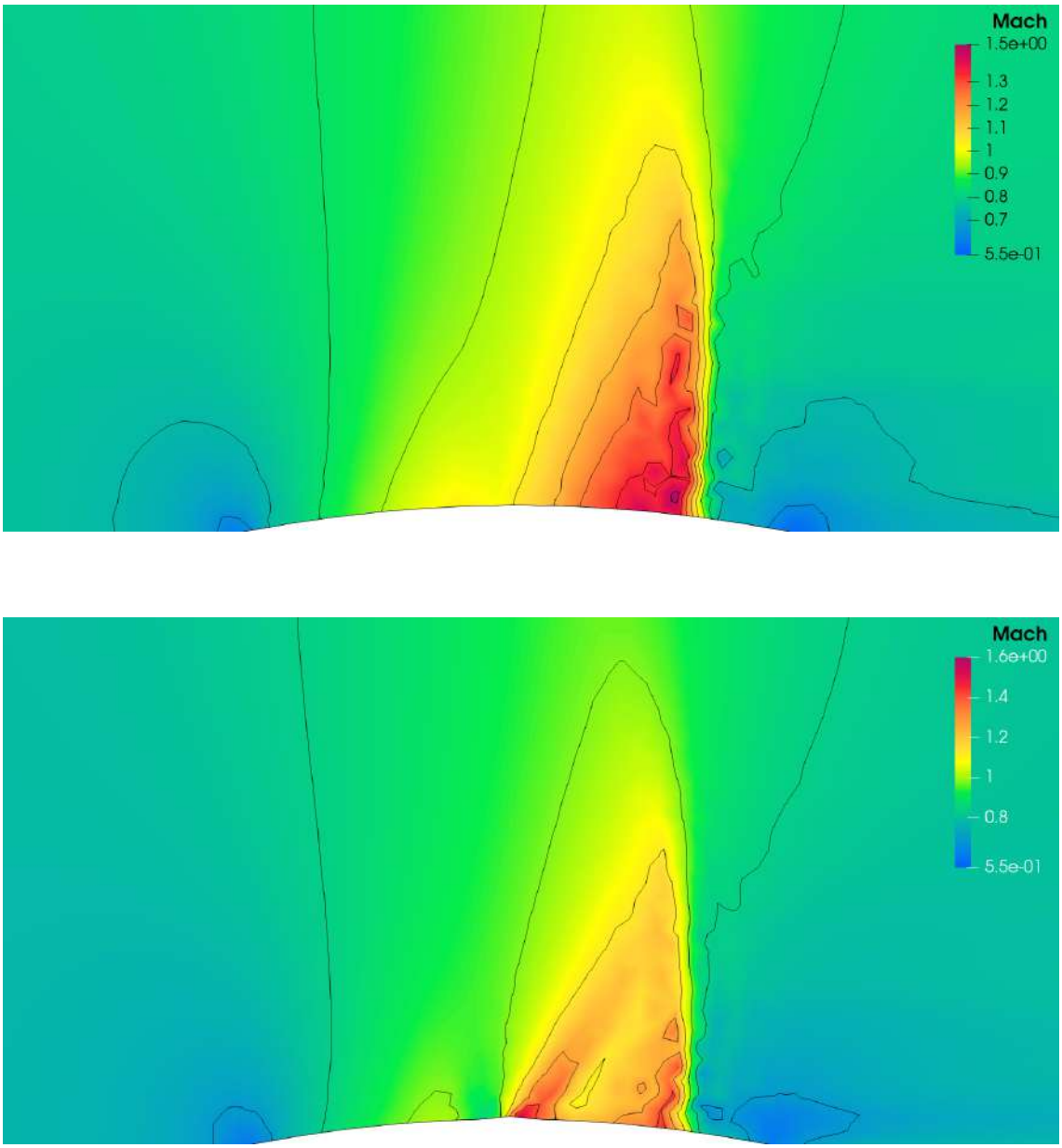


Figure 3.17: *Channel Case B: Mach number field computed on the initial mesh, on a cross-section across the symmetry plane (upper) and one that includes the peak (lower), after the first 1000 iterations of the solver. A shock wave can be spotted near the end of the bump. The two Mach number fields differ due to the influence of the peak on the bump.*

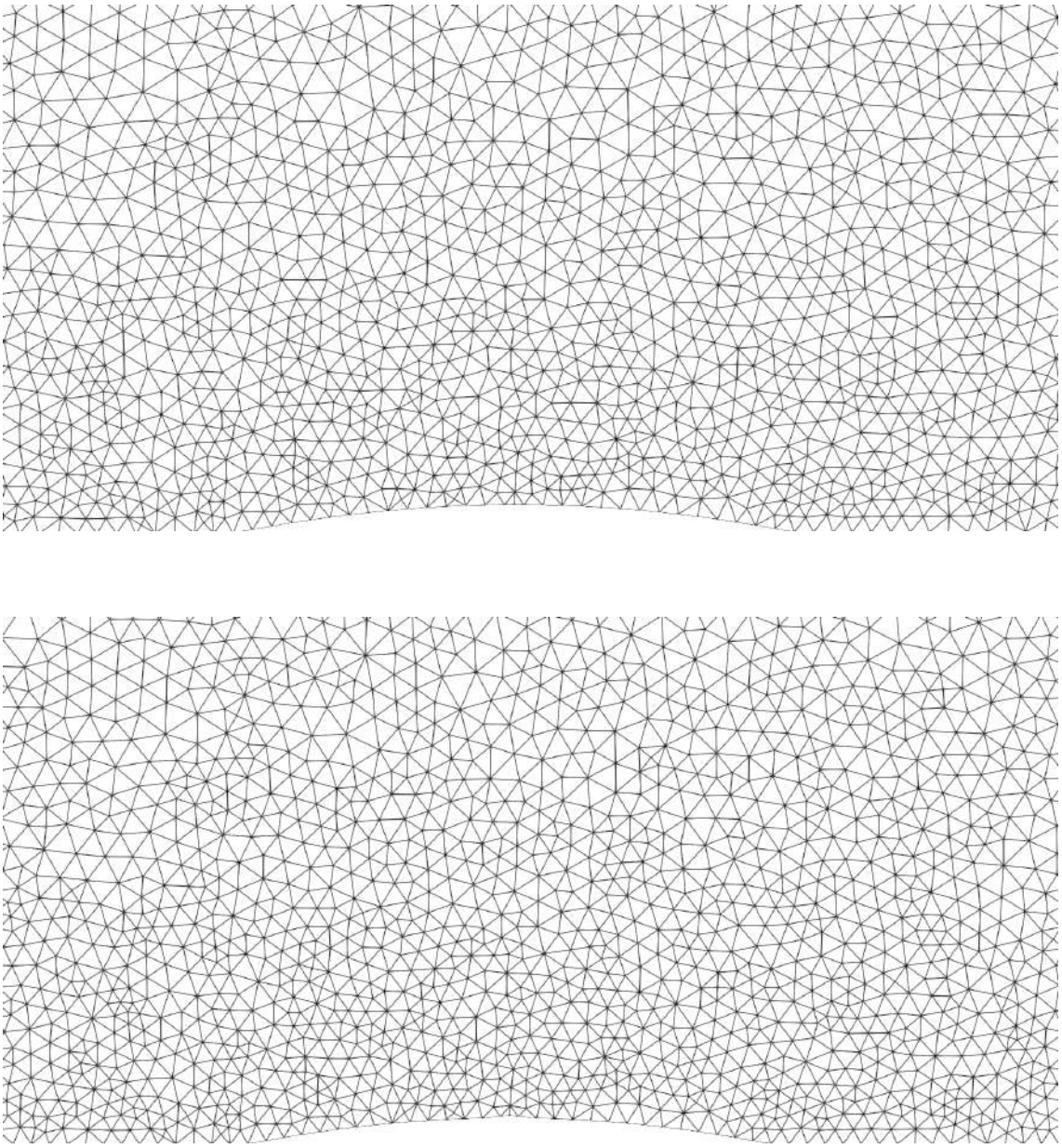


Figure 3.18: *Channel Case B: View of the initial mesh of the channel near the bump, on a cross-section along the symmetry plane (upper) and one that includes the peak (lower).*

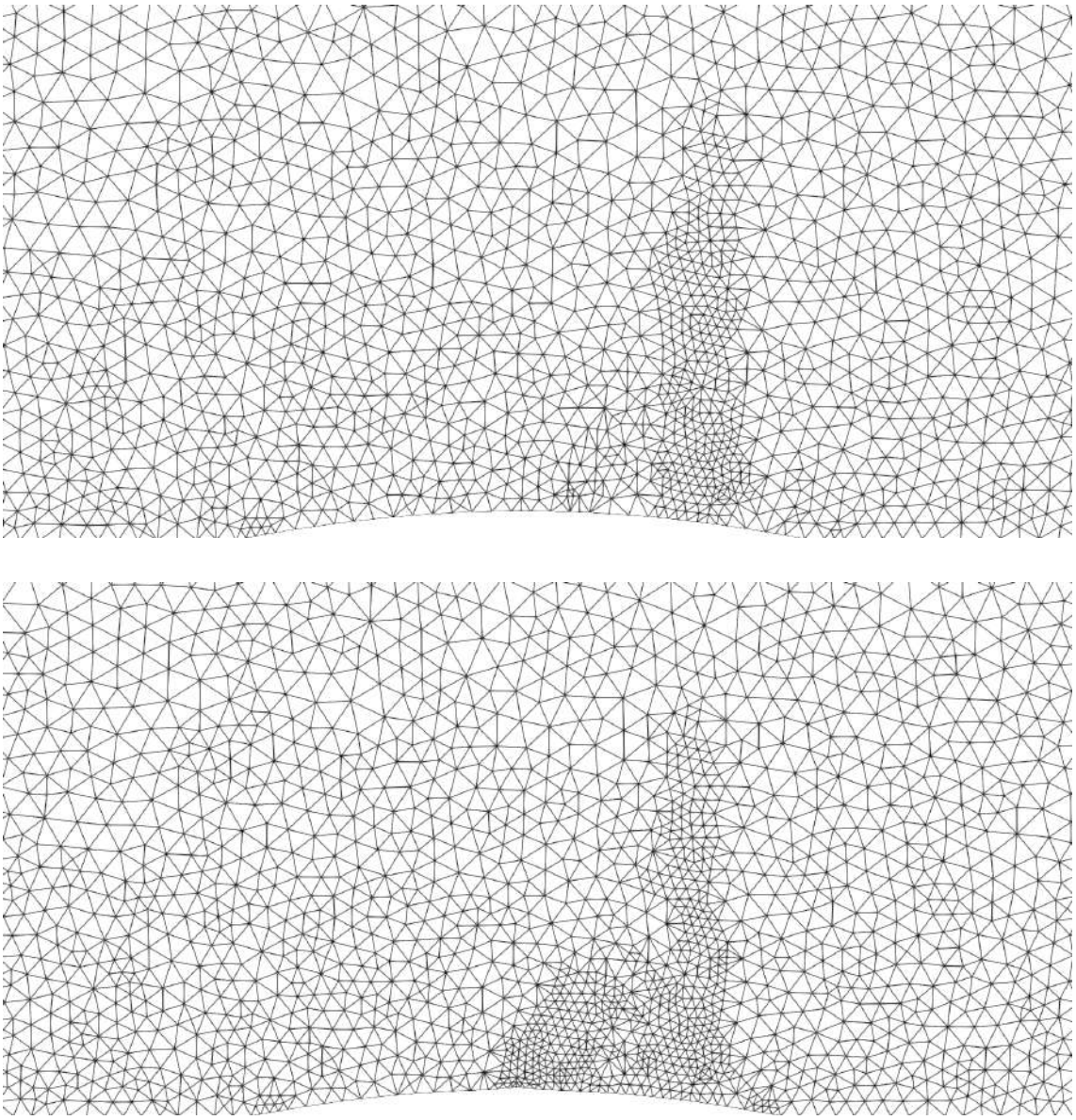


Figure 3.19: *Channel Case B: View of the mesh of the channel near the bump, on a cross-section along the symmetry plane (upper) and one that includes the peak (lower), after the 1st refinement cycle.*

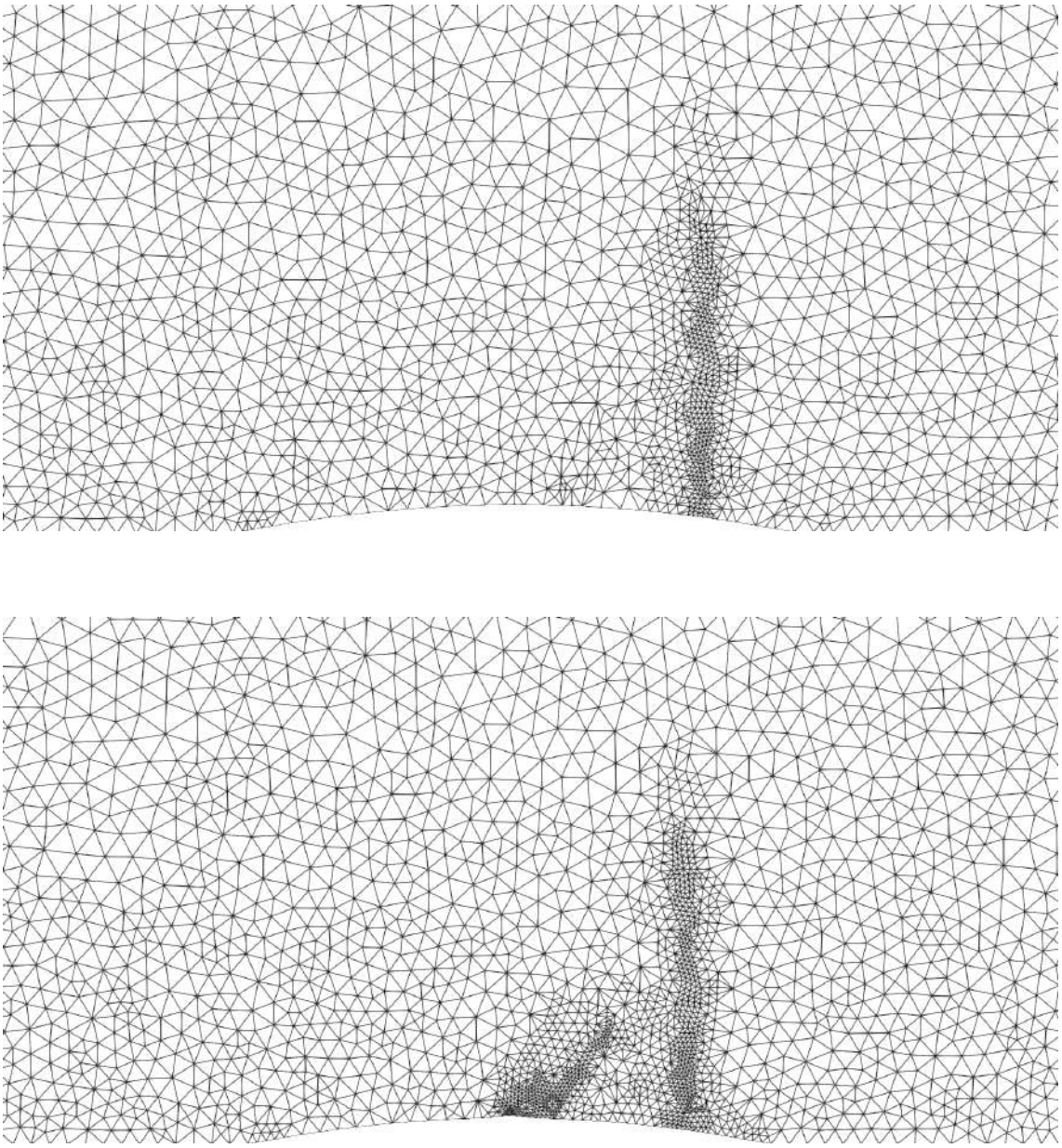


Figure 3.20: *Channel Case B: View of the mesh of the channel near the bump, on a cross-section along the symmetry plane (upper) and one that includes the peak (lower), after the 2nd refinement cycle.*

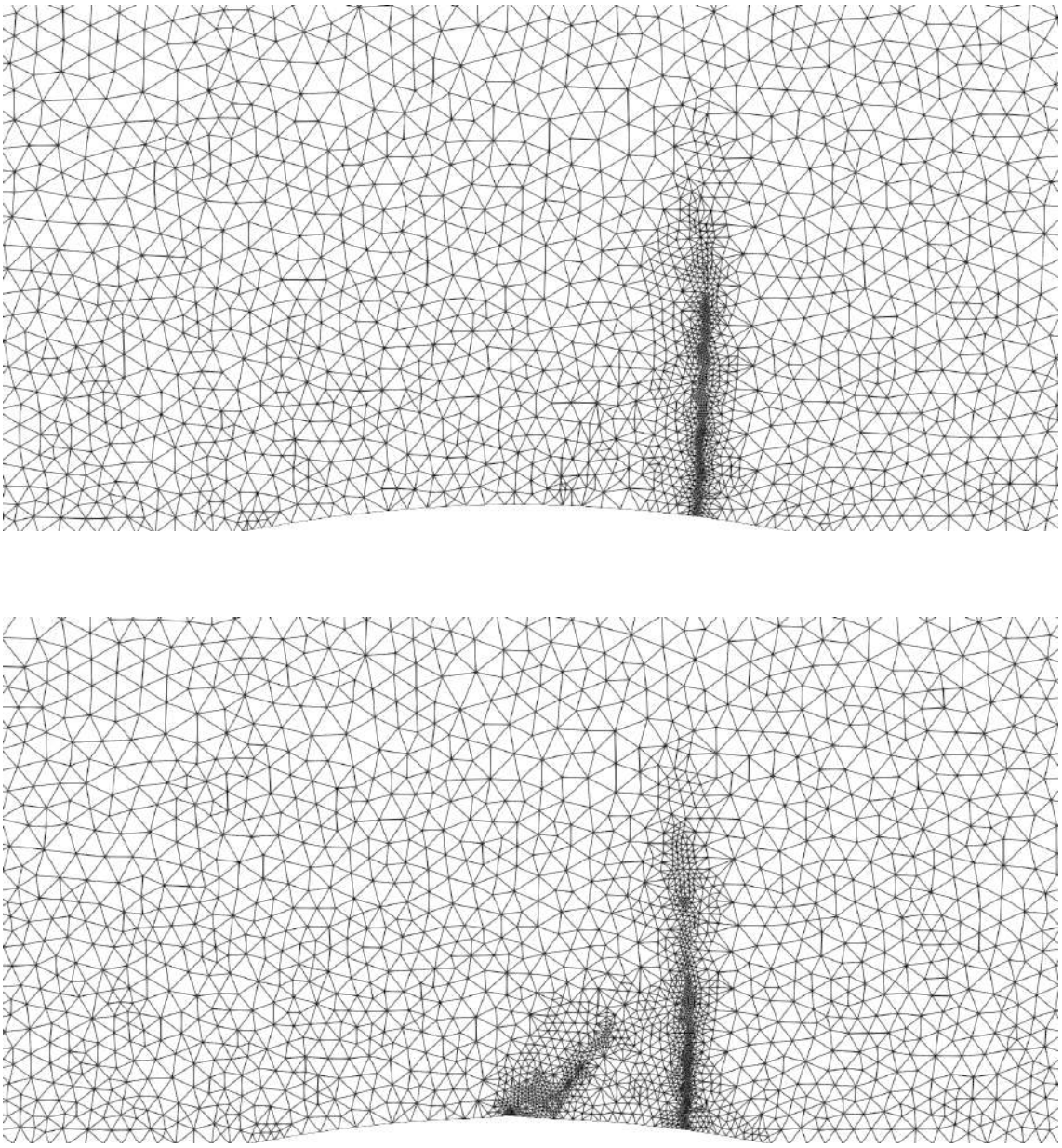


Figure 3.21: *Channel Case B: View of the mesh of the channel near the bump, on a cross-section along the symmetry plane (upper) and one that includes the peak (lower), after the 3rd refinement cycle.*

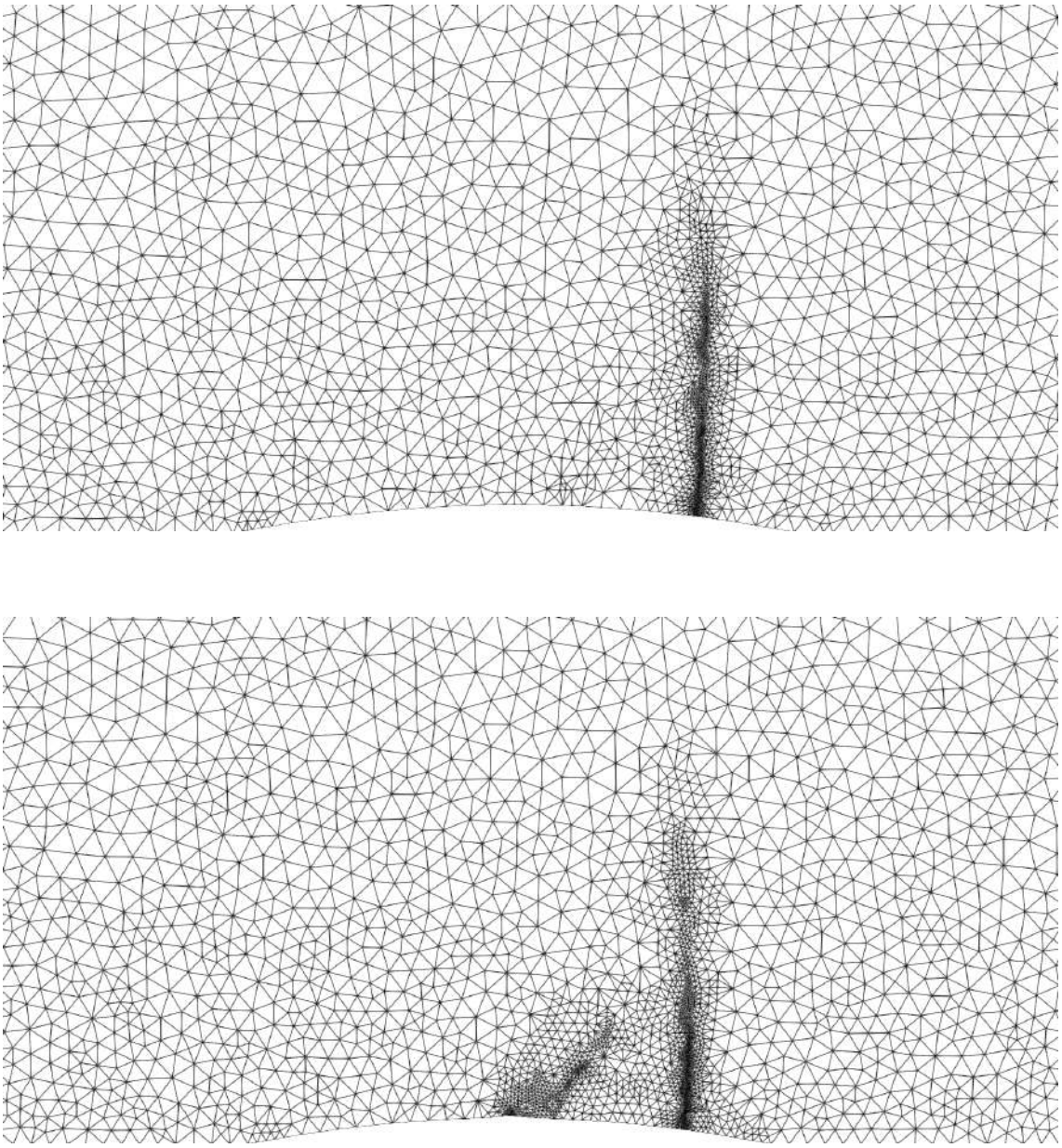


Figure 3.22: *Channel Case B: View of the mesh of the channel near the bump, on a cross-section along the symmetry plane (upper) and one that includes the peak (lower), after the 4th refinement cycle.*

Figures 3.18- 3.22 illustrate the progression of the mesh refinement over the course of the entire simulation, across two different cross-sections: one along the symmetry plane and one through the center of the added peak. With each refinement cycle, the mesh becomes progressively denser, in both cross-sections, in the region where the shock wave appears. However, an additional refinement is observed near the peak, specifically at its downstream. This occurs because the peak alters the flow field and generates a region of elevated Mach number gradients, which trigger the

mesh refinement.

The final mesh, after the fourth refinement cycle, exhibits the largest concentration of elements at the shock wave's position, as well as the downstream of the peak. The increased number of nodes and tetrahedra of the final mesh, compared to the final mesh of Case A, is attributed to the higher element count in the initial mesh, which was constructed in order to effectively capture the 3D features of the flow, as previously noted.

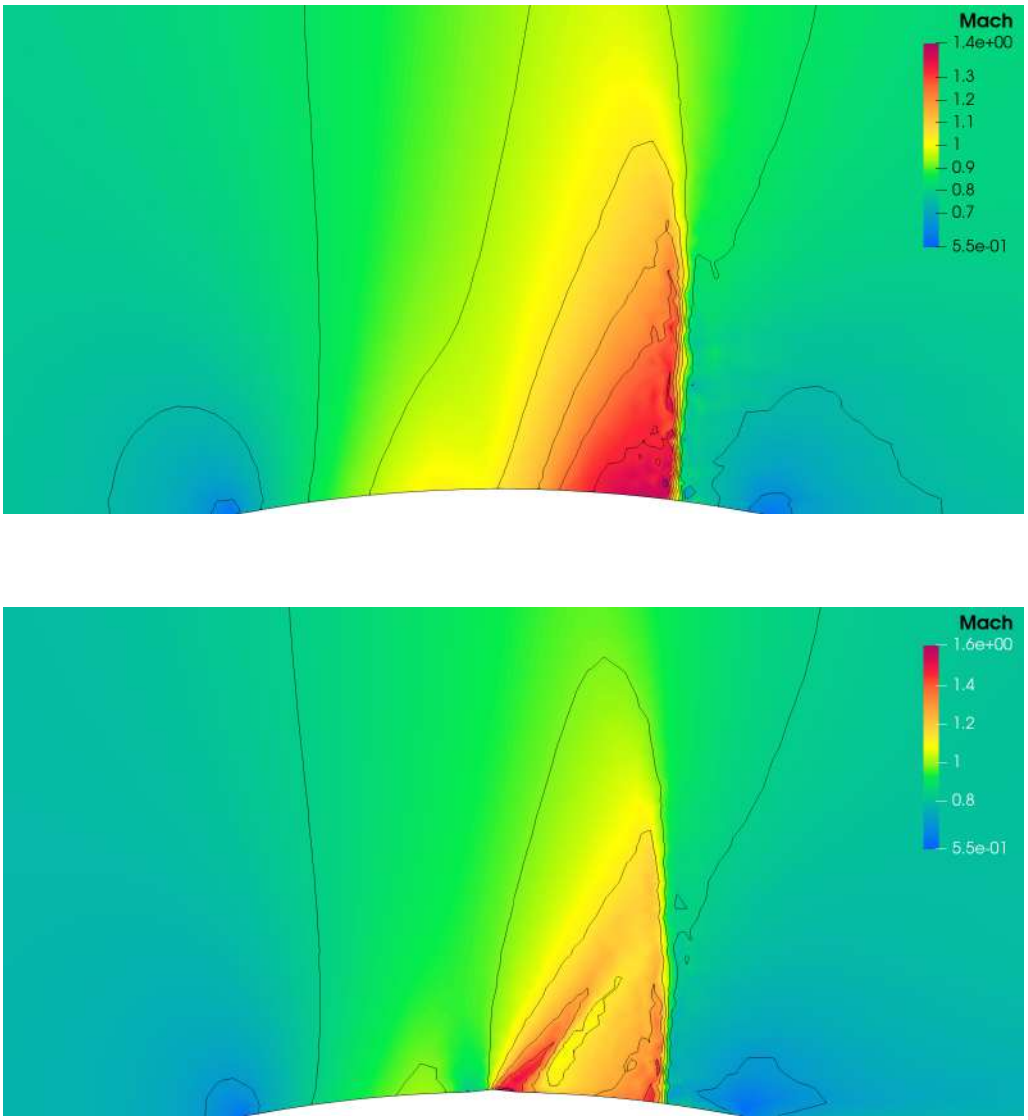


Figure 3.23: *Channel Case B: Mach number field near the bump, on a cross-section across the symmetry plane (upper) and one that includes the peak (lower), prior to the 2nd refinement cycle (computed using the mesh from the 1st refinement cycle).*

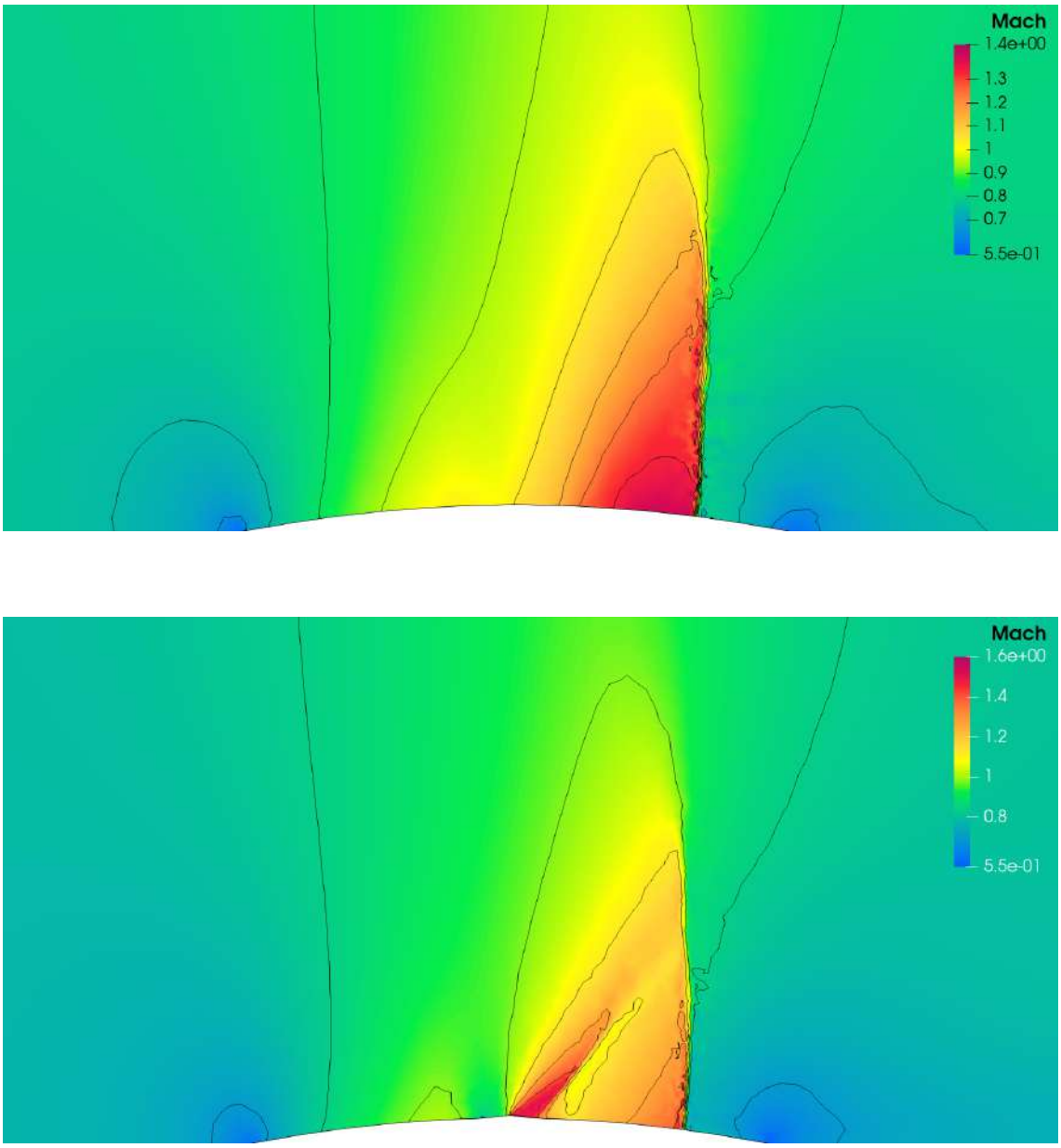


Figure 3.24: *Channel Case B: Mach number field near the bump, on a cross-section across the symmetry plane (upper) and one that includes the peak (lower), prior to the 3rd refinement cycle (computed using the mesh from the 2nd refinement cycle).*

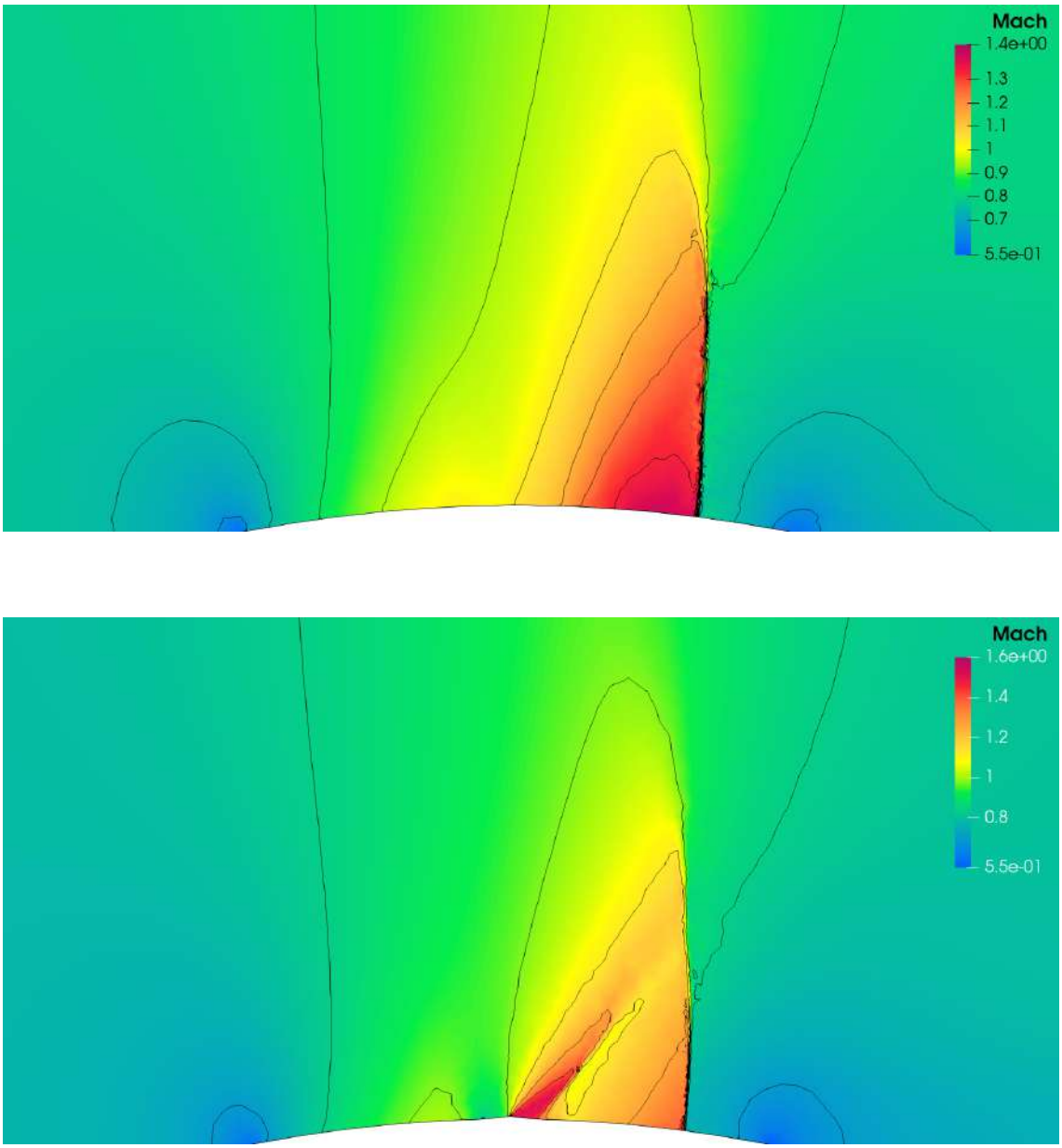


Figure 3.25: *Channel Case B: Mach number field near the bump, on a cross-section across the symmetry plane (upper) and one that includes the peak (lower), prior to the 4th refinement cycle (computed using the mesh from the 3rd refinement cycle).*

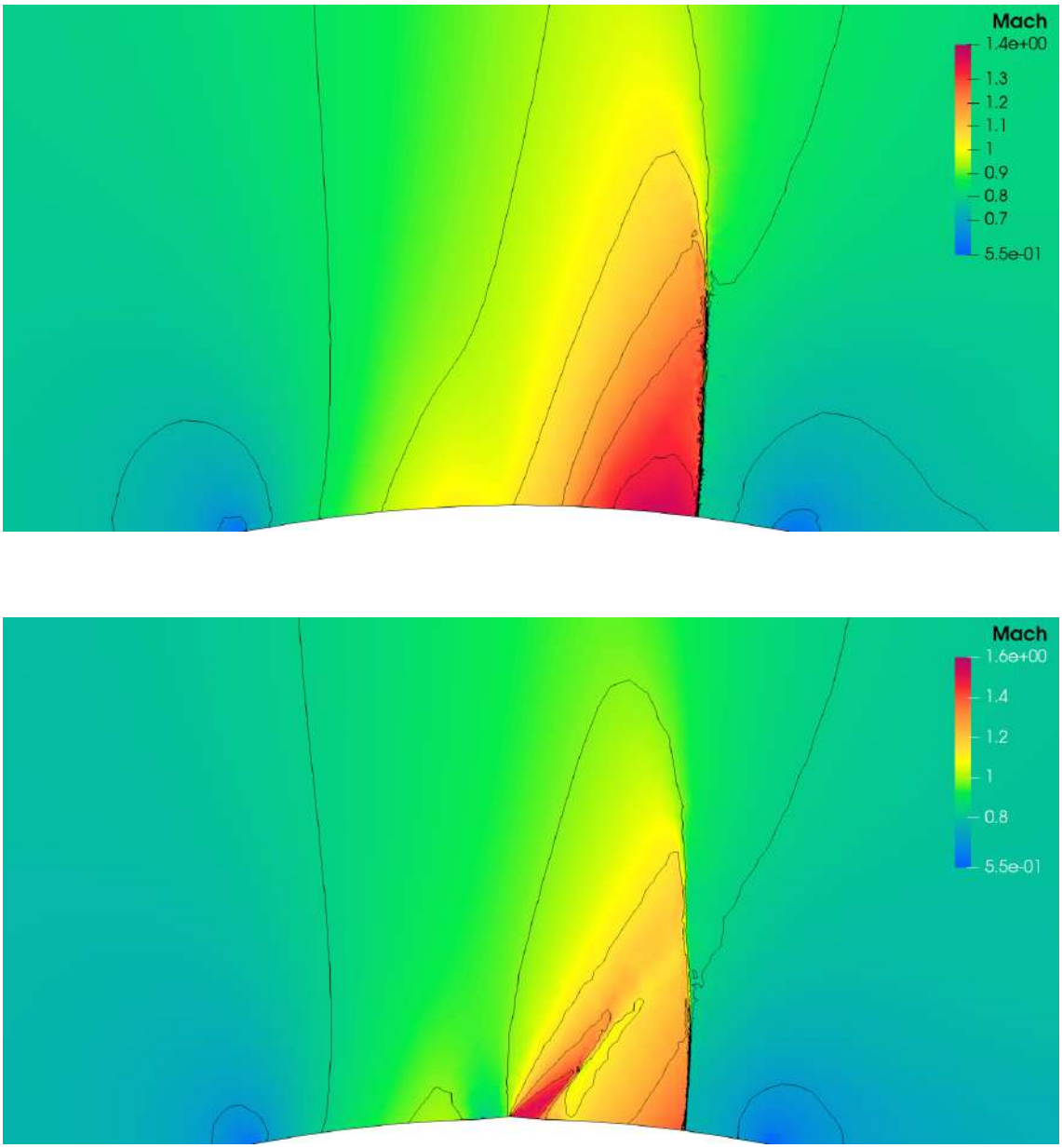


Figure 3.26: *Channel Case B: Mach number field near the bump, on a cross-section across the symmetry plane (upper) and one that includes the peak (lower), after the 4th and final refinement cycle.*

Figures 3.23- 3.26 illustrate the progression of the Mach number field across the refinement cycles in two different cross-sections: one along the symmetry plane and one through the center of the added peak. Initially, as shown in figure 3.17, a shock wave appears unrealistically thick with very low resolution, due to the poor initial mesh quality. Additionally near the peak, a localized disturbance in the flow field is observed, due to its presence.

Similar to Case A, the high Mach number gradients lead to the refinement of these

areas of the domain. By the end of the refinement, the shock wave's thickness is significantly reduced, in both cross-sections, while the flow field downstream of the peak is smoother and more accurately resolved, demonstrating the refinement algorithm's effectiveness in handling all aspects of a 3D flow.

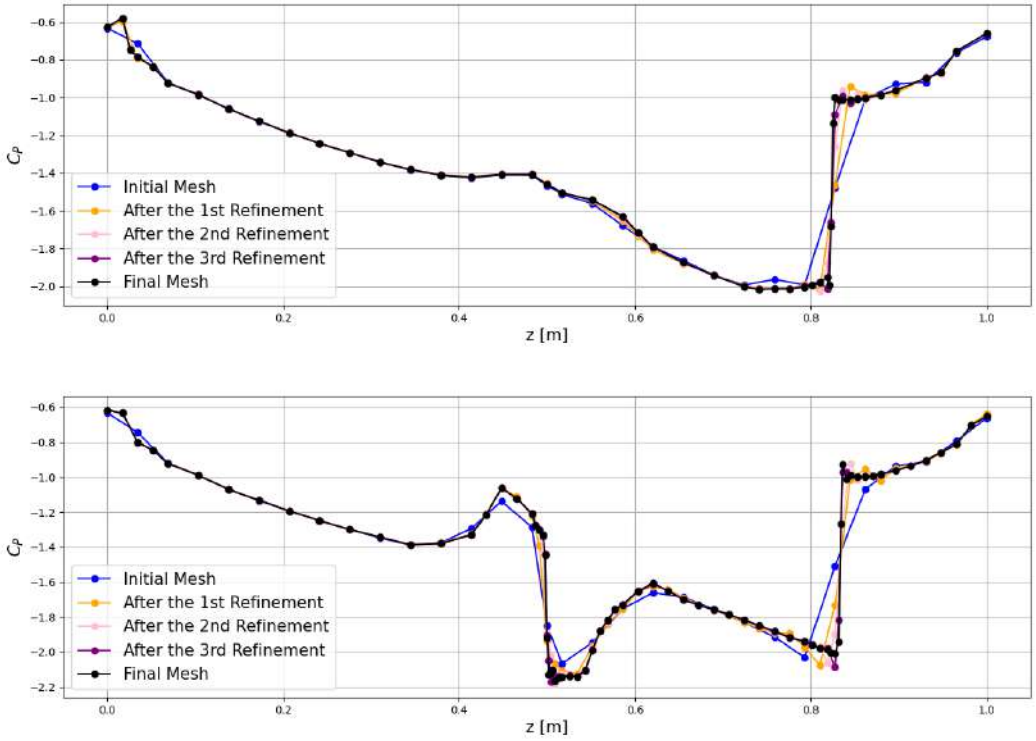


Figure 3.27: *Channel Case B: Distribution of the pressure coefficient, C_P, across the length of the bump on a cross-section along the symmetry plane (upper) and one that includes the peak (lower).*

In both graphs shown in figure 3.27, the location of the shock wave can be identified by the sudden increase of C_P. Additionally, the influence of the added peak, at the center of the bump, can be observed with a large decrease in C_P. As the mesh is refined and the resolution increases, the the C_P curve in those areas becomes increasingly steeper, almost horizontal, effectively capturing the large gradients.

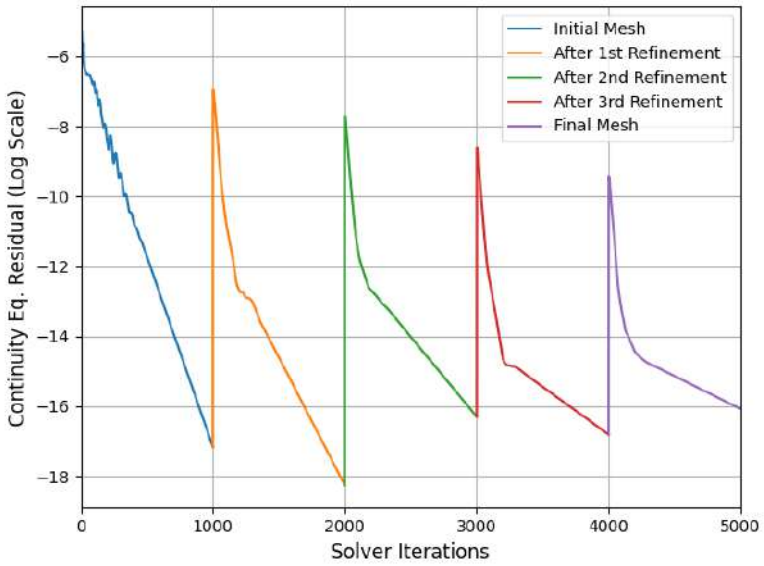


Figure 3.28: *Channel Case B: Convergence of the residual of the continuity equation, in logarithmic scale, across the entire simulation.*

The residuals of the continuity equation, shown in figure 3.28, follow a trend similar with those in Case A (figure 3.14). However, a slight increase in their magnitude can be observed, across all refinement cycles, compared to Case A. This can be attributed to the more complex geometry and resulting flow field introduced by the asymmetric addition of the small peak at the top of the bump.

Lastly, the number of nodes and tetrahedra of each one of the meshes shown in figures 3.18- 3.22 is presented in table 3.2.

Mesh	**Number of nodes**	**Number of tetrahedra**
Initial Mesh	52783	289590
After the 1st Refinement	68526	378391
After the 2nd Refinement	101872	570021
After the 3rd Refinement	160504	905766
Final Mesh	273922	1554873

Table 3.2: *Channel Case B: Number of nodes and tetrahedra of each mesh of the simulation.*

Chapter 4

Mesh Refinement in a Flow Around a Sphere

This chapter focuses on the application of the mesh refinement algorithm to an external aerodynamic problem, specifically the flow around a sphere. The problem setup is based on and compared to a previous study on compressible, inviscid transient flows around spheres at different Mach numbers [5]. All CFD simulations are conducted using PUMA [2] and the results presented in this chapter are compared with [5].

Two variants of the problem will be examined: one with a freestream Mach number of 0.8 and one with a freestream Mach number of 0.95. The flow around a sphere is inherently a transient problem, however the steady-state mean flow equations will be solved for both cases in order to provide a simplified yet insightful analysis regarding mesh refinement. It is also important to highlight that the reference study [5] investigates a transient flow and consequently, the results obtained from this chapter will be compared with the corresponding mean values computed over the time span of the transient simulation conducted in that study, in order to ensure consistency. As showcased in that study, the flow is transonic, therefore the Mach number is selcted as the sensor for the refinement. In both cases, the sphere has a radius of 1m and its surface is described using an analytical equation in order to allow refinement by fully respecting the original shape. All the simulations begin with a coarse initial mesh that consists of 3973 nodes and 21129 tetrahedra and is illustrated in figure 4.1.

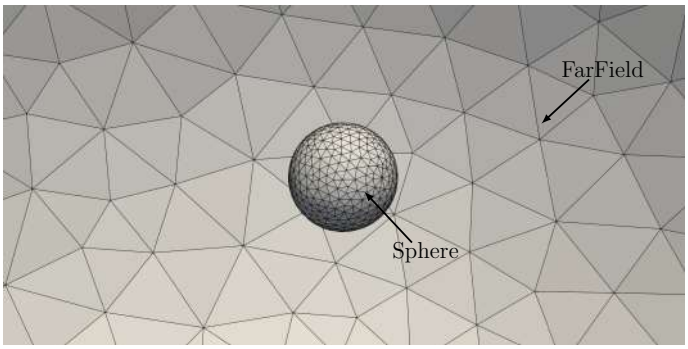


Figure 4.1: *Geometry of the initial mesh of the sphere case.*

4.1 Case A: Freestream Mach 0.8

4.1.1 Case Description

The first case considers a freestream Mach number of 0.8, with an inviscid flow of a compressible fluid. The boundary conditions, are set as noted in figure 4.1.

4.1.2 Results

For this simulation, the maximum number of solver iterations is set to 6000, with 3 refinement cycles in total. Each refinement cycle is performed after every 1500 iterations of the solver.

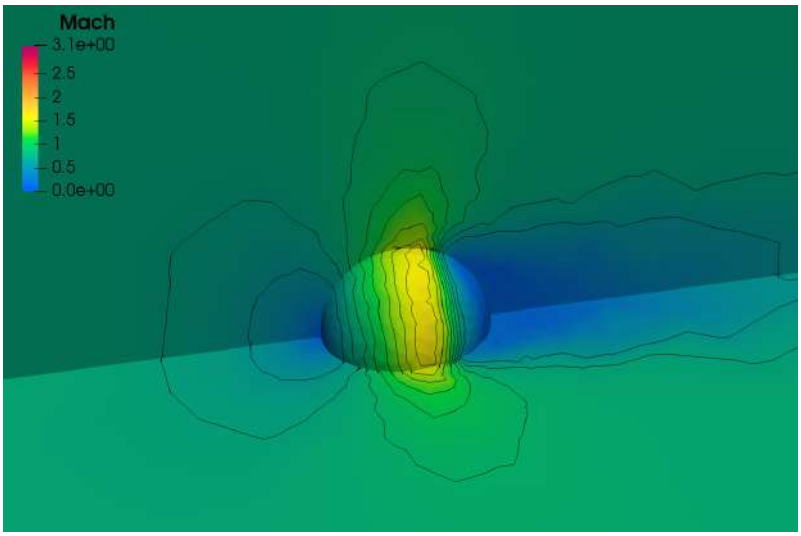


Figure 4.2: *Sphere Case A: Mach number field computed on the initial mesh near the sphere, presented in a quarter cross-section, after the first 1500 solver iterations. The poor initial mesh resolution causes the shock wave to be unrealistically thick.*

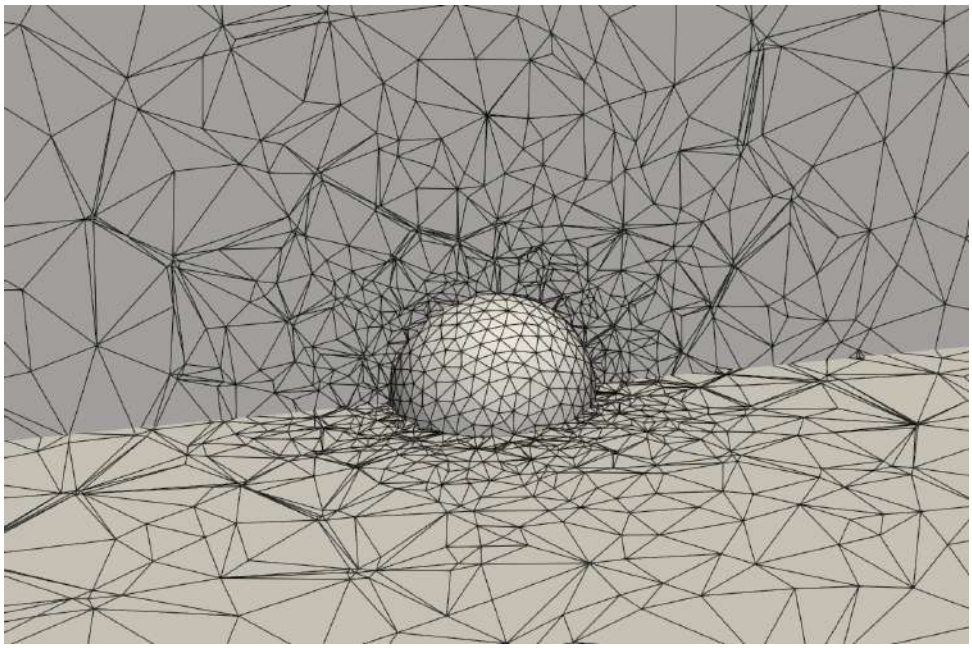


Figure 4.3: *Sphere Case A: View of the initial mesh of the channel near the sphere, presented in a quarter cross-section.*

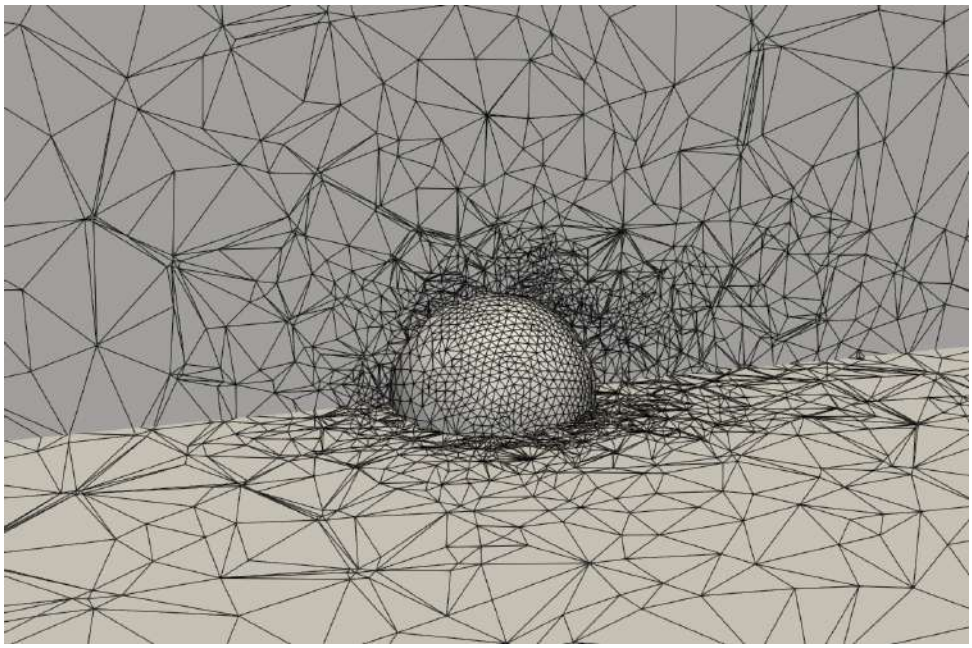


Figure 4.4: *Sphere Case A: View of the mesh near the sphere, presented in a quarter cross-section, after the 1st refinement cycle. The mesh consists of 10727 nodes and 58125 tetrahedra.*

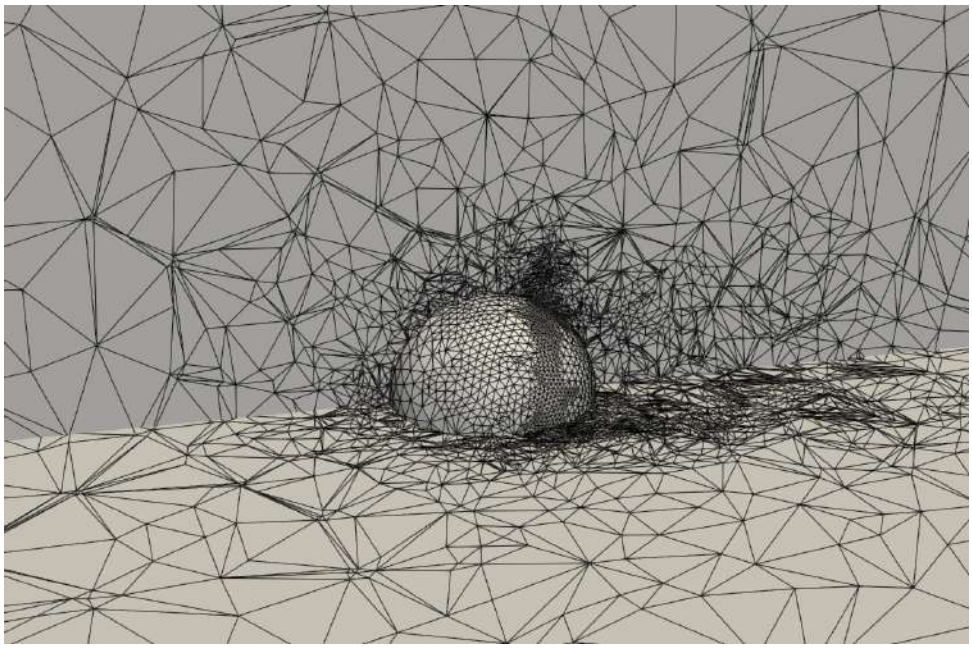


Figure 4.5: *Sphere Case A: View of the mesh near the sphere, presented in a quarter cross-section, after the 2nd refinement cycle.*

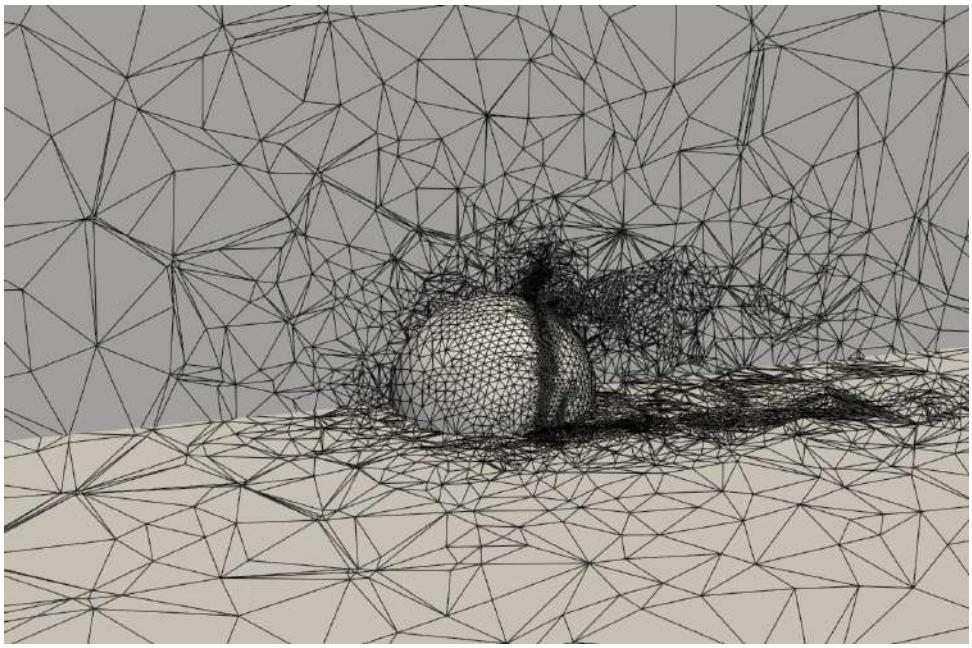


Figure 4.6: *Sphere Case A: View of the mesh near the sphere, presented in a quarter cross-section, after the 3rd refinement cycle.*

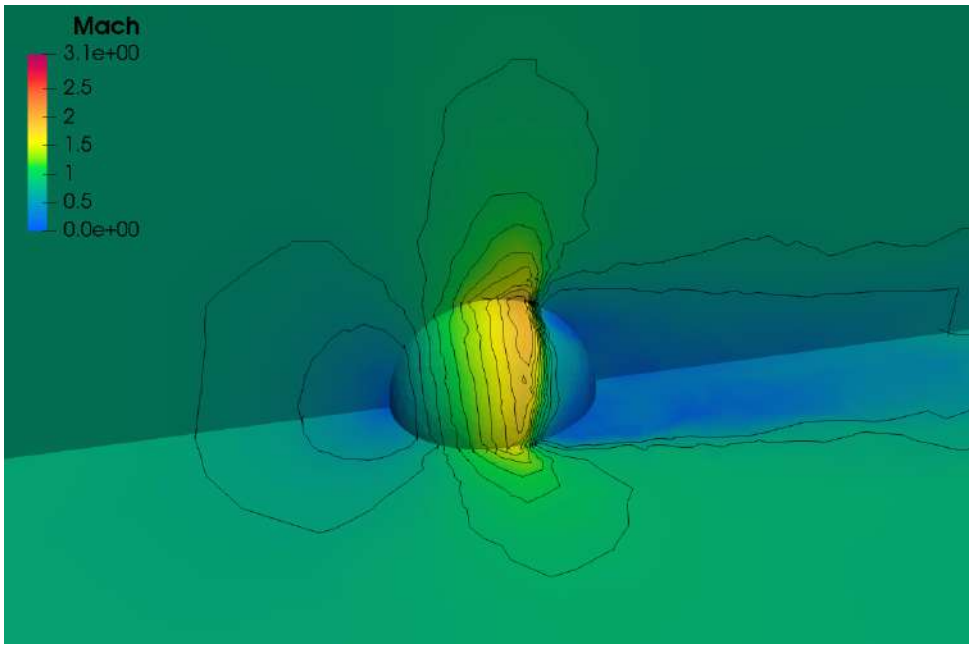


Figure 4.7: *Sphere Case A: Mach number field near the sphere, presented in a quarter cross-section, prior to the 2nd refinement cycle (computed using the mesh from the 1st refinement cycle).*

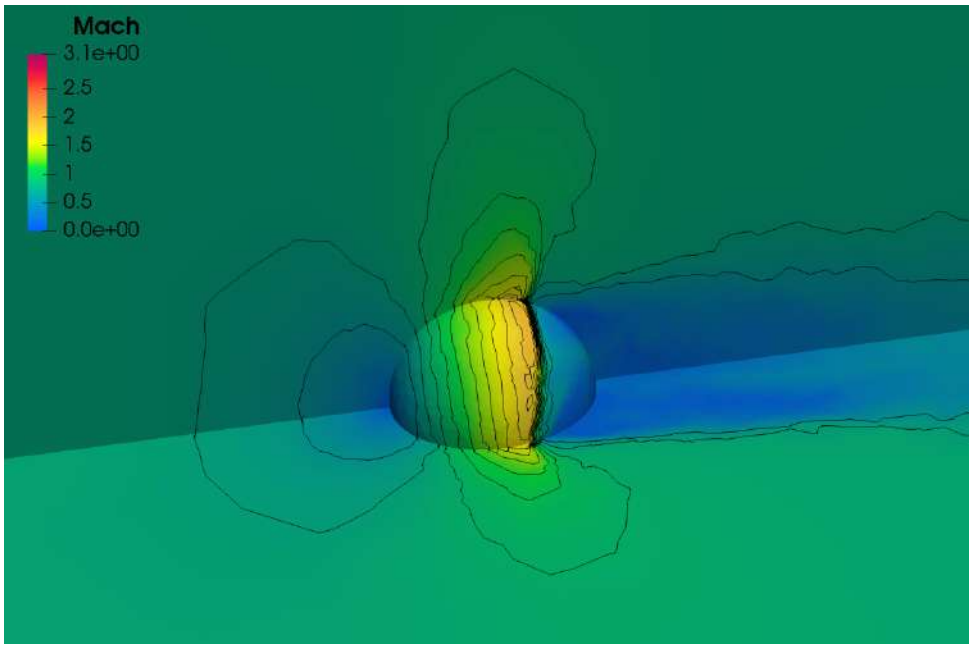


Figure 4.8: *Sphere Case A: Mach number field near the sphere, presented in a quarter cross-section, prior to the 3rd refinement cycle (computed using the mesh from the 2nd refinement cycle).*

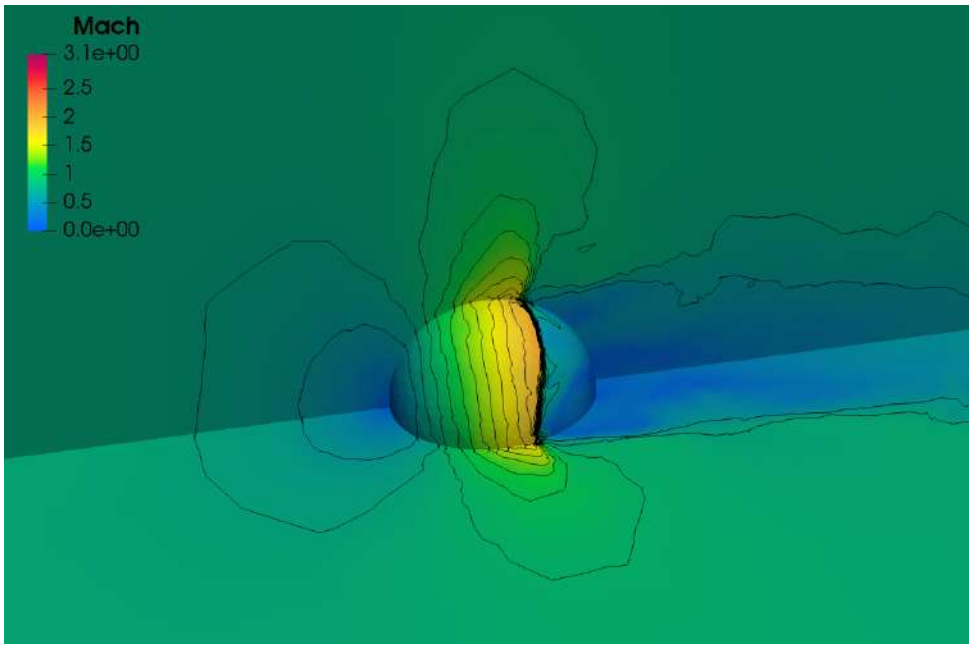


Figure 4.9: *Sphere Case A: Mach number field near the sphere, presented in a quarter cross-section, prior to the 4th refinement cycle (computed using the mesh from the 3rd refinement cycle).*

Figures 4.3- 4.6 illustrate the progression of the mesh refinement throughout the entire simulation. With each refinement cycle, the mesh becomes progressively finer in the region where the shock wave is formed, both on the surface of the sphere as well as in the surrounding field. Additionally, refinement is also observed in the region of the sphere's wake, where the flow velocity decreases and the transient phenomena appear (in a transient simulation). Initially, the first refinement cycle refines the mesh more uniformly over the surface of the sphere. It is only after the 2nd cycle that the refinement becomes more concentrated on the area of the shock wave. The final mesh demonstrates a substantial concentration of elements at the shock wave's position as well as at the downstream of the sphere.

The effect of the refinement in those regions can be seen in the evolution of the Mach number fields shown in figures 4.7- 4.9. As the mesh is refined, the shock wave's thickness is reduced and its resolution is increased. By the final refinement cycle, the shock wave's thickness is significantly reduced, approaching a straight line.

Comparing the final Mach number field (figure 4.9) with the one in [5], several similarities can be observed, however, the length of the shock wave is not identical. The primary reason for this occurs is because the flow field presented in this study represents a steady-state mean flow, whereas the reference study considers a fully transient flow. As a result, the shock wave's location may vary over time in the transient case, whereas in this study, it is depicted in an averaged time-independent setup.

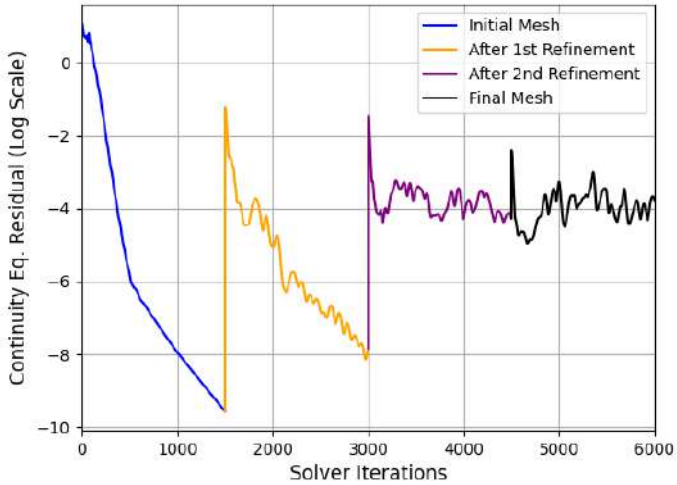


Figure 4.10: *Sphere Case A: Convergence of the residual of the continuity equation, in logarithmic scale, across the entire simulation. After the 2nd refinement cycle, the residuals exhibit significant oscillations.*

Figure 4.10 shows the convergence of the residuals of the continuity equation throughout the simulation. Since the inviscid flow around a sphere is inherently a transient problem, a more dissipative discretization scheme is employed in order to promote convergence on the initial coarse meshes. While the initial meshes exhibit a good convergence, as the mesh resolution increases, the flow unsteadiness becomes more apparent. As a result of that, the rate of residual reduction stagnates, and an oscillatory behavior can be observed (starting from the 2nd refined mesh).

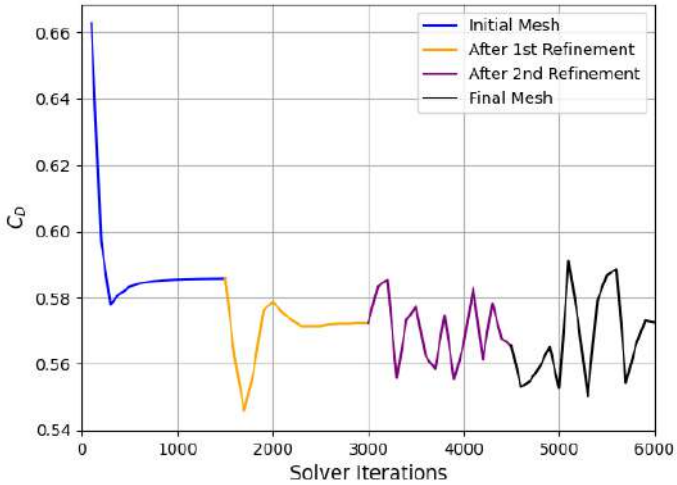


Figure 4.11: *Sphere Case A: Convergence of the drag coefficient, C_D, across the entire simulation.*

The convergence of the drag coefficient, C_D, of the sphere is presented in figure 4.11. Similar to the residuals (figure 4.10), it exhibits an oscillatory behavior. On the initial mesh, the low mesh resolution makes the flow field appear as a steady mean flow, leading to the drag coefficient seemingly converging to a fixed value. As the mesh resolution increases and the unsteadiness of the flow is revealed, oscillations also appear to C_D. The mean value around which C_D oscillates is approximately 0.57, which is close the time-averaged value of 0.575, calculated using the data in the reference study [5].

Table 4.1 shows the number of nodes and tetrahedra of each one of the meshes shown in figures 4.3- 4.6.

Mesh	**Number of nodes**	**Number of tetrahedra**
Initial Mesh	3973	21129
After the 1st Refinement	10727	58125
After the 2nd Refinement	25414	140551
Final Mesh	70567	398723

Table 4.1: *Sphere Case A: Number of nodes and tetrahedra of each mesh of the simulation.*

4.2 Case B: Freestream Mach 0.95

4.2.1 Case Description

The second variant of the case considers a freestream Mach number of 0.95, with an inviscid flow of a compressible fluid, similar to the one in [5]. The boundary conditions are set as described in Case A.

4.2.2 Results

For this simulation, the maximum number of solver iterations is also set to 6000, with three refinement cycles in total. Each refinement cycle is performed after every 1500 iterations of the solver. Similar to Case A, the problem is expected to exhibit a transient behavior as the mesh resolution increases. Therefore, a discretization scheme with increased dissipation is employed to ensure convergence on the initial coarse meshes.

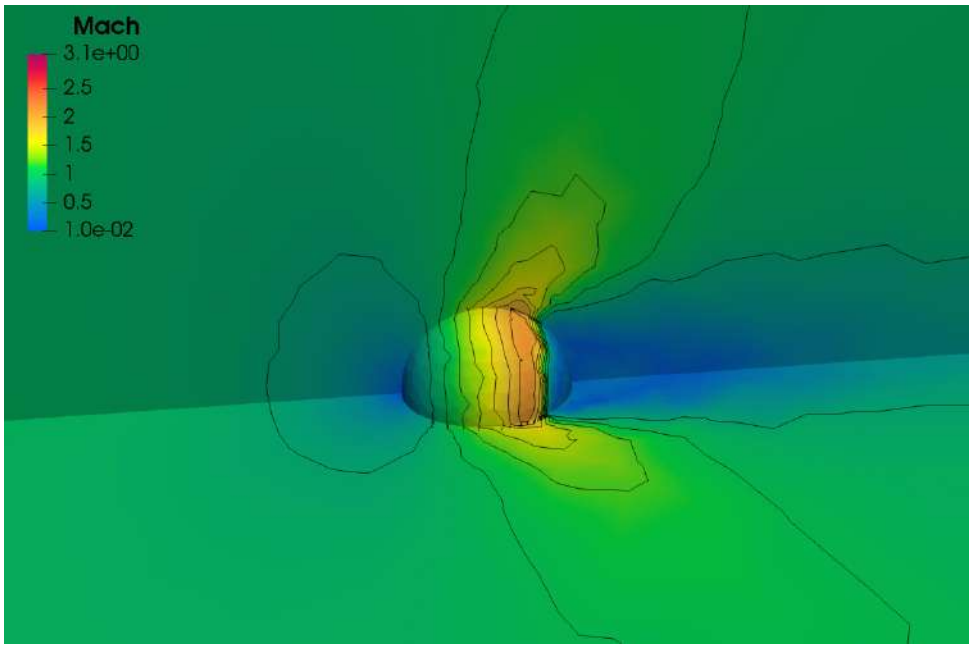


Figure 4.12: *Sphere Case B: Mach Number field computed on the initial mesh near the sphere, presented in a quarter cross-section, after the first 1500 solver iterations. A shock wave, with increased thickness due to low initial mesh resolution can be observed on the surface of the sphere.*

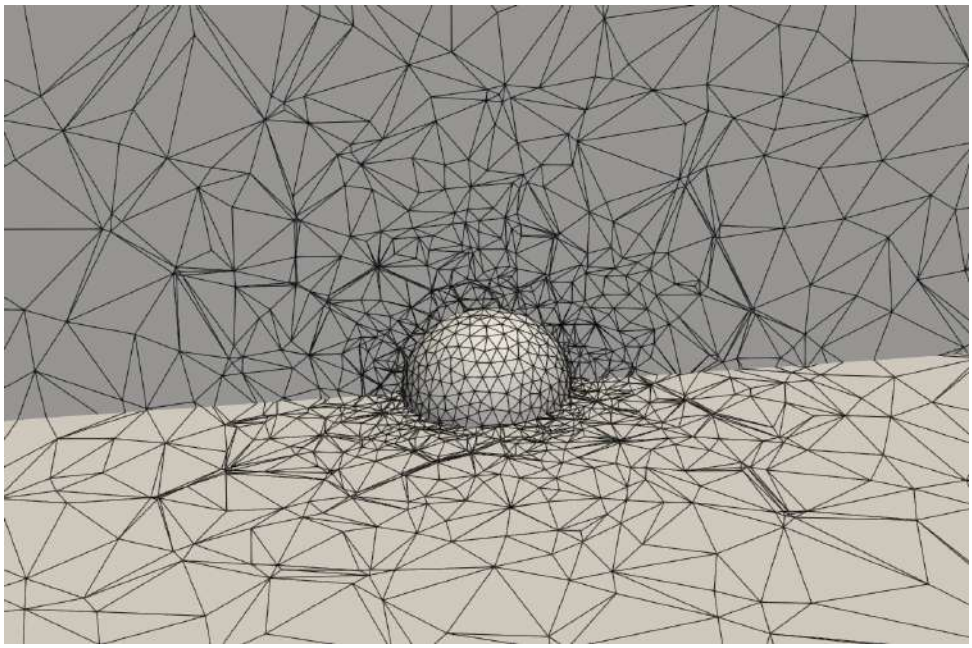


Figure 4.13: *Sphere Case B: View of the initial mesh of the channel near the sphere, presented in a quarter cross-section.*

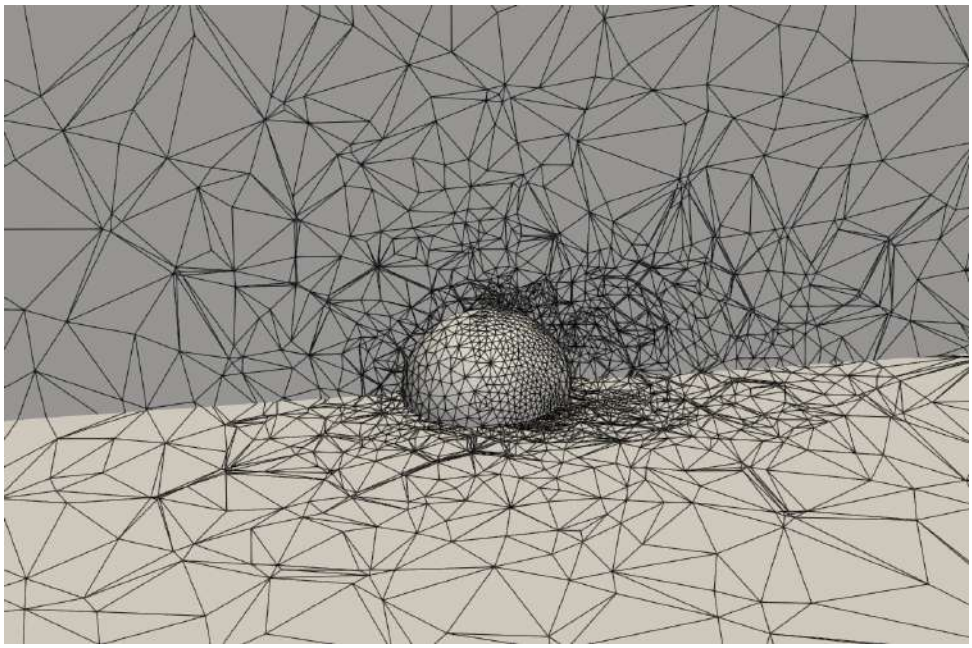


Figure 4.14: *Sphere Case B: View of the mesh near the sphere, presented in a quarter cross-section, after the 1st refinement cycle.*

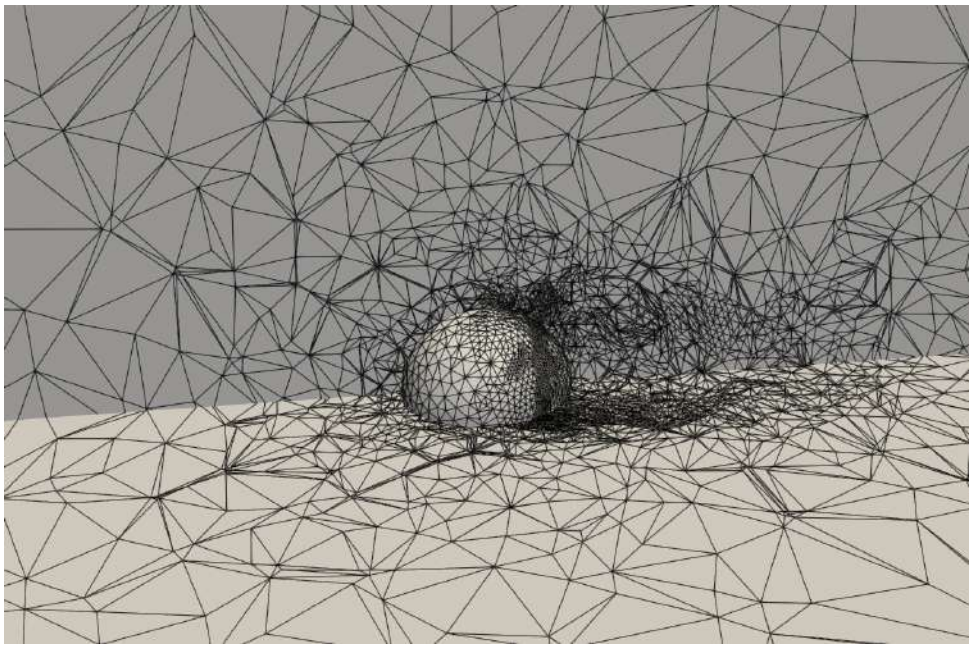


Figure 4.15: *Sphere Case B: View of the mesh near the sphere, presented in a quarter cross-section, after the 2nd refinement cycle.*

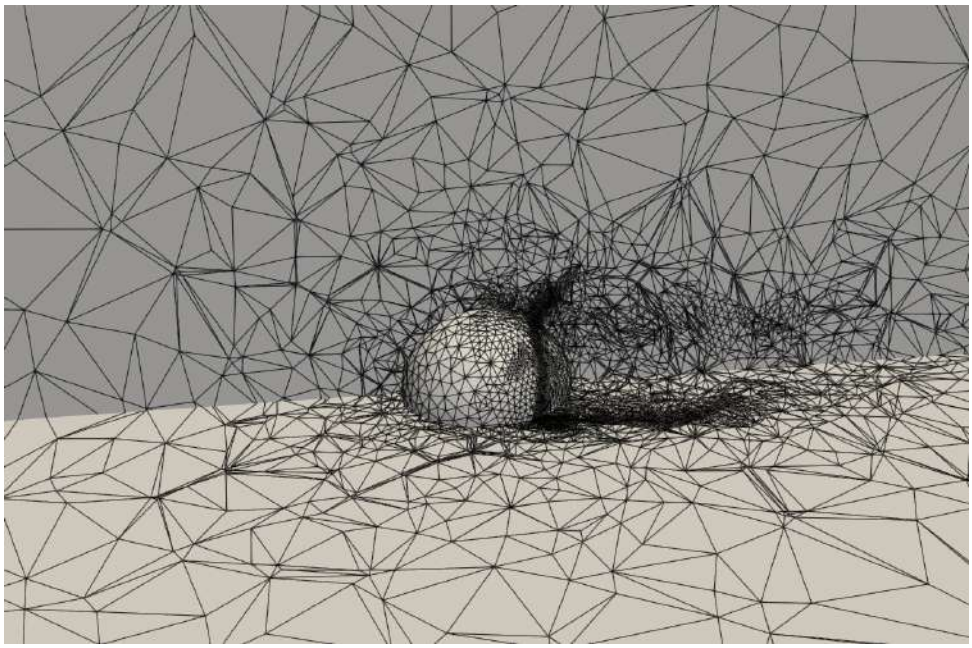


Figure 4.16: *Sphere Case B: View of the mesh near the sphere, presented in a quarter cross-section, after the 3rd refinement cycle. The mesh consists of 63276 nodes and 358934 tetrahedra.*

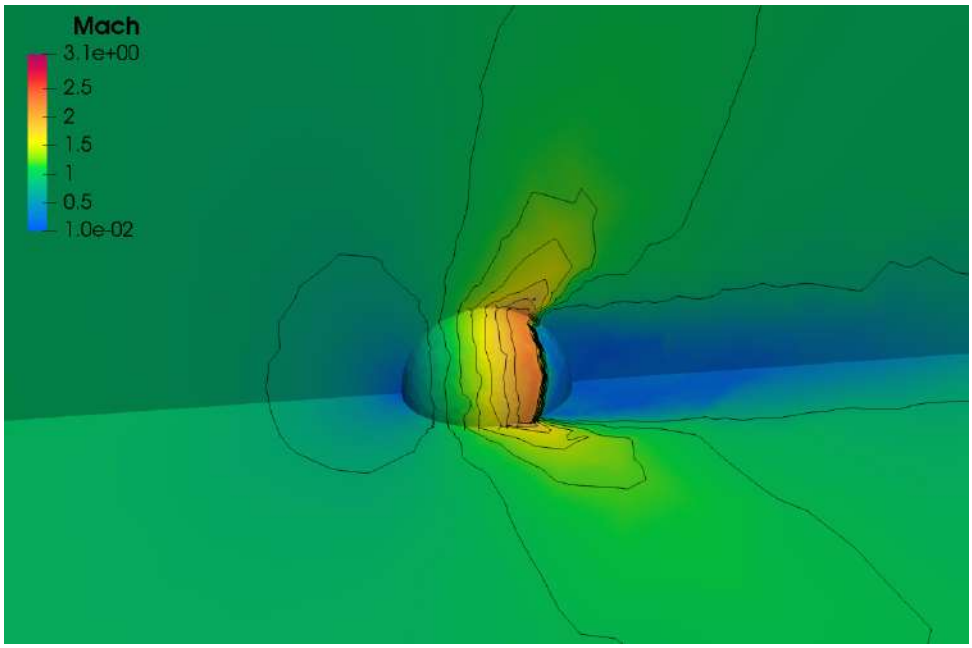


Figure 4.17: *Sphere Case B: Mach number field near the sphere, presented in a quarter cross-section, prior to the 2nd refinement cycle (computed using the mesh from the 1st refinement cycle).*

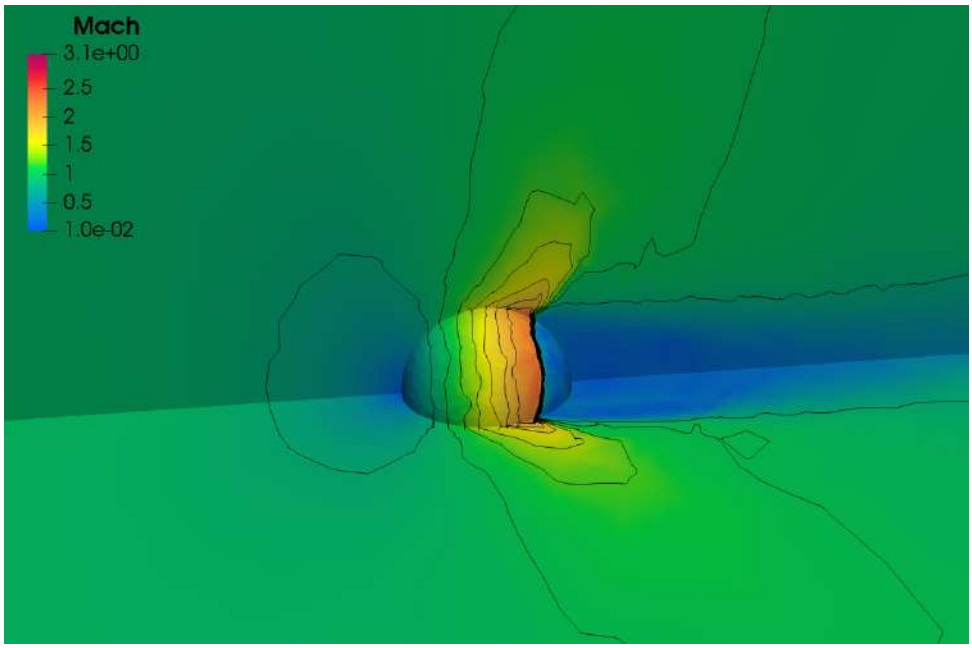


Figure 4.18: *Sphere Case B: Mach number field near the sphere, presented in a quarter cross-section, prior to the 3rd refinement cycle (computed using the mesh from the 2nd refinement cycle).*

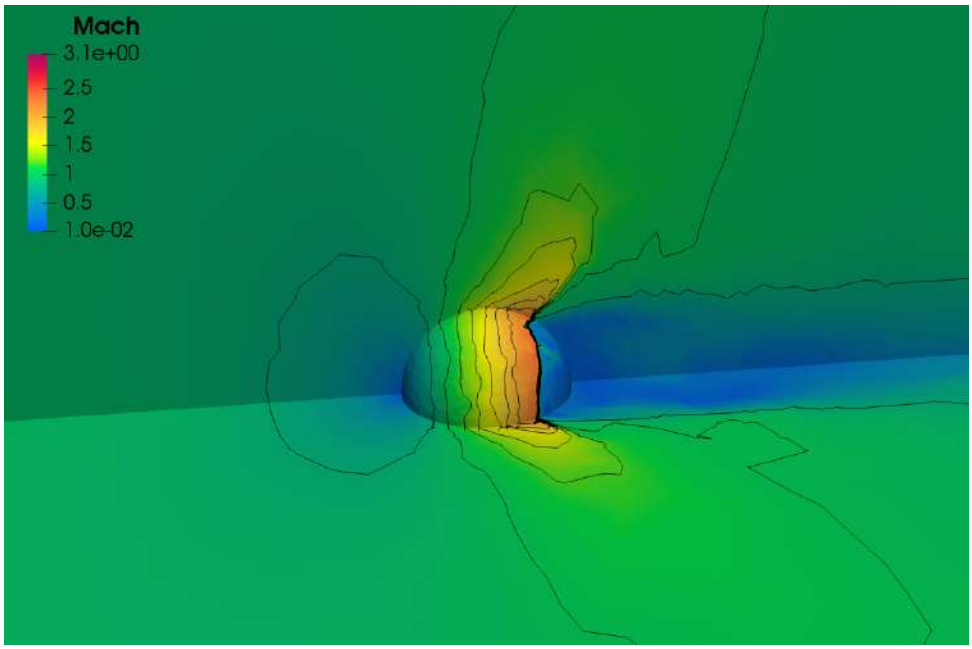


Figure 4.19: *Sphere Case B: Mach Number field near the sphere, presented in a quarter cross-section, prior to the 4th refinement cycle (computed using the mesh from the 3rd refinement cycle).*

Figures 4.13- 4.16 show the progression of the mesh refinement throughout the entire

simulation. With each refinement cycle, the mesh becomes progressively denser in the region on the surface of the sphere, where the shock wave is formed, as well as in the region of the sphere's wake. The final mesh, demonstrates a substantial concentration of elements at the shock wave's position as well as at the downstream of the sphere.

The effect of the refinement in those regions is evident in the evolution of the Mach number fields (figures 4.17- 4.19). As the mesh is refined in those regions, the shock wave's thickness is reduced and its resolution is increased. However, comparing the fields produced using the mesh from the second and third refinement cycle (figures 4.8 and 4.19, respectively), the shock wave can be observed slightly offset in the area of the sphere. This results from the flow unsteadiness becoming more apparent as the mesh resolution is increased. However, even in this context, the mesh refinement is still able to capture a thinner, higher resolution shock wave than the initial mesh.

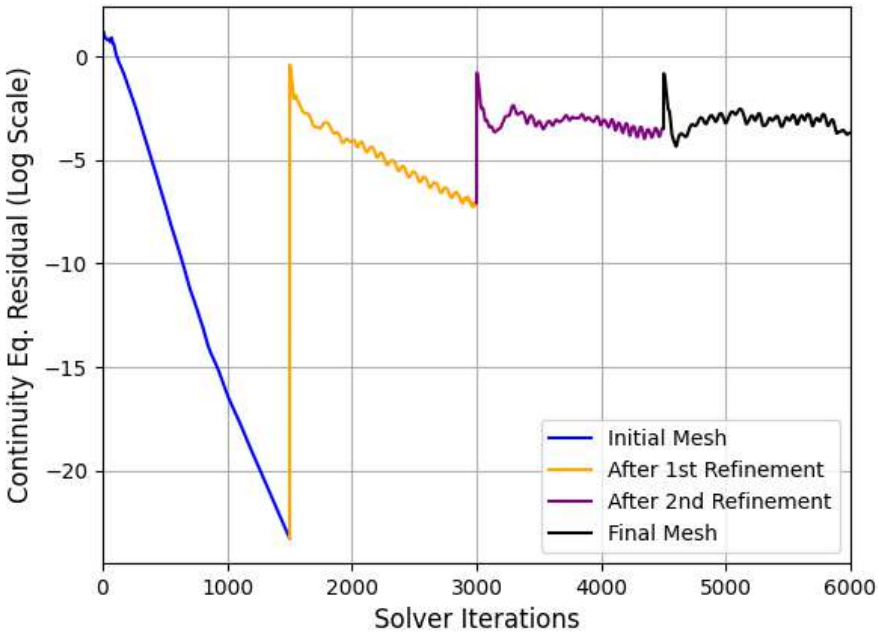


Figure 4.20: *Sphere Case B: Convergence of the residual of the continuity equation, in logarithmic scale, across the entire simulation.*

The convergence of the residuals of the continuity equation, shown in figure 4.20, follows a similar trend with the ones of Case A (figure 4.10). After the second refinement cycle, the flow unsteadiness makes the residuals oscillate.

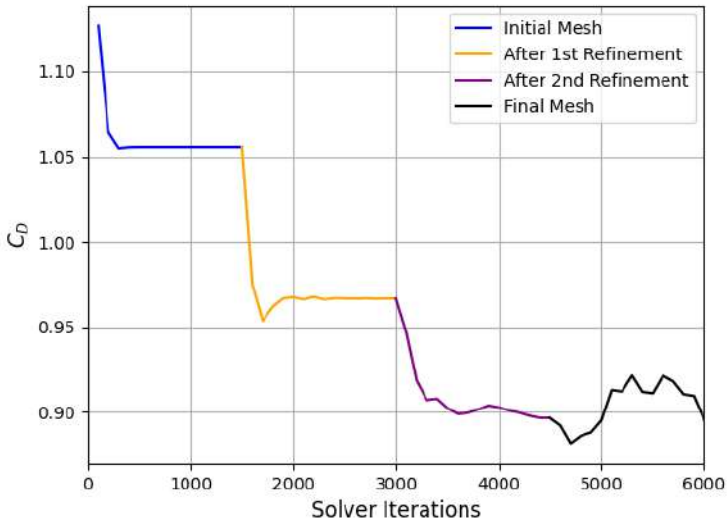


Figure 4.21: *Sphere Case B: Convergence of the drag coefficient (C_D) across the entire simulation.*

Consequently, the drag coefficient, C_D (figure 4.21) follows a similar trend. The first two meshes appear to converge to a value and only after the third refinement cycle its value starts oscillating. The mean value around which C_D oscillates is approximately 0.9, which is close the time-averaged value of 0.945, calculated using the data in the reference study [5].

Table 4.2 contains the number of nodes and tetrahedra of each one of the meshes shown in figures 4.13- 4.16.

Mesh	**Number of nodes**	**Number of tetrahedra**
Initial Mesh	3973	21129
After the 1st Refinement	9054	49139
After the 2nd Refinement	23232	129814
Final Mesh	63276	358934

Table 4.2: *Sphere Case B: Number of nodes and tetrahedra of each mesh of the simulation.*

Chapter 5

Conclusions and Recommendations for Future Work

5.1 Summary

The aim of this diploma thesis was to implement and showcase a feature-based h-refinement algorithm on tetra meshes, and then apply it to different problems. The theoretical foundations of refinement on tetrahedral meshes were thoroughly detailed, covering all aspects of the method. A C++ program with its own dedicated data structure, was developed, following all the principles of the algorithm described. This program is set to run alongside PUMA [2]. The algorithm's performance and effectiveness was then tested on multiple problems. The first case involved an inviscid flow of a compressible fluid inside a channel with a bump, with two variants being explored: one featuring a symmetrical bump and one with an asymmetrical configuration, featuring a peak on the surface of the bump. The second case was added to test the algorithm's effect on a truly 3D flow field. The next set of cases involved analyzing the flow around a sphere, at two different freestream Mach numbers. Despite the inherently unsteadiness of the flow, the studies were conducted using a mean-flow equation solver. The goal was to demonstrate the impact of the mesh refinement algorithm on the resulting flow field, as well as its ability to capture the flow unsteadiness, starting with a very coarse initial mesh. In all applications, the quality of the refined meshes is evaluated based on the resulting flow fields, as well as results from similar case studies.

5.2 Conclusions

After completing the development of the software during the course of this thesis, along with all the studies that were conducted, the following conclusions can be made:

- Feature-based mesh refinement is an efficient method that does not add any computational load to the overall CFD simulation. In fact it optimizes the mesh as the simulation is progressing, providing a quick and efficient method of constructing an ideal mesh for the problem, reducing the dependency on the initial mesh quality.
- The refinement of a mesh significantly improves the resolution of critical flow features. In this diploma thesis, mainly shock wave dominated problems were studied. In these problems, as the mesh is refined, the discontinuities become sharper and are captured more accurately. At the same time, the number of elements in the mesh increases, only in those regions, thus not unnecessarily overloading the simulation.
- The quality of the initial mesh does not significantly impact the progression of the refinement process. Even when starting with a very coarse mesh, such as the one in Chapter 4, the refinement is still able to locate the critical areas and increase their resolution, provided that the correct sensor is selected.
- When starting from a very coarse initial mesh, multiple refinement cycles are necessary to accurately capture the flow phenomena. As observed in all the cases presented in this thesis, the first refinement cycle generally results in a broad refinement near key flow areas, improving the overall resolution. It is only after the second cycle, when the mesh already has a higher resolution near those regions that the refinement becomes more localized, effectively concentrating the new elements at the precise location of the physical phenomenon.
- In transient problems, such as the one studied in Chapter 4, mesh refinement, if not employed during the entire transient simulation, could serve as a useful tool for the preliminary study of a problem. In the sphere cases, the final refined mesh could be used as a stating mesh in order to initiate the transient simulation, or give a view of areas were the mesh needs to be more dense.

5.3 Future Work Proposals

The methods and tools developed as part of this diploma thesis offer significant potential for further development. Some recommendations for future work are:

- Expansion on the developed software to improve the interpolation of boundary

nodes. The inclusion of a generalized numerical interpolation method (such as splines) in order to enable the handling of arbitrary geometries, beyond those defined by analytical expressions.

- Extending the mesh refinement principles to hybrid meshes. This work can focus on the investigation of the various subdivision cases for hexahedrons as well as the development of a software capable of performing h-refinement efficiently, based on these principles. The theory and software have already been developed by the PCOpt Unit for 2D meshes. However the 3D context presents additional challenges such as the introduction of extra element types (prisms and pyramids) and thus a great increase in distinct subdivision cases that must be addressed, as discussed in [3].
- The inclusion of coarsening in the refinement algorithm. This can also be integrated with the previous recommendation regarding hybrid meshes in order to develop a generalized, all-purpose mesh refinement software, capable of handling any initial mesh whether it is structured, unstructured or hybrid (provided the solver is designed for unstructured).
- Employing parallelization strategies for mesh refinement. Given that modern CFD heavily relies on these techniques, mesh refinement could also benefit from them, especially in large scaled projects. Future work could focus on evaluating its computational advantages, followed by the development of such a software designed to run in parallel architectures.

Bibliography

[1] Paraview, http://paraview.org/

[2] Asouti, V., Trompoukis, X., Kampolis, I., Giannakoglou, K.: Unsteady CFD computations using vertex–centered finite volumes for unstructured grids on Graphics Processing Units. International Journal for Numerical Methods in Fluids **67**(2), 232–246 (May 2011)

[3] Biswas, R., Strawn, R.C.: Tetrahedral and hexahedral mesh adaptation for cfd problems. Applied Numerical Mathematics **26**(1), 135–151 (1998). https://doi.org/https://doi.org/10.1016/S0168-9274(97)00092-5, https://www.sciencedirect.com/science/article/pii/S0168927497000925

[4] Gerhart, P.M., Gerhart, A.L., Hochstein, J.I., Munson, B.R., Young, D.F., Okiishi, T.H.: Munson, young, and Okiishi's fundamentals of Fluid Mechanics. Wiley (2016)

[5] Karanjkar, P.V.: Inviscid Transonic Flow Around a Sphere. Diploma Thesis, University of Florida (2008)

[6] Lynn, J., van Leer, B., Lee, D.: Multigrid solution of the euler equations with local preconditioning (01 1997). https://doi.org/10.1007/BFb0107097

[7] Löhner, R.: An adaptive finite element scheme for transient problems in cfd. Computer Methods in Applied Mechanics and Engineering **61**(3), 323–338 (1987). https://doi.org/https://doi.org/10.1016/0045-7825(87)90098-3, https://www.sciencedirect.com/science/article/pii/0045782587900983

[8] Löhner, R., Baum, J.D.: Adaptive h-refinement on 3d unstructured grids for transient problems. International Journal for Numerical Methods in Fluids **14**(12), 1407–1419. https://doi.org/https://doi.org/10.1002/fld.1650141204, https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.1650141204

[9] Park, M., Krakos, J., Michal, T., Loseille, A., Alonso, J.: Unstructured grid adaptation: Status, potential impacts, and recommended investments toward cfd vision 2030 (06 2016). https://doi.org/10.2514/6.2016-3323

[10] Parthasarathy, V., Graichen, C., Hathaway, A.: A comparison of tetrahedron quality measures. Finite Elements in Analysis and De-

sign **15**(3), 255–261 (1994). https://doi.org/https://doi.org/10.1016/0168-874X(94)90033-7, https://www.sciencedirect.com/science/article/pii/0168874X94900337

[11] Pengcheng, C., Yongqian, Z., Peimin, X., Bin, L., Mingsheng, M., Guiyu, Z.: The comparison of adjoint-based grid adaptation and feature-based grid adaptation method. Journal of Physics: Conference Series **2280**, 012003 (06 2022). https://doi.org/10.1088/1742-6596/2280/1/012003

[12] Speares, W., Berzins, M.: a 3d unstructured mesh adaptation algorithm for time-dependent shock-dominated problems. International Journal for Numerical Methods in Fluids **25**, 81–104 (07 1997). https://doi.org/10.1002/(SICI)1097-0363(19970715)25:13.3.CO;2-S

[13] Venditti, D.A., Darmofal, D.L.: Grid adaptation for functional outputs: Application to two-dimensional inviscid flows. Journal of Computational Physics **176**(1), 40–69 (2002). https://doi.org/https://doi.org/10.1006/jcph.2001.6967, https://www.sciencedirect.com/science/article/pii/S0021999101969670

[14] Wackers, J., Visonneau, M., Ali, Z.: Automatic grid adaptation for unstructured finite volumes. International Journal of Engineering Systems Modelling and Simulation **2** (01 2010). https://doi.org/10.1504/IJESMS.2010.031866

[15] Γ. Μπεργελές: Υπολογιστική Ρευστομηχανική. Εκδόσεις Συμεών (2012)

[16] Κ. Χ. Γιαννάκογλου: Γένεση και Προσαρμογή Αριθμητικών Πλεγμάτων. Διδακτικό Σύγγραμμα, Διατμηματικό Πρόγραμμα Μεταπτυχιακών Σπουδών (1999)

[17] Κ. Χ. Γιαννάκογλου: Συνεκτικές Ροές στις Στροβιλομηχανές. Πανεπιστημιακές Παραδόσεις, Πανεπιστημιακές Εκδόσεις Ε.Μ.Π. (2004)

[18] Κ.Χ. Γιαννάκογλου: Μέθοδοι Βελτιστοποίησης στην Αεροδυναμική (2006)

[19] Κ.Χ. Γιαννάκογλου, Ι. Αναγνωστόπουλος, Γ. Μπεργελές: Αριθμητική Ανάλυση για Μηχανικούς (2003)

[20] Π. Λιακόπουλος: Γένεση μη-δομημένων πλεγμάτων και διαχείρισή τους σε μεθόδους ανάλυσης και βελτιστοποίησης συνιστωσών στροβιλομηχανών και εφαρμογές, αξιοποιώντας τεχνολογίες πλέγματος (Grid computing). Ph.D. thesis, Εργαστήριο Θερμικών Στροβιλομηχανών, Ε.Μ.Π., Αθήνα (2008)

[21] Φ. Ι. Χριστακόπουλος: Αλγόριθμος προσαρμογής διδιάστατων υβριδικών πλεγμάτων στην υπό εξέλιξη λύση ενός πεδίου ροής και πιστοποίηση. Διπλωματική Εργασία, Εργαστήριο Θερμικών Στροβιλομηχανών, Ε.Μ.Π. (2007)

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Μηχανολόγων Μηχανικών
Τομέας Ρευστών
Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής
& Βελτιστοποίησης

η-Εμπλουτισμός Πλεγμάτων Τετραεδρικών Στοιχείων. Προγραμματισμός λογισμικού και εφαρμογές στην Υπολογιστική Ρευστοδυναμική

Διπλωματική Εργασία - Εκτενής Περίληψη στα Ελληνικά

Δημήτριος Χονδρουδάκης

Επιβλέπων: Κυριάκος Χ. Γιαννάκογλου, Καθηγητής ΕΜΠ

Αθήνα, 2025

Εισαγωγή

Τις τελευταίες δεκαετίες η Υπολογιστική Ρευστοδυναμική (ΥΡΔ) έχει γνωρίσει σημαντική ανάπτυξη τόσο σε ερευνητικό επίπεδο όσο και σε βιομηχανικές εφαρμογές λόγω της ραγδαίας τεχνολογικής ανάπτυξης. Ένα πλέγμα αποτελεί τη βάση της ΥΡΔ και επηρεάζει την ακρίβεια και αξιοπιστία των αποτελεσμάτων. Επομένως είναι απαραίτητη η κατασκευή ενός πλέγματος ικανού να παράγει μια σωστή λύση εξασφαλίζοντας ταυτόχρονα καλή σύγκλιση των εξισώσεων ροής. Στην πράξη ωστόσο, είναι αρκετά δύσκολο να δημιουργηθεί ένα πλέγμα με τη σωστή πυκνότητα σε ακρίβεια. Μια λύση που προτείνεται σε αυτό το πρόβλημα είναι η προσαρμογή πλέγματος στην υπό εξέλιξη λύση, μέσω διαδικασίας εμπλουτισμού.

Υπάρχουν δύο βασικές κατηγορίες πλεγμάτων, τα δομημένα και τα μη-δομημένα πλέγματα. Τα δομημένα πλέγματα χαρακτηρίζονται από μια απλή τοπολογική δομή ενώ τα μη δομημένα χρησιμοποιούν μια περίπλοκη τοπολογική δομή για να περιγραφούν. Τα δομημένα πλέγματα αποτελούνται από τετράπλευρα (σε 2Δ) ή εξάεδρα (σε 3Δ), ενώ τα μη-δομημένα από τρίγωνα (σε 2Δ) και τετράεδρα (σε 3Δ). Σε περίπτωση που ένα πλέγμα είναι δομημένο σε μια περιοχή και ταυτόχρονα μη-δομημένο σε κάποια άλλη, τότε πρόκειται για ένα υβριδικό πλέγμα.

Υπάρχουν δύο βασικές κατηγορίες προσαρμογής πλεγμάτων: η προσαρμογή βασισμένη σε χαρακτηριστικά της ροής (Feature-based Mesh Adaptation) και αυτή που βασίζεται σε συζυγείς μεθόδους (Adjoint-based Mesh Adaptation). Η πρώτη κατηγορία χρησιμοποιεί μια φυσική ποσότητα της ροής σε συνδυασμό με μια μαθηματική συνάρτηση για να αξιολογήσει ποιες περιοχές του πλέγματος χρειάζονται τροποποίηση ενώ η δεύτερη βασίζεται στη διατύπωση και επίλυση ενός προβλήματος βελτιστοποίησης με χρήση των συζυγών εξισώσεων. Η προσαρμογή πλέγματος στην υπό εξέλιξη λύση περιλαμβάνει τον εμπλουτισμό και τον απεμπλουτισμό. Ο εμπλουτισμός συνίσταται στην προσθήκη νέων στοιχείων ενώ ο απεμπλουτισμός στην αφαίρεση παλαιότερων. Στόχος είναι η δημιουργία ενός πλέγματος με κατάλληλη πυκνότητα μόνο σε περιοχές υψηλού ενδιαφέροντος. Ο εμπλουτισμός εφαρμόζεται κυρίως σε μη-δομημένα πλέγματα λόγω της δυνατότητας τοπικής επέμβασης μόνο στις περιοχές υψηλού ενδιαφέροντος.

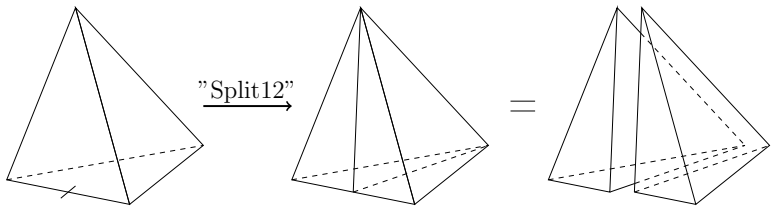
Η διπλωματική εργασία επικεντρώνεται στη διατύπωση και τον προγραμματισμό ενός αλγορίθμου για τον h-εμπλουτισμό πλεγμάτων τετραεδρικών στοιχείων. Ο αλγόριθμος αυτός γράφτηκε στην γλώσσα προγραμματισμού C++ και σχεδιάστηκε για να συνεργάζεται εξωτερικά με τον επιλύτη ροής PUMA της ΜΠΥΡΔ&Β του ΕΜΠ. Για την οπτικοποίηση των αποτελεσμάτων χρησιμοποιήθηκε το λογισμικό ανοιχτού κώδικα ParaView. Η εργασία δεν εξετάζει διαδικασίες απεμπλουτισμού.

Εμπλουτισμός Πλεγμάτων Τετραεδρικών Στοιχείων

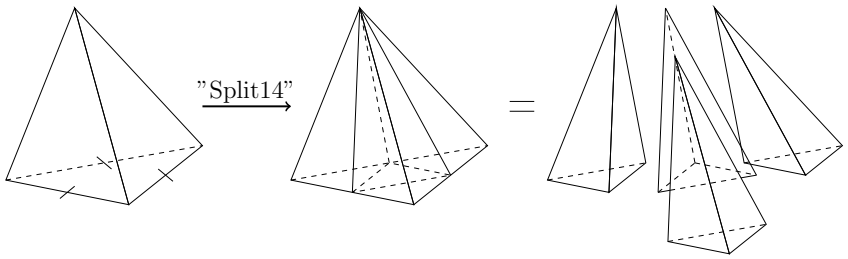
Στο παρόν κεφάλαιο διατυπώνονται οι βασικές αρχές που αφορούν την προσαρμογή πλεγμάτων τετραεδρικών στοιχείων.

Ο εμπλουτισμός πλεγμάτων λαμβάνει χώρα ανα συγκεκριμένο πλήθος επαναληπτικών βημάτων του επιλύτη της ροής. Τα στοιχεία του αρχικού πλέγματος θα αναφέρονται ως F1. Κάθε αλγόριθμος προσαρμογής υπολογιστικών πλεγμάτων υπακούει σε συγκεκριμένους κανόνες. Οι κανόνες που υιοθετούνται στην παρούσα διπλωματική εργασία είναι οι εξής:

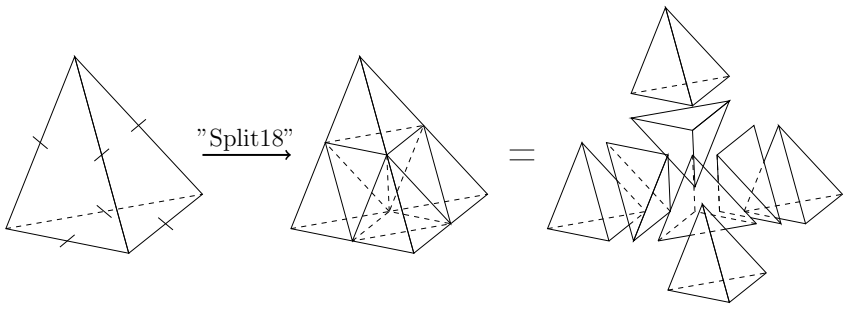
- **Κανόνας 1:** Δεν επιτρέπεται η ¨συγκόλληση' στοιχείων του αρχικού πλέγματος F1 προς σχηματισμό νέων, μεγαλύτερων.
- **Κανόνας 2:** Ένα τετράεδρο επιτρέπεται να διασπαστεί με τέσσερις μόνο τρόπους οι οποίοι παρουσιάζονται στα σχήματα 1- 4, μαζί με τις κωδικές τους ονομασίες.



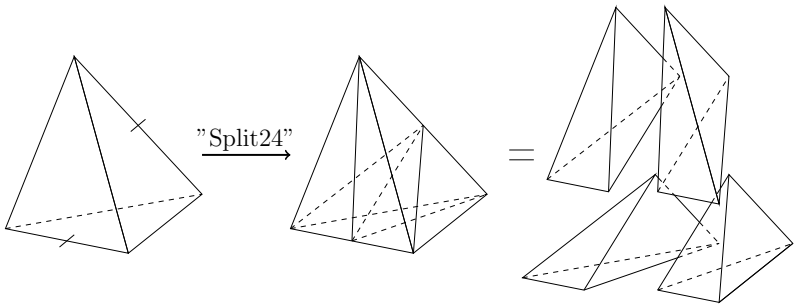
Σχήμα 1: *Πρώτη επιτρεπόμενη μορφή διάσπασης ενός τετραέδρου σε 2 τετράεδρα, με ονομασία "Split12".*



Σχήμα 2: *Δεύτερη επιτρεπόμενη μορφή διάσπασης ενός τετραέδρου σε 4 τετράεδρα, με ονομασία "Split14".*

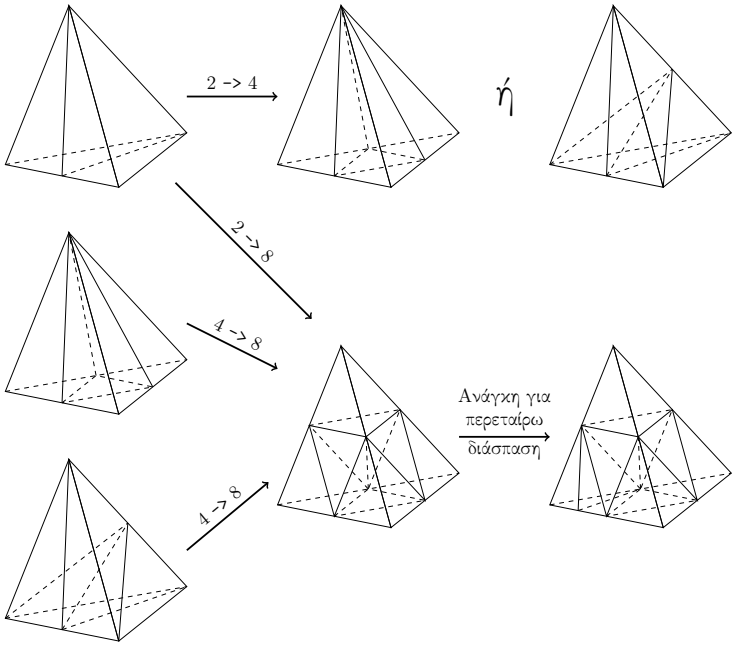


Σχήμα 3: *Τρίτη επιτρεπόμενη μορφή διάσπασης ενός τετραέδρου σε 8 τετράεδρα, με ονομασία "Split18".*



Σχήμα 4: *Τέταρτη επιτρεπόμενη μορφή διάσπασης ενός τετραέδρου σε 4 τετράεδρα, με ονομασία "Split24".*

- **Κανόνας 3:** Ένα τετράεδρο που έχει προέλθει από τη διάσπαση τετραέδρου προηγούμενης γενιάς σε δύο ή τέσσερα δεν επιτρέπεται να διασπαστεί περαιτέρω. Πρέπει να προηγηθεί επανενεργοποίηση του μητρικού και ύστερα να ακολουθήσει η διάσπασή του. Η διαδικασία αυτή, για αρκετές περιπτώσεις, παρουσιάζεται στο σχήμα 5.



Σχήμα 5: *Σχηματική απεικόνιση του Κανόνα 3.*

Ο λόγος για τον οποίο επιβάλλεται αυτή η διαδικασία είναι η αποφυγή δημιουργίας στοιχείων με υψηλό λόγο επιμήκους, τα οποία ενδέχεται να προκαλέσουν αριθμητικά σφάλματα στην επίλυση.

- **Κανόνας 4:** Ένα τετράεδρο που έχει προέλθει από τη διάσπαση του μητρικού του σε 8 τετράεδρα ("Split18") επιτρέπεται να ξαναδιασπαστεί χωρίς κανέναν περιορισμό.

- **Κανόνας 5:** Τα τετράεδρα δεν επιτρέπεται να διασπώνται επ΄ άπειρο. Με αυτόν τον τρόπο αποφεύγεται η δημιουργία πολύ μικρών στοιχείων τα οποία μπορεί να προκαλέσουν προβλήματα σύγκλισης και αναίτια αύξηση της απαιτούμενης μνήμης.

- **Κανόνας 6:** Όταν ένα τετράεδρο διασπάται, ενδέχεται να παρασύρει και ορισμένα από τα γειτονικά του προκειμένου να διατηρηθεί η φυσιογνωμία του τυπικού μη-δομημένου πλέγματος.

- **Κανόνας 7:** Στα περιοδικά πλέγματα, κάθε τροποποίηση πάνω στο περιοδικό όριο πρέπει να συνοδεύεται από αντίστοιχη τροποποίηση στην περιοχή που αντιστοιχεί σε αυτό μέσω της περιοδικότητας.

Διαχείριση Δεδομένων κατά την Προσαρμογή

Λόγω της φύσης των μη-δομημένων πλεγμάτων η διαχείρισή τους και ακόμα περισσότερο η προσαρμογή τους είναι μια περίπλοκη διεργασία. Επομένως η επιλογή μιας βάσης δεδομένων που να παρέχει όλες τις απαραίτητες πληροφορίες χωρίς να συνοδεύεται από υπερβολική και άσκοπη χρήση μνήμης αποτελεί μια σημαντική απόφαση. Στο πλαίσιο ενός αλγορίθμου προσαρμογής πλεγμάτων, τα στοιχεία του (τετράεδρα και ακμές) διαχωρίζονται σε τρείς κατηγορίες:

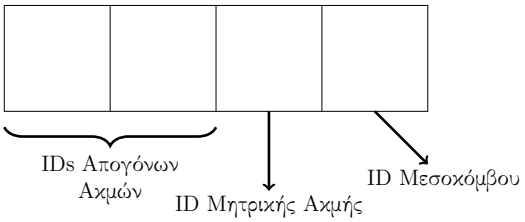
- **Ενεργά:** είναι τα στοιχεία τελευταίας γενιάς τα οποία είναι υποψήφια για εμπλουτισμό.

- **Ανενεργά:** είναι τα στοιχεία προηγούμενης γενιάς που έχουν αντικατασταθεί από νεότερα λόγω της διάσπασής τους σε κάποιον κύκλο προσαρμογής. Οι πληροφορίες τους ωστόσο παραμένουν αποθηκευμένες σε περίπτωση που ξαναγίνουν ενεργά λόγω του Κανόνα 3.

- **Εξουδετερωμένα:** είναι τα στοιχεία που έχουν μαρκαριστεί για διαγραφή λόγω εφαρμογής του Κανόνα 3. Οι πληροφορίες τους δε χρειάζονται πλέον και διαγράφονται.

Προκειμένου να αναπτυχθεί ένας αλγόριθμος εμπλουτισμού υπολογιστικών πλεγμάτων, εκτός από τις τοπολογικές πληροφορίες που χρειάζονται για την επίλυση της ροής, εισάγεται ένας νέος τύπος πληροφοριών, οι γενεαλογικές πληροφορίες. Οι γενεαλογικές πληροφορίες περιλαμβάνουν όλες τις απαραίτητες πληροφορίες για την πραγματοποίηση της προσαρμογής, όπως συσχετίσεις στοιχείων από τον έναν κύκλο εμπλουτισμού στον επόμενο.

Στον αλγόριθμο εμπλουτισμού που αναπτύχθηκε, προτείνεται η αποθήκευση των πληροφοριών αυτών πάνω σε κάθε ακμή και τετράεδρο του πλέγματος. Συγκεκριμένα, υιοθετείται η χρήση 4 ακεραίων ανά ακμή και 6 ακεραίων ανά τετράεδρο. Τα δεδομένα αυτά, μαζί με τις πληροφορίες που αντιπροσωπεύουν, περιγράφονται σε μορφή πίνακα στα σχήματα 6- 7.



Σχήμα 6: *Γενεαλογικά δεδομένα μιας ακμής του πλέγματος σε μορφή πίνακα.*

Σχήμα 7: *Γενεαλογικά δεδομένα ενός τετράεδρου σε μορφή πίνακα..*

Κριτήρια Εμπλουτισμού

Η επιλογή των περιοχών του πλέγματος που πρέπει να τροποποιηθούν σε μια προσαρμογή γίνεται με κριτήρια που εφαρμόζονται πάνω σε ακμές. Επομένως, όσα αναφέρονται παρακάτω αντιστοιχούν σε κεντροκομβική διατύπωση των εξισώσεων ροής από το λογισμικό επίλυσης. Προκειμένου να είναι επιτυχημένη η διαδικασία εμπλουτισμού πρέπει να ληφθούν υπόψη τρείς βασικές παράμετροι:

1. Η κατάλληλη επιλογή φυσικής ποσότητας (η οποία ονομάζεται Αισθητήριο) που θα μπορεί να παρέχει επαρκείς ενδείξεις σχετικά με τις περιοχές του πεδίου στις οποίες είναι επιθυμητη μια πιο ακριβής εικόνα.

2. Η χρήση μιας κατάλληλης αναλυτικής συνάρτησης (η οποία ονομάζεται Συνάρτηση Κρίσης) η οποία χρησιμοποιώντας το επιλεγμένο αισθητήριο θα είναι ικανή να παρέχει μια αριθμητική ένδειξη της ευαισθησίας κάθε ακμής του πλέγματος.

3. Ο καθορισμός κατάλληλων ορίων για τις τιμές που παρέχει η συνάρτηση κρίσης. Οι τιμές αυτές ονομάζονται κατώφλια και χρησιμοποιούνται για να καθορίσουν ποιες ακμές θα μαρκαριστούν για εμπλουτισμό.

Συναρτήσεις Κρίσης και Κατώφλια Εμπλουτισμού

Υπάρχουν πολλών ειδών συναρτήσεις κρίσεων οι οποίες συνήθως βασίζονται στη μεταβολή του αισθητήριου κατά μήκος μιας ακμής. Ενδεικτικά παρουσιάζονται δύο εκφράσεις μέσω των εξισώσεων 1 & 2.

$$f = |\Phi_1 - \Phi_2| \tag{1}$$

$$f = \left| \frac{d\Phi}{dx} \right| \sim \left| \frac{\Phi_1 - \Phi_2}{\Delta x} \right| \tag{2}$$

όπου f είναι η συνάρτηση κρίσεως και Φ η επιλεγείσα ποσότητα αισθητήριο.

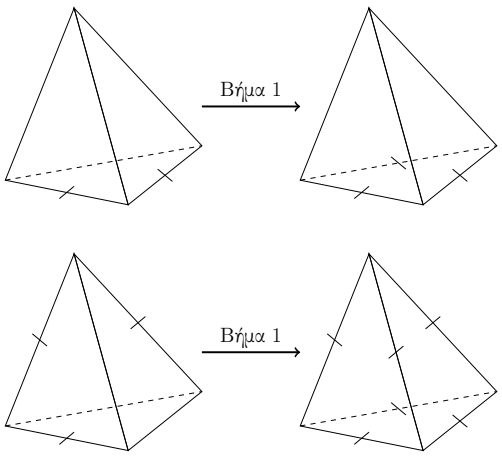
Στο πλήρες κείμενο της διπλωματικής εργασίας στην αγγλική γλώσσα παρουσιάζονται περισσότερες εκφράσεις συναρτήσεων κρίσεως μαζί με παραλλαγές τους, συνοδευόμενες από επεξήγηση των εφαρμογών τους. Εφόσον επιλεγεί η κατάλληλη ποσότητα αισθητήριο, ακολουθεί ο καθορισμός των κατωφλιών. Ακμές όπου η τιμή της συνάρτησης κρίσεως είναι μεγαλύτερη από το κατώφλι εμπλουτισμού, μαρκάρονται για εμπλουτισμό. Οι τιμές τους συνήθως βασίζονται σε στατιστικά μεγέθη της συνάρτησης κρίσεως. Η έκφραση που προτείνεται για τον καθορισμό του κατωφλιού εμπλουτισμού περιγράφεται από την εξίσωση 3.

$$f_1 = \alpha \cdot f_{mean} + \beta \cdot f_{dev} \tag{3}$$

Αλγόριθμος Εμπλουτισμού

Η διαδικασία του εμπλουτισμού ξεκινά με τη σάρωση όλων των ακμών του πλέγματος και το μαρκάρισμά τους με έναν κατάλληλο δείκτη ο οποίος υποδεικνύει ποιες ακμές πρέπει να διασπαστούν και ποιες όχι. Αφού ολοκληρωθεί αυτό το αρχικό μαρκάρισμα, ξεκινά η διαδικασία του εμπλουτισμού η οποία περιλαμβάνει τα εξής βήματα:

- **Βήμα 1:** Αφού έχει ολοκληρωθεί το αρχικό μαρκάρισμα των ακμών του πλέγματος για εμπλουτισμό, ενδέχεται να έχουν σχηματιστεί μοτίβα μαρκαρισμένων ακμών πάνω σε τετράεδρα που να μη συμβαδίζουν με μια από τις επιτρεπόμενες μορφές διάσπασης, όπως αυτές ορίζονται από τον Κανόνα 2. Έτσι ακολουθεί ένα επιπλέον μαρκάρισμα των ακμών κάθε τετράεδρου ώστε να προκύψει μια από τις επιτρεπόμενες μορφές. Ενδεικτικά, παρουσιάζονται δύο από τις πιθανές περιπτώσεις στο σχήμα 8.



Σχήμα 8: *Παράδειγμα του Βήματος 1 του αλγορίθμου εμπλουτισμού.*

Όλες οι περιπτώσεις επιπλέον μαρκαρισμάτων για εμπλουτισμό αναλύονται στο πλήρες κείμενο.

- **Βήμα 2:** Σε περίπτωση κατά την οποία ένα τετράεδρο που έχει προκύψει από διάσπαση τύπου "Split12", "Split14" ή "Split24" εντοπιστεί με σημειωμένη κάποια ακμή του για εμπλουτισμό, τότε, σύμφωνα με τον Κανόνα 3, ακολουθεί επανενεργοποίηση του μητρικού του και μαρκάρισμα των πλευρών του βάση των επιτρεπόμενων μορφών διάσπασης. Η αναλυτική διαδικασία του Βήματος 2 περιγράφεται στο πλήρες κείμενο. Είναι φανερό ότι τα βήματα 1 και 2 τροποποιούν την εικόνα του πλέγματος όσον αφορά τις μαρκαρισμένες πλευρές. Για τον λόγο αυτό, απαιτείται μια επαναληπτική μέθοδος που αποτελείται από τη συνεχή εκτέλεση των βημάτων αυτών μέχρι του σημείου όπου δεν εμφανίζονται νέα μαρκαρίσματα ακμών.

- **Βήμα 3:** Αφού έχει ολοκληρωθεί το μαρκάρισμα των ακμών του πλέγματος για εμπλουτισμό, πραγματοποιείται η διάσπασή τους. Κάθε μαρκαρισμένη ακμή διασπάται σε δύο και δημιουργείται ένας νέος κόμβος. Ο νέος κόμβος τοποθετείται στο γεωμετρικό μέσο της ακμής, εκτός εάν ανήκει σε κάποιο οριακό σημείο του χωρίου. Σε αυτήν την περίπτωση, οι συντεταγμένες του καθορίζονται με ακριβή παρεμβολή προκειμένου να διατηρηθεί η πραγματική γεωμετρία του χωρίου της ροής. Στη συνέχεια, υπολογίζονται οι μεταβλητές του πεδίου ροής στους νέους κόμβους μέσω αριθμητικής παρεμβολής.

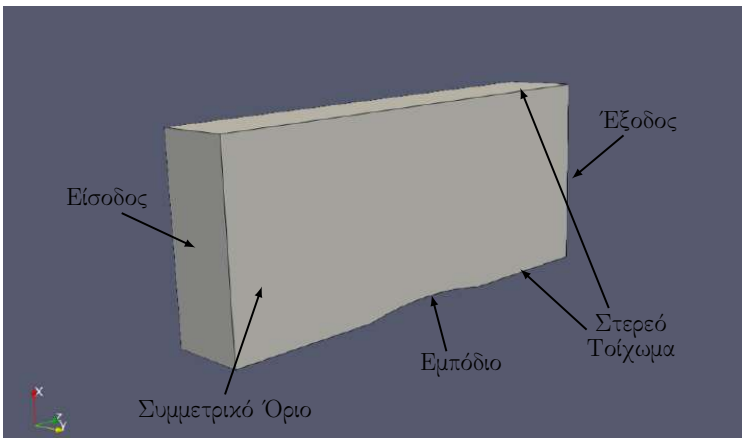
- **Βήμα 4:** Το τελικό βήμα της διαδικασίας εμπλουτισμού είναι η δημιουργία των νέων τετραέδρων που θα αντικαταστήσουν τα μητρικά, μαζί με τον καθορισμό και την ενημέρωση όλων των τοπολογικών και γενεαλογικών τους πληροφοριών.

Ακολουθεί η συνέχιση της επίλυσης των εξισώσεων ροής με το νέο, εμπλουτισμένο, πλέγμα. Η επίλυση συνεχίζεται από το σημείο όπου διακόπηκε για την πραγματοποίηση της προσαρμογής, περιλαμβάνοντας και τα μεγέθη του πεδίου που καθορίστηκαν μέσω παρεμβολής στους νέους κόμβους. Όταν ολοκληρωθεί το προκαθορισμένο πλήθος επαναληπτικών βημάτων, ξεκινά ο επόμενος κύκλος προσαρμογής.

Εμπλουτισμός σε Ροή σε Κανάλι με Εμπόδιο

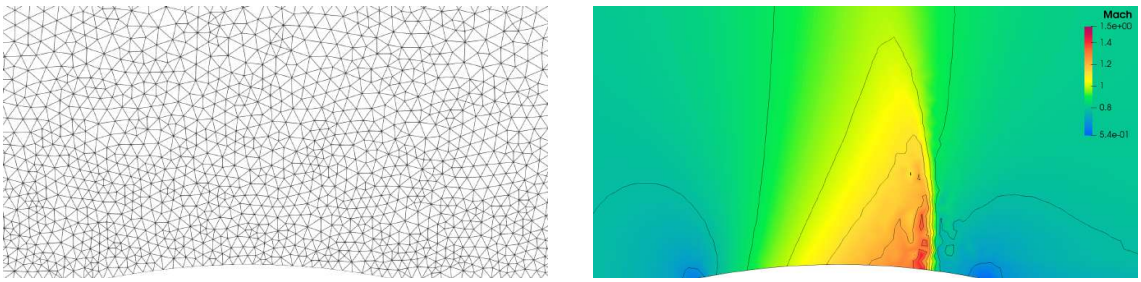
Σε αυτό το κεφάλαιο, ο αλγόριθμος προσαρμογής που αναπτύχθηκε εφαρμόζεται σε ένα πρόβλημα εσωτερικής ροής εντός ενός καναλιού με ένα εμπόδιο. Εξετάζονται δύο παράλλαγές του προβλήματος, μια όπου το εμπόδιο είναι συμμετρικό και μια όπου εισάγεται μια κορυφή στην επιφάνειά του, με σκοπό η ροή να πάψει να είναι, στην ουσία, 2Δ. Η γεωμετρία του χωρίου, συνοδευόμενη από τις οριακές συνθήκες, παρουσιάζεται στο σχήμα 9. Η ροή που εξετάζεται αφορά συμπιεστό ρευστό και θεωρείται μη-συνεκτική με αριθμό Mach 0.8 στην είσοδο. Στη συγκεκριμένη μελέτη, και στις δύο παραλλαγές της, ο μέγιστος αριθμός επαναλήψεων του επιλύτη της ροής τίθεται ίσος με 5000 και συνολικά πραγματοποιούνται 4 κύκλοι προσαρμογής, ένας ανά 1000

επαναληπτικά βήματα.

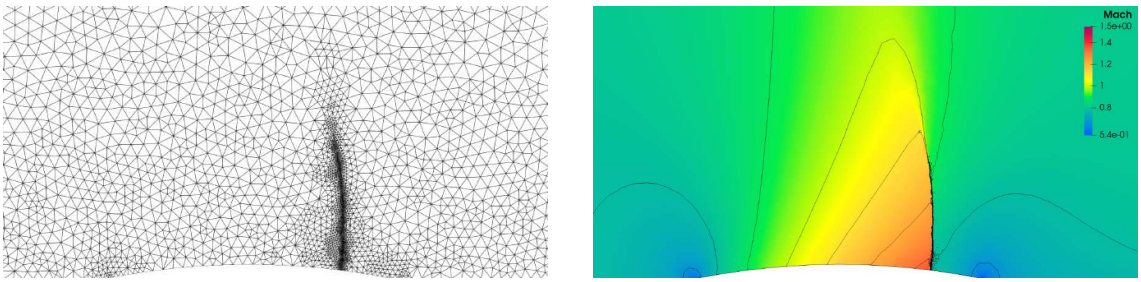


Σχήμα 9: *Γεωμετρία του καναλιού με το εμπόδιο, συνοδευόμενη από τις οριακές συνθήκες.*

Ενδεικτικά στα σχήματα 10- 11 παρουσιάζεται το αρχικό και τελικό πλέγμα μαζί με τα αντίστοιχα πεδία του αριθμού Mach, για την πρώτη περίπτωση συμμετρικού εμποδίου



Σχήμα 10: *Περίπτωση Καναλιού Α: Αρχικό πλέγμα κοντά στο εμπόδιο (αριστερά) και πεδίο του αριθμού Mach (δεξιά). Το πλέγμα αποτελείται από 14324 κόμβους και 68830 τετράεδρα.*

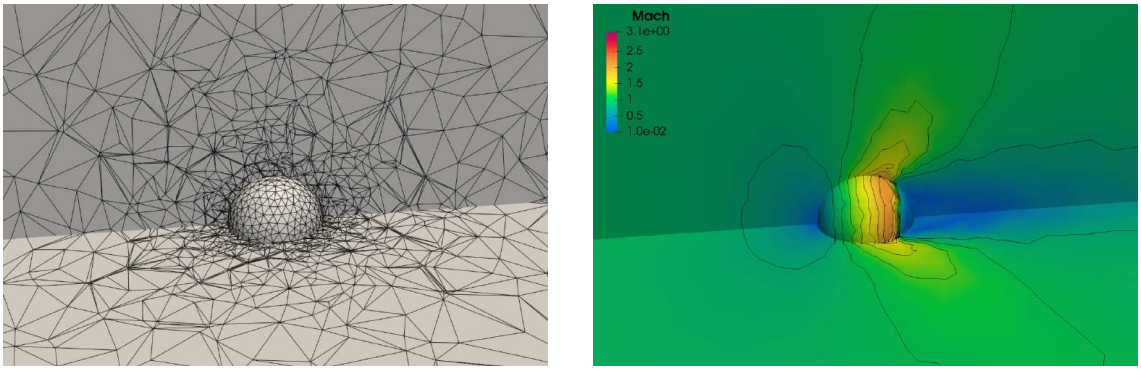


Σχήμα 11: *Περίπτωση Καναλιού Α: Τελικό πλέγμα κοντά στο εμπόδιο (αριστερά) και πεδίο του αριθμού Mach (δεξιά). Το πλέγμα αποτελείται από 122465 κόμβους και 687557 τετράεδρα.*

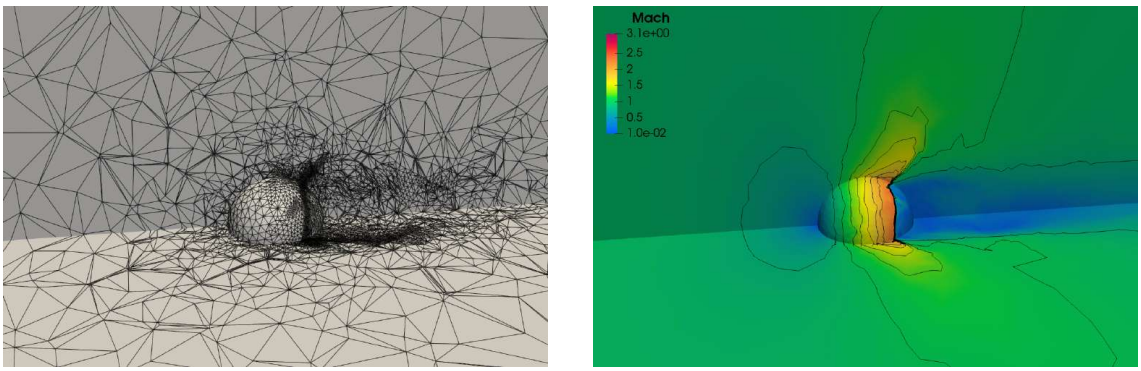
Εμπλουτισμός σε Ροή γύρω από Σφαίρα

Αυτό το κεφάλαιο αφορά ένα πρόβλημα εξωτερικής ροής γύρω από μια σφαίρα. Εξετάζονται δύο παράλλαγές του προβλήματος, με αριθμό Mach της επ΄ άπειρο ροής ίσο με 0.85 και 0.95. Οι μελέτες αυτές βασίζονται σε αντίστοιχη προϋπάρχουσα μελέτη [5]. Η ροή θεωρείται συμπιεστή, μη-συνεκτική. Στη συγκεκριμένη μελέτη, και στις δύο παραλλαγές της, ο μέγιστος αριθμός επαναλήψεων του επιλύτη της ροής τίθεται ίσος με 6000 και συνολικά πραγματοποιούνται 3 κύκλοι προσαρμογής, ένας ανά 1500 επαναληπτικά βήματα.

Ενδεικτικά στα σχήματα 12- 13 παρουσιάζεται το αρχικό και τελικό πλέγμα μαζί με τα αντίστοιχα πεδία του αριθμού Mach, για τη δεύτερη περίπτωση με αριθμό Mach επ΄ άπειρο ροής ίσο με 0.95.



Σχήμα 12: *Περίπτωση Σφαίρας Β: Αρχικό πλέγμα κοντά στην σφαίρα (αριστερά) και πεδίο του αριθμού Mach (δεξιά) υπολογισμένο με χρήση του πλέγματος αυτού. Το πλέγμα αποτελείται από 3973 κόμβους και 21129 τετράεδρα.*



Σχήμα 13: *Περίπτωση Σφαίρας Β: Τελικό πλέγμα κοντά στην σφαίρα (αριστερά) και πεδίο του αριθμού Mach (δεξιά) υπολογισμένο με χρήση του πλέγματος αυτού. Το πλέγμα αποτελείται από 63276 κόμβους και 358934 τετράεδρα.*

Τέλος, στον πίνακα 5.1 παρουσιάζονται οι τιμές του συντελεστή αντίστασης C_D, σε σύγκριση με τις αντίστοιχες τιμές της μελέτης αναφοράς [5].

Αριθμός Mach	Αποτελέσματα με Προσαρμογή	Μελέτη Αναφοράς
0.8	0.57	0.575
0.95	0.9	0.945

Πίνακας 5.1: *Σύγκριση μέσων χρονικά τιμών του συντελεστή αντίστασης, C_D της μελέτης αναφοράς και των προσομοιώσεων που πραγματοποιήθηκαν με εμπλουτισμό.*

Ανακεφαλαίωση και Προτάσεις για Μελλοντική Μελέτη

Ο σκοπός της διπλωματικής εργασίας ήταν η διατύπωση των αρχών του h-εμπλουτισμού πλεγμάτων τετραεδρικών στοιχείων, η ανάπτυξη λογισμικού για την εφαρμογή τους και η δοκιμή του σε διάφορα προβλήματα ροής. Αρχικά παρουσιάστηκε η θεωρία του εμπλουτισμού τετράεδρων, ακολουθούμενη από περιγραφή ενός αλγορίθμου που εφαρμόζει τις αρχές αυτές, για τον οποίο αναπτύχθηκε αντίστοιχο λογισμικό. Στη συνέχεια, το λογισμικό εφαρμόστηκε σε δύο προβλήματα ροής, το καθένα με δύο υποπεριπτώσεις με σκοπό την αξιολόγηση της αποτελεσματικότητάς του. Τα αποτελέσματα επιβεβαιώνουν ότι ο εμπλουτισμός πλεγμάτων αποτελεί έναν αξιόπιστο τρόπο δημιουργίας υψηλής ποιότητας πλεγμάτων, προσαρμοσμένων στις απαιτήσεις κάθε προβλήματος. Το λογισμικό που αναπτύχθηκε απέδωσε αξιόπιστα αποτελέσματα και διαθέτει σημαντικές προοπτικές για βελτιώσεις και περαιτέρω επέκταση.