Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingeniería Aeronáutica y del Espacio

Unstructured Grid Generation using Overset-Mesh Cutting and Single-Mesh Reconstruction

Doctoral Thesis

by

Gennaro Abbruzzese

Madrid, October 2018

Universidad Politécnica de Madrid
**Escuela Técnica Superior de Ingeniería Aeronáutica y
del Espacio**
**Departamento de Matemática Aplicada a la
Ingeniería Aeroespacial**

Unstructured Grid Generation Using Overset-Mesh Cutting and Single-Mesh Reconstruction

Doctoral Thesis for
Ph.D. in Aerospace Engineering

Gennaro Abbruzzese

Supervisors

Prof. Mariola Gómez and Prof. Marta Cordero-Gracia

Madrid, October 2018

Tribunal nombrado por el Sr. Rector Magfco. de la Universidad Politécnica de Madrid, el día.....de.....de 20....

Presidente: _____

Vocal: _____

Vocal: _____

Vocal: _____

Secretario: _____

Suplente: _____

Suplente: _____

Realizado el acto de defensa y lectura de la Tesis el día.....de.....de 20 ...
en la E.T.S.I. /Facultad.....

Calificación

EL PRESIDENTE

LOS VOCALES

EL SECRETARIO

Abstract

In this work a new approach for merging overlapping grids for CFD simulations is presented. The proposed strategy is a preprocessor that integrates the overset meshes into a single mesh suitable for unstructured finite volume vertex-centered Navier-Stokes solvers. This strategy aims to deal with complex industrial problems, especially in case of moving components and overlapping boundary layers. The algorithm consists of two key steps: (1) the definition of a hole cutting area over each component mesh, where nodes of the other mesh are removed, and (2) the subsequent remeshing of the gap generated between both meshes. In this process, the first issue is particularly critical and it employs auxiliary coarse Cartesian meshes to simplify the involved computations like wall distance calculations or point inclusion test. The proposed approach has been demonstrated on 2D test configurations using the DLR-TAU solver.

After a brief introduction of the topic and a state-of-the-art review, the steps of the algorithm conceived are shown. The algorithm is built with two main steps. The meshes cell removal and the hull remeshing to join the two in a single unstructured mesh. The strategy of the first main step consists in the creation of a cartesian auxiliary mesh tailored to perform all the calculations needed to precisely mark the cells to be eliminated from the original meshes. In this process several geometrical methods will be employed to ensure a satisfying outcome, such as Delaunay triangulation, Voronoi diagram and marching cubes. The remeshing step description follows along with remarks on the mesh preparation, outcome evaluation, and some caveats. The whole procedure is detailed for both, 2D and 3D cases. The baseline idea of the 3D algorithm matches with the simpler 2D case but the procedure differs in some critical steps where additional work is needed, like the adjustment of the hull boundaries for hexaedral and tetrahedral cells.

The following chapter describes the results of the CFD simulation run with *DLR-TAU* on the meshes supplied by the developed tool following the previously presented algorithm. The cases tested are two. The first is on a *GARTEUR A310* airfoil where all the three components are meshed separately and then are joined with the algorithm explained. The second case, still in 2D, is a *NACA0012* airfoil where a shockwave is

meshed separately and, still, the meshes are joined in a single one.

Resumen

En este trabajo se presenta un nuevo enfoque para fusionar mallas superpuestas utilizadas habitualmente en simulaciones de CFD. La estrategia propuesta consiste en un preproceso que integra las mallas superpuestas en una única malla, válida para ser utilizada en un solver típico de Navier-Stokes de volúmenes finitos no estructurado y centrado en los vértices. Esta estrategia tiene como objetivo tratar problemas industriales complejos, especialmente en el caso de componentes móviles o capas límite superpuestas. El algoritmo consta de dos pasos clave: (1) la definición de un área común a cada malla componente, donde se eliminan los nodos de una de las mallas, y (2) el subsiguiente remallado del hueco generado entre ambas mallas. En este proceso, el primer punto es particularmente crítico y emplea mallas auxiliares cartesianas bastas para simplificar algunos cálculos involucrados como el cálculo de la distancia a la pared o las pruebas de inclusión de un punto. El enfoque propuesto se ha testado en configuraciones de prueba 2D utilizando el solver DLR-TAU.

Después de una breve introducción del tema y una revisión del estado del arte, se muestran los pasos del algoritmo propuesto. El algoritmo está construido en dos pasos principales. La eliminación de las celdas de las mallas y el remallado del hueco generado, para unir las dos en una sola malla no estructurada. La estrategia del primer paso consiste en la creación de una malla auxiliar cartesiana diseñada para realizar todos los cálculos necesarios para marcar con precisión las celdas de las mallas originales que se eliminarán. En este proceso, se emplearán varios métodos geométricos para garantizar un resultado satisfactorio, como la triangulación de Delaunay, el diagrama de Voronoi y los marching cubes. Para el proceso de remallado hay que seguir una línea de preparación de la malla y evaluación de los resultados, teniendo en cuenta ciertas precauciones. El procedimiento completo se presenta para los casos 2D y 3D. Los fundamentos del algoritmo en 3D coinciden con los del caso 2D más sencillo, pero el procedimiento difiere en algunos pasos importantes, donde es necesario realizar un trabajo adicional, como por ejemplo el ajuste de los límites del hueco para la comunicación entre celdas hexaédricas y tetraédricas.

El siguiente capítulo describe los resultados de la simulación CFD realizada con el

solver *DLR-TAU* sobre varias mallas originalmente superpuestas y unidas con la herramienta desarrollada para este trabajo. Los casos testados son dos. El primero es un perfil de *GARTEUR A310* donde los tres componentes se han mallado por separado y luego se unen con el algoritmo explicado. El segundo caso, también en 2D, es un perfil aerodinámico *NACA0012* donde se ha mallado separadamente una onda de choque y posteriormente se han unido ambas mallas.

Acknowledgements

I would like to express my sincere thanks to Eusebio for welcoming me in the research group and for giving me this wonderful opportunity. A profound gratitude goes to Marta and Mariola, truly dedicated and wise advisors. Similarly, I am in debt with Simone and Scott for their precious time and support during my permanence at AIRBUS. I also wish to thank the entire UPM staff and researchers for the friendly environment, the good advices and the collaboration. A special mention goes to my labmates and friends Silvia, Nuno, Kamil, Raul, Maria Chiara, Moritz and Ollie. Last but not least, I would like to thank my family, my old friends and all the new friends I have met during my PhD, Rossella, Dani, Bibi, Federica, Maria Cristina, Carmen, Ruth and Beatriz.

I would like to acknowledge the European Commission for supporting the AIRUP project: FP7-PEOPLE-2011-ITN, Grant Agreement 2013-608087, and the AIRBUS Company.

ai miei piccoli

Contents

Abstract	i
Resumen	iii
Acknowledgements	v
List of Tables	xi
List of Figures	xiii
	1
1 Introduction	1
1.1 CFD overview	1
1.2 Motivation of the thesis	3
1.3 Previous studies	4
1.4 Some premises	5
1.5 Thesis overview	6
2 The algorithm	9
2.1 The algorithm in two dimensions	10
2.1.1 Hole cutting procedure	10

2.1.1.1	Auxiliary Cartesian mesh	13
2.1.2	Auxiliary mesh refinement	15
2.1.3	Hole-cutting area interface	19
2.1.4	Remeshing	23
2.2	The algorithm in three dimensions	26
2.2.1	The wall mesh	27
2.2.1.1	The zipper mesh	29
2.2.1.2	Hole-cutting volume interface	33
2.2.1.3	Boundary adjustment	40
3	Test cases	49
3.1	<i>GARTEUR A310</i> : Multiple objects case setup	49
3.2	<i>NACA0012</i> : Feature driven case setup	50
3.2.1	Results	53
3.3	NASA CRM: 3D case	59
4	Conclusions	61
5	Appendix	63
5.1	The Delaunay triangulation	63
5.2	The Voronoi diagram	67
5.3	The marching cube method	68

List of Tables

3.1	Test cases mesh dimensions.	50
3.2	Operating conditions of the simulations.	51
3.3	Converged lift and drag coefficients and relative error for the <i>GARTEUR</i> <i>A310</i>	54

List of Figures

1.1	CFD workflow.	2
1.2	vertex - centered cell and cell - centered cell. Ω indicates the cell area . . .	6
1.3	Meshes with invalid cells	7
2.1	Overlapping meshes of the three components of the <i>GARTEUR A310</i>	11
2.2	Overlapping boundary layers of the wing and slat of <i>GARTEUR A310</i> . . .	13
2.3	Grid superimposed on the slat geometry.	14
2.4	Quadtree-like mesh of the slat grid.	15
2.5	Scheme of the hole cutting identification. The zone covered with points is the overlapping area. The striped zone is the portion of the domain that is closer to the slat than to the wing. The hole cutting area is the intersection of the two zones.	16
2.6	Scheme of two different cases for the interpolation when a cell crosses a wall. All distances are referred to the same wall in Q_2 cell. This is not the case for Q_1 and Q_3 cells.	18
2.7	Voronoi diagram filtering: (a) Unfiltered Voronoi diagram, (b) α -angle associated to the Delaunay edge (c) Voronoi edges associated to angles $\alpha < 0.01\pi$ or $\alpha > 0.99\pi$ are filtered, (d) Voronoi edges associated to angles $\alpha < 0.05\pi$ or $\alpha > 0.95\pi$ are filtered.	19
2.8	The two overlapping auxiliary meshes obtained to approximate and map the distance function of the slat and wing.	20

2.9	Contour maps of the interpolated distances with and without mesh refinement.	21
2.10	Auxiliary mesh reshaping: (a) Detail of the slat auxiliary mesh, (b) Interface calculated in correspondence of wall distance difference equal to zero, (c) Auxiliary mesh reshaped on the interface.	22
2.11	Cutting of cells external to the area defined by the distance from the walls.	23
2.12	Flap overset and wing background meshes mended.	24
2.13	Zoom of the boundary layers of the wing and slat of: (a) <i>CENTAUR</i> mesh, (b) incrementally generated mesh.	24
2.14	Final incrementally generated mesh.	25
2.15	$C^2A^2S^2E^2$ mesh employed for Chimera simulation[21].	26
2.16	Full airplane and wing-fuselage junction meshes.	28
2.17	Wall mesh degeneration.	28
2.18	The last row of cells belonging to the eliminated cells is retrieved. On the right we can notice that the high-priority border segments cross all these elements	30
2.19	The strip of cells regenerated defines the planes of triangulation to merge the the two surface meshes. The red and blue are the hems of the two meshes separated by the gap.	31
2.20	A scheme of the gap separating the two borders. On the left the set of segment lying in the same A element belonging to the layer retrieved from the low-priority eliminated cells form, together with the correspondent low-priority border points, a triangulation plane with normal of the A element (in grey). On the right a cross-element segment. The triangulation plane (in white) has the normal of the hi priority cell and contains the points of the segment and the low-priority points belonging to the elements between A and B	32
2.21	Zipper mesh joining the two meshes.	32
2.22	Wall piercing in 3D.	34
2.23	Facets hit by the rays are recovered to be used for the Voronoi filter. . . .	34

2.24	Delaunay triangulation performed on the intersection points.	35
2.25	The filtered Voronoi diagram shown in red.	36
2.26	The refined auxiliary mesh in 3D.	37
2.27	The wall distance mapped on the auxiliary meshes for the full airplane (on the left) and for the wing-wake (on the right).	37
2.28	The interface obtained with the marching cube method for the scalar field calculated as the difference of the wall distance of the wing-wake auxiliary mesh and the wall distance of the full airplane auxiliary mesh (interpolated on the first auxiliary mesh).	38
2.29	The cutting interface coming from the marching cube stage. Thanks to marching cubes method we can smooth with tetrahedra the degeneration volume, which otherwise would be defined by only voxels.	39
2.30	The degeneration of the two CFD meshes produces an empty hull between the two. The hull is then defined by three clusters of cells. In red the wing-wake mesh boundary cells, in green the full airplane boundary cells and in blue the 2D cells reconstructed in the previous steps.	41
2.31	Boundary cells of the inner mesh that compose the hull surface are quadrilateral.	42
2.32	Two strategies for hexaahedral cells transformation. The dark cell is the one on the CFD original mesh boundary.	43
2.33	<i>DLR-TAU</i> quality check for pyramidal cells.	44
2.34	Problematic cells are the boundary layer cells where the curvature of the surface is prevalent on the height.	44
2.35	Hexahedron split in pyramids. The defective cell is in black.	45
2.36	Case of defective cell on the boundary and specular to a boundary cell. The defective cell is in black.	46
2.37	Case of defective cell inside the volume. Two neighbor pyramids are transformed in four tetrahedra.	46
2.38	Position of defective cells in a cut mesh.	47

3.1	CENTAUR mesh generated for the three components: (a) whole mesh, (b) zoom of the boundary layers between the slat and wing.	50
3.2	<i>NACA0012</i> airfoil CENTAUR mesh.	52
3.3	Shock outline of the <i>NACA0012</i> profile.	53
3.4	Hole cutting area around shock outlines of the <i>NACA0012</i> profile.	54
3.5	Overlapping meshes around <i>NACA0012</i> airfoil.	55
3.6	Mesh incrementally generated for <i>NACA0012</i> airfoil.	56
3.7	Pressure coefficients on the <i>GARTEUR A310</i> airfoil: (a) slat, (b) wing, (c) flap	57
3.8	Pressure coefficients of the simulations on the <i>NACA0012</i> airfoil.	58
3.9	CP contour plot over the CRM wing surface and in the Y-slice of the wing.	60
5.1	affine hulls and convex hulls	64
5.2	An example of Delaunay and non-Delaunay triangulation. The correct Delaunay triangulation on the left is made of triangles with a d-hypersphere not including any point.	65
5.3	An example of Voronoi tessellation. Each vertex is the circumcenter of the triangle made of three points	68
5.4	After the identification of the cell nodes and surface-edges contact points the scheme is used to search the correspondent triangulation in the lookup table.	70
5.5	Scheme of the cell nodes and surface-edges contact points. The scheme is used to search the correspondent triangulation in the lookup table.	71
5.6	The surface outcome of the marching cubes method on meshes with hanging nodes (the auxiliary mesh described in Sect. 2.2). The nodes calculated by interpolation on the edges may not match. This produces holes in the surface.	71
5.7	Once identified the voxels cut, the surfaces are triangulated and the their volumes tetrahedralized.	71

Chapter 1

Introduction

1.1 CFD overview

As the reader of the present work might know, the computational fluid dynamics (CFD) is the branch of the physics of the fluid which has the purpose of calculating the conservation laws of the fluids by mean of numerical methods, with the aid of the computer. This science studies the mathematical methods and their implementation in computer software that find generally application to (but they are not limited to) the Navier-Stokes partial differential equations.

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial t}(\rho u_i) = 0$$

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_j u_i) = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} \quad (1.1)$$

$$\frac{\partial}{\partial t}[\rho(e + \frac{1}{2}u_i u_i)] + \frac{\partial}{\partial x_j}[\rho u_j(h + \frac{1}{2}u_i u_i)] = \frac{\partial}{\partial x_j}(u_j \tau_{ij}) - \frac{\partial q_j}{\partial x_j}$$

The complexity, the non-linear nature of the equation and the rarity of cases with analytic solution force us to solve these equations with numerical methods for PDEs. This methods require a huge amount of calculations, reason why we need to resort to

computational methods. These methods are designed to solve, with an iterative approach, the discretized partial differential equations. These equations are dependent on space and time and their solution will depend on the discretization of these quantities. The CFD work-flow might be summarized as in the picture 1.1.

Figure 1.1: CFD workflow.

We realize that the entire topic can be divided in two macro-areas, the solver design, that concerns the strategies to approximate, discretize and define hypothesis and boundary conditions of the equations, and the discretization of the computational domain, described by time and space.

1.2 Motivation of the thesis

As briefly anticipated above, mesh generation is the science that studies the approximation strategies of the physical domain to a computational domain by mean of polygon (in 2D) or polyhedral cells (in 3D). It is a big area of the CFD macro-topic and it can be considered the real bottleneck in the CFD analysis process in industry since meshing time consumption widely exceed the solvers' running time and in man-hours number, taking in account that no calculation can be started without a mesh. Furthermore each solver needs a particular mesh, not only for quality requirement but also for topological requirements. The impossibility by the solvers to process any kind of mesh or damaged mesh can cause a huge time loss from the mesh generation side. In fact, each solver requires a tailored mesh, with precise characteristics and quality criteria. Common practices suggest to use classical geometrical quality criteria [44] to evaluate computational grids. Especially for schemes as vertex-centered finite volume, which manipulate the initial mesh, this evaluation is not sufficient if not misleading. A "good looking" mesh does not bring always an accurate solution. However the influence of mesh features on the CFD predictions are far from being clear [27, 26, 25] and the mesh fitness criteria are often a trade-off between quality and resolution[10, 41]. In order to overcome lack of accuracy or resolution in a mesh, generally are employed techniques as adaptation to increase resolution, a-posteriori evaluation of the mesh, accurate schemes on the solver side or interpolation of solution on overlapping grids (Chimera approach). All these methods are useful but cannot be considered substitutive of a good quality mesh generator or optimization, even because they are able to compensate truncation errors but they give no help in preventing instability of the scheme. High fidelity computations demand the meshing of increasingly more detailed complex geometries with a large range of geometric scales, which requires more user intervention within the mesh generation process in order to fulfill the adequate mesh resolution requirements for accurate computations. Therefore new approaches to increase the flexibility of the mesh generation and to reduce the human intervention in the loop are demanded nowadays.

The intent of the present work is to explore an alternative way to generate in a flexible

and lean way a mesh suitable for DLR-TAU solver.

1.3 Previous studies

Currently, one of the most challenging problem within the aerospace industry is the CFD prediction of the unsteady aerodynamics of complex geometries with moving parts. This problem is present in several applications, *e.g.* moving control surfaces, high lift devices deployment, store releasing, and helicopter/propeller blades. Several techniques have been developed to deal with this kind of problems but the most usual is the overlapping grid technique or Chimera grid approach [11, 39]. This technique permits multiple overset body-conforming grids which are separately generated and can be moved or deformed independently to represent changes of the geometry and relative motion between geometric components. Chimera methodology is gaining popularity in both industry and academic community. In particular, *TAU* solver developed by *DLR* [38, 61, 33] has a robust implementation of this scheme [51, 70]. Chimera is versatile for highly complex geometries, it is relatively well tested and avoids the problem of having to remesh for the simulation of components in relative motion. It consists of a set of high quality meshes generated independently that can communicate with each other and exchange information by interpolation procedures in overlapping zones. Some weaknesses of this methodology are that the grid assembly process is not always fully automated, and the data interpolation in the overlapped regions do not guarantee the conservation of the flow variables [20, 22, 40].

In any case, Chimera scheme is continuously improving and we can mention the progress in the efficiency and the automation of the hole cutting process achieved by Noack *et al.* in Suggar and Suggar++ tools [81, 55], the methodology implemented in TAU (see Spiering [70]) and the work of Chan *et al.* [17][18] whose approach is an evolution of the algorithm introduced by Meakin [54], which employs the computational *x*-rays for points inclusion check in a defined volume. Other approaches have been developed for multiple configurations meshing in order to avoid the interpolation process inherent in Chimera methodology. Already 1986 Rai [59] and 1987 Thompson [72] proposed an approach to generate patched meshes, and in 1995, Kao & Liou [40] introduced the so called

DRAGON mesh as a way to generate a hybrid mesh, avoiding the use of Chimera interpolation. They proposed to join tetrahedra viscous layers with a structured background mesh. On top of this research, new methods have been developed for a two dimensional or semi-2D applications, as the one presented in 2014 by Zhao *et al.* [82] which works on the same concept and tries to automate the hole cutting process to obtain a structured dominant grid. Gremmo *et al.* [35, 34] have presented research notes where it is explained the implementation of a tool for partial remeshing of the background mesh when a solid body and the relative mesh are subject to large displacements. A different way to bypass interpolation problems among overset grids in DLR-TAU solver was proposed by Sørensen [69], generating a single dual mesh from the overset dual meshes obtained by this solver. In his work, the author builds an algorithm in ten steps to merge two different grids in a single one. The aim of the author is to fully automate the meshing process in order to employ the same solver of a single component grid, avoiding interpolation and having control on the whole domain mesh. Based on Sørensen's work, we aim to integrate overlapping primary meshes into a single one in such a way that the developed methodology is independent both of the mesh generation and of the CFD solver. Petersson [56] and Henshaw [36] suggest in their works two ways to assemble overlapping grids. Although for a different goal, some of the issues discussed in these papers (such as the surface reconstruction) are also central in the development of the algorithm presented here.

1.4 Some premises

The software used for the calculations in this work is DLR-TAU. As explained before the idea is to optimize the mesh for this solver. Of course the optimum of criterion for a mesh is strictly related to the scheme used by the solver. For this reason we may find beneficial to outline the main feature of this solver, especially for the one of its finite volume formulations, the vertex centered.

The scheme adopted by DLR-TAU comes from the family of the Godunov type methods [74], wide spread for compressible flow prediction, which smooths the fluxes distribution solving the Riemann problem between each cell. Main characteristic of vertex

– centered scheme is the definition of control volume [77]. In contrast to cell-centered scheme where the control volume is the cell itself, in the vertex – centered the control volume is built around the vertex and the boundaries redefined by new nodes (conservative variables are stored in the vertex of the primary mesh). The mesh built around the vertices is called dual mesh (built by DLR-TAU with a “pre-processing” step), and, precisely, the one employed in DLR-TAU is a “median dual mesh”. Fig. 1.2 shows how the meshes are defined in the two schemes.

Figure 1.2: vertex - centered cell and cell - centered cell. Ω indicates the cell area

This kind of solver has been implemented to calculate gradients and model turbulence on a set of predetermined simple geometries as triangle, square, tetrahedron and hexahedron. It follows that it is impossible to feed the solver with the mesh in presence of hanging nodes or complex polygons/polyhedra Figs. 1.3a and 1.3c. Furthermore, by definition of the finite volume, the grid must cover the entire control volume, the cells must not overlap and cannot have negative volume Fig. 1.3b. In the cases presented above the pre-processor of DLR-TAU is not able to generate a dual mesh for the solver.

1.5 Thesis overview

In this work we have developed a method to manage those meshes typically employed by vertex-centered DLR-TAU solver in such a way that it could be used to process any kind of

Figure 1.3: Meshes with invalid cells

overset, hybrid or non-matching boundaries grids. Therefore, the proposed algorithm must be considered as a pre-processing tool previous to the solver. In the following sections a detailed description of the algorithm is presented. The method is based on two main steps, hole cutting procedure and remeshing. The first one is the most sensitive and it essentially involves blanking cells of a grid in overlapping regions. This stage is based on employing an auxiliary coarse Cartesian mesh designed for such a purpose.

The thesis is structured as follows. In Section 2 we describe in detail the proposed algorithm and the strategies for tackling the different steps: identifying the overlapping zones, how to empty these areas by nodes and cells removal and, finally, remeshing the created hole region in order to merge the remaining meshes. In Section 3 we present results of two numerical simulations done with the DLR-TAU solver on the *GARTEUR A310* and

NACA0012 airfoils. We compare the experimental results [52, 53] with the numerical ones obtained in three meshes: (1) a single mesh generated by CENTAUR mesh software [1]; (2) overlapping meshes solved with Chimera scheme; and (3) the single mesh generated by the designed algorithm from the overlapping meshes. We have prepared an appendix where some of the methods used or cited throughout the development and the explanation of the algorithm. We have chosen *VTK* [4], open source C++ class library for the mesh data structure manipulation and storage and for its visualization algorithms, and the related software for data analysis and visualization *Paraview* [3] to render the meshes and draw the pictures to show the results.

Chapter 2

The algorithm

In the present chapter we discuss the algorithm designed for the purpose explained previously. The algorithm is the product of an iterative process of research of the suitable solution. For this reason the first section of this chapter will show an outline of the procedure, motivating the choices and anticipating the stages of the algorithm. Each step has been presented in detail in the following paragraphs.

The main goal of this work is to develop a tool capable to merge several overlapping meshes into a single one, in a fully automated way and without interfering with the mesh generator nor the CFD solver. The resulting mesh has to be suitable for obtaining accurate solutions, maintaining the original grid quality as much as possible. The proposed algorithm consists of the following steps, considering the case of two overlapping grids.

- Hole cutting procedure.
 - Identification of the overlapping area, *i.e.* zone covered by two different grids.
 - Assignment of a priority level to each overlapping cell that determines the nodes and cells that are going to be eliminated and replaced by the nodes and cells of the other mesh. The process is designed to define an interface, *i.e.* boundary of the intersection where the two meshes are going to be blanked.
 - Points and cells removal.

- Remeshing of the interface to connect the hanging nodes that are orphaned after removal.

The algorithm is designed in order to work both in the two-dimensional and three-dimensional cases. As we will show in the following sections the main steps will remain the same in the two cases but the third dimension poses further challenges. After the explanation of a fully functional example in 2D we will show the additional treatments needed in the three-dimensional case and the problems arising from its complexity.

2.1 The algorithm in two dimensions

2.1.1 Hole cutting procedure

From a theoretical point of view, finite volume methods (FVM) do not allow two cells to cover the same area, because the calculation would end in a violation of the conservation laws. In practice, many FVM industrial solvers are equipped with preprocessing tools to detect such situations. Hence, it is a natural conclusion that the overlap area of two meshes must be identified and covered univocally by a single mesh. This leads to some considerations.

First, the efficient identification of the overlap area is a key aspect of the algorithm. As initial approach, a way to accomplish the task is by inspection of the points of one mesh recursively, in order to know if it is included in some cell of the other mesh. Although this strategy is easy to implement it becomes unaffordable for large size meshes involving millions of nodes. An added problem arises when one mesh crosses a solid wall of the other mesh and falls outside the computational domain (see Fig. 2.1). In this case, the overlap is even more problematic to identify since this area is not twice meshed. At this stage we can perform the inclusion test on a simplified Cartesian mesh, conveniently refined on the boundaries.

Once the overlap is identified, the assignation of the priority to the overlapping cells is an important issue and it has to be performed according to a specific criterion applied

Figure 2.1: Overlapping meshes of the three components of the *GARTEUR A310*.

to each point or cell. For example, even if one of the two meshes were finer and more detailed, it may not be suitable for the phenomena occurring in the other mesh. Thus, the flap mesh may not be well designed to solve the boundary layer of the wing. It is clearly necessary to define a criterion which takes into account this issue. In the present work, we have chosen the distance from the node to each body wall to assign the priority, and we have addressed the efforts to optimize the algorithm on the basis of this criterion. As the calculation has to be performed for each node, some fast search algorithms like $k - d$ tree based approaches (quadtree, ADT, etc) or PDE based numerical methods can be used [12, 75]. In the present work, we have decided to take advantage of the simplified Cartesian mesh that has been generated to carry on the previous inclusion test in such a way that only wall distances of its nodes have to be calculated. Furthermore, taking into account that all components have been meshed separately, it is convenient to generate auxiliary Cartesian meshes for each original CFD mesh attached to each component. The calculated wall distances are transferred among the Cartesian meshes via bilinear interpolation.

In any case, user defined cells, like the structured mesh of the boundary layer for viscous flow simulations should remain unchanged as much as possible. Assuming that all grids were generated for the same purpose, it is reasonable to think that they have similar y^+ and growth ratio, so their cells have comparable dimensions in the proximity of the interface as well. The imposed condition excludes the possibility that the overlapping mesh penetrates the surface of the physical wall of the background mesh.

Finally, an interface has to be identified, which separates the high priority region from the low priority one. This can be done finding an isoline of the adopted criterion value. For this task, we have used the marching squares algorithm, widely employed to generate a contour for a two or three dimensional scalar field [50]. The marching squares method is based on the intersection of the working mesh with a Cartesian mesh designed to this aim. Since a Cartesian mesh is needed in several tasks of the proposed hole cutting process we have decided to generate an unique multi-purpose Cartesian mesh which is used in several different steps of the algorithm.

Since the application of the proposed method includes fast simulation of multiple configuration geometries (*e.g.* different slat positions in a design process of a high lift device) and simulation of moving geometries (*e.g.* store separation) the computational efficiency is a key aspect of the hole cutting and filling algorithm. For this reason in the proposed method we try to perform the majority of the calculations in the early stages of the process. Thus we can avoid repeating some operations each time in case of mesh displacements.

Throughout this section we are going to use two components (slat and wing) of the *GARTEUR A310* to illustrate the algorithm. It should be noted that, in this case, the slat wall lies in the boundary layer mesh of the wing (see Fig. 2.2). This is a typical example where unstructured grid generation software finds difficulties to handle the complexity of the geometry. Let Ω_s be the slat overset mesh and Ω_w the wing background mesh. The internal and external boundaries will be denoted by $\delta\Omega_k^{int}$ and $\delta\Omega_k^{ext}$, respectively, with $k = s, w$ depending on the component.

Figure 2.2: Overlapping boundary layers of the wing and slat of *GARTEUR A310*.

2.1.1.1 Auxiliary Cartesian mesh

A simplified auxiliary mesh is going to be used to define the hole cutting region, D . As we are assuming that there are two overlapping meshes, the hole cutting process is meant to identify the region where one mesh (Ω_w) has to be blanked and replaced by the cells of the overlapping mesh (Ω_s). The main tasks involved in this process are the interface definition and the point inclusion test, to target the points to remove. The use of an auxiliary Cartesian map is specially suited to accomplish these two tasks since it is a simple, robust and relatively fast method.

A common way to perform a polygon point inclusion test is to use a ray casting method. An example of the x -ray approach for overset meshes can be found in [17]. Although this method can be very efficient, the authors acknowledge that it requires significant user interaction which can be sometimes tedious depending on the geometry. For this reason, following the strategy described in [18], we chose to work directly on a Cartesian mesh generated in a similar way to a quadtree mesh. This methodology requires

very little user intervention.

As the wall nodes may have an uneven distribution, a new set of points has been generated to guarantee a minimum dimension of the pixels on the walls. For this purpose, a grid with x -constant and y -constant lines is superimposed on the geometry (see Fig. 2.3), and its intersection with the boundary edges is obtained. The points obtained will be used to guide the generation of the auxiliary quadtree mesh and to construct the Voronoi diagram in a later step.

Figure 2.3: Grid superimposed on the slat geometry.

The outcome of such process for the slat mesh without any additional treatment is shown in Fig. 2.4, where all cells have been refined once. To end this process it is only necessary to eliminate the quadtree nodes lying outside the external boundary of the original mesh. It should be noticed that the cells of the Cartesian mesh inside the body wall are not removed, as they are going to be used to map the distances needed to define the hole cutting interface.

Figure 2.4: Quadtree-like mesh of the slat grid.

2.1.2 Auxiliary mesh refinement

The boundaries of the hole cutting area are defined by the original mesh outer boundary, $\delta\Omega_s^{ext}$, but also by an interface obtained by the criterion imposed by the user. In this case the wall distance has been used, in such a way that elements lying in the overlapping zone and having points closer to a wall belonging to the other component lose priority and are going to be replaced by cells of that other mesh or, if needed, by new cells specifically created. As we want to find an interface that lies at the same distance from the two body walls, we need to compute the distance of each point of the auxiliary meshes to all walls contained in those meshes. To simplify the process, the minimum distance to its own object walls is calculated at each node of the two auxiliary meshes. Then, the nodes are flagged in such a way that the minimum distance is mapped as negative if the node lies

inside the body. These values are transferred from one auxiliary mesh to the other one via interpolation. The difference of the two values are mapped and used to define the interface. The distance interpolation is not carried on if all the vertices of a cell are inner to the body. In this case all the nodes of the other mesh that are contained in this cell are marked to be blanked. This procedure of using a Cartesian mesh to transfer the distance to the points contained in, is similar to that described by Chan & Pandya [18] since the involved algorithms do not require exact values. The more accurate the interpolation is, more precise the interface is. Fig. 2.5 shows a scheme of the hole cutting area obtained with the slat and wing auxiliary meshes.

Figure 2.5: Scheme of the hole cutting identification. The zone covered with points is the overlapping area. The striped zone is the portion of the domain that is closer to the slat than to the wing. The hole cutting area is the intersection of the two zones.

In order to interpolate accurately the wall distance, some further treatments of the auxiliary mesh are needed. For example, in the particular case of highly concave geometries, big cells may have vertices close to two different parts of the body, resulting in a misleading wall distance transfer. To clarify the problem, in Fig. 2.6 we depict an exam-

ple of three cells crossing the body walls ($\delta\Omega_s$ in this case). The minimum distances are successfully transferred to every node inside Q_2 cell. This is not the case for Q_1 and Q_3 cells. As both cells cross the midline inside the body, the slat camber line, some of their vertices are closer to the upper surface and others to the lower surface, so the interpolation will result in an unrealistic value. This example shows that it may occur in geometries with even slight concavity. For this reason cells that cross the midline between the upper and lower surfaces of the component must be refined. So we need to obtain, for each body, the midline in between of its walls and Voronoi diagrams are specially advantageous for this purpose [29].

The nodes previously obtained with the rectangular grid are now used to build the Voronoi diagram. The Voronoi diagram associated to a set of points, called sites, tessellates the space in cells such that each one defines the region of points that are closer to the associated site than to any other. A property of the Voronoi diagram is that each edge of the diagram is dual to an edge of the Delaunay triangulation of the same point set [29]. This means that any two adjacent Voronoi cells correspond to two sites that are connected in the Delaunay triangulation.

The unfiltered Voronoi diagram of the point set obtained by the wall piercing is shown in Fig. 2.7.a. With no filtering, the diagram would guide a refinement even where not needed. Then, it is necessary to identify the different zones of the wall and to isolate the iso-distance lines separating them. Information about the geometry of the wall can be associated to the points extracting the angle of the edge that is crossed at the moment of the wall piercing. As each Voronoi edge is dual to a Delaunay edge, the filter will be applied to the Delaunay edges. Fig. 2.7.b shows that each Delaunay edge is associated to two points on the wall and consequently with two wall edges and two angles. These angles define the relevance of the Delaunay and the Voronoi edges. Low values of such angles mean that the two points belong to the same portion of surface. On the other hand, one angle close to $\pi/2$ means that the Voronoi edge defines the line between two different regions of the wall. In this work, we have designed a filter based on this idea. Two different filtered Voronoi diagrams are shown in Figs. 2.7.c and 2.7.d. In Fig. 2.7.c all Voronoi edges associated to angles $\alpha < 0.01\pi$ or $\alpha > 0.99\pi$ have been filtered, while in

Figure 2.6: Scheme of two different cases for the interpolation when a cell crosses a wall. All distances are referred to the same wall in Q_2 cell. This is not the case for Q_1 and Q_3 cells.

Fig. 2.7.d the filtered angles are $\alpha < 0.05\pi$ and $\alpha > 0.95\pi$. As can be noticed, the second filter is just enough to remove most of the unnecessary edges and lead a more accurate refinement.

In this case, where the slat geometry ends in a cusp defined by more than one node, some lines outside the body can not be eliminated by the filter, as those ones outside the slat in Fig. 2.7. For our purpose these lines have no impact on subsequent calculations, so no additional filtering to remove them has been considered. The final Cartesian meshes,

Figure 2.7: Voronoi diagram filtering: (a) Unfiltered Voronoi diagram, (b) α -angle associated to the Delaunay edge (c) Voronoi edges associated to angles $\alpha < 0.01\pi$ or $\alpha > 0.99\pi$ are filtered, (d) Voronoi edges associated to angles $\alpha < 0.05\pi$ or $\alpha > 0.95\pi$ are filtered.

Q_s and Q_w , after refinement are shown in Fig. 2.8.

2.1.3 Hole-cutting area interface

At this stage, for every node, P_k^i of the auxiliary Cartesian mesh, Q_k , the distances to its walls, $\delta\Omega_k^{int}$, are calculated, d_k^i . Even in case of relative movement of the objects belonging to different meshes, the wall distances will remain fixed. This means that all the calculations performed up to this point remain constant and only the next ones will

Figure 2.8: The two overlapping auxiliary meshes obtained to approximate and map the distance function of the slat and wing.

have to be recalculated in case of displacements.

The information about the wall distance of the wing auxiliary mesh has to be transferred to the slat auxiliary mesh. For each node of such slat mesh, Q_s , we look for the cell of the wing auxiliary mesh, Q_w , that contains it. Then, the wall distance value of the slat node, $\tilde{d}_s^i$, is obtained from the wing auxiliary mesh nodes by using bilinear interpolation.

In order to highlight the need to refine the auxiliary Cartesian mesh to drive the distance interpolation, Fig. 2.9 depicts the contour maps of the interpolated distances from the slat Cartesian mesh to the wing mesh nodes in two cases, with and without refinement. It can be noticed that close to the gap, the interpolation without refinement is not suitable for our purpose.

These operations allow to map the difference between the two distances, $\Delta d_s^i = d_s^i - \tilde{d}_s^i$, which is used both to define the interface and to assign the priority in the mesh

Figure 2.9: Contour maps of the interpolated distances with and without mesh refinement.

cutting process. In this case, the interface is chosen in such a way that this difference is equal to zero, which means that the interface is equidistant to both components walls.

The last step to obtain the interface consists in drawing the isolines and we have used the marching squares algorithm [50] to define them. This method requires to set a particular value for the isoline. This value is searched on the edges of the mesh interpolating the values of the corresponding vertices. Once these points have been found, they must be connected. The marching squares lookup table supplies the configuration corresponding to each possible combination of points lying on the square edges. The method

requires a remapping of the values on a Cartesian mesh, but in this case the auxiliary mesh itself, Q_s , has been used again to perform the operations. This method applied to on a non-constant cell size mesh can end up in non-contiguous interfaces. As we only need to reshape the auxiliary mesh in the hole cutting area, the method is suitable because a high accurate detection of the interface is not needed. The resulting mesh is shown in Fig. 2.10. Starting from the slat auxiliary mesh (Fig. 2.10.a), the interface is defined (Fig. 2.10.b) and, finally, all cells crossed by the interface are cut and triangulated (Fig. 2.10.c) defining the hole cutting area, D .

Figure 2.10: Auxiliary mesh reshaping: (a) Detail of the slat auxiliary mesh, (b) Interface calculated in correspondence of wall distance difference equal to zero, (c) Auxiliary mesh reshaped on the interface.

Notice that the same interface could have been obtained using the Voronoi diagram as it was shown in an early version of the algorithm [5]. However, the new proposed methodology provides a greater flexibility, since we can input any iso-value to determine the interface. This feature can also be used to create a larger gap area. Prescribing an appropriate value allows to ensure the non-intersection of overlapping edges after the removal step. Furthermore, it allows to enlarge the hole cutting area in order to have a smooth transition between the two meshes. In this way, the creation of bad quality cells in small interstices of the blanked area is avoided along the mesh reconstruction.

All cells of the original CFD wing mesh, Ω_w , with some vertex lying inside the obtained hole cutting area, D , are removed. On the contrary, a cell of the original slat

mesh, Ω_s , is removed if some vertex falls outside of the obtained hole cutting, D . An example of the outcome of such cutting process is shown in Fig. 2.11.

Figure 2.11: Cutting of cells external to the area defined by the distance from the walls.

2.1.4 Remeshing

After the hole cutting step, a thin void hull appears between the two meshes, and it must be meshed in the filling process. The simplest and most efficient way to mesh an arbitrary shaped hull is by triangulation. For this task, *CGAL* library [2] has been employed. *CGAL* is a software project that provides easy access to efficient and reliable geometric algorithms in the form of a C++ library. It is one of the most robust and efficient libraries for computational geometry and it is available with dual license. This tool guarantees high quality of the output grid with a good level of automation. *CGAL* constrained Delaunay 2D triangulation routine can force to hold some edges imposed as constraints. In this case, the imposed constraints are the boundaries of the void hull in order to obtain a triangulation that matches the original meshes elements. The polygonal cells outside the void hull frontier are ignored and the remaining ones are added to the full mesh to link the two degenerated meshes. After the single mesh has been obtained it

is possible to apply a smoothing techniques if some bad quality cells appear. In Fig. 2.12 it is shown the stage in which the flap mesh has already been cut and mended. The same process has to be applied to the slat.

Figure 2.12: Flap overset and wing background meshes mended.

Figure 2.13: Zoom of the boundary layers of the wing and slat of: (a) *CEN TAUR* mesh, (b) incrementally generated mesh.

We can compare the mesh automatically generated with *CEN TAUR* (Fig. 2.13a) with the single one incrementally generated with the proposed algorithm (Fig. 2.13b).

The wing has been magnified in the gap area. In the first case, as the slat and wing are really close it is really difficult to increase the size of the corresponding boundary layers. A large part of the gap need to be meshed with triangular cells. In the second case, the boundary layers extend further and the triangular cells area is reduced to a thin band. A good resolution in this area is critical for the accuracy of the computations. The proposed algorithm is specifically designed to keep not only the original topology of the overlapping meshes but also their boundary layer thicknesses.

Figure 2.14: Final incrementally generated mesh.

In Fig. 2.14 we depict the whole single incrementally generated mesh after reconstruction. Notice that the filled mesh at the outer boundaries of the slat and flap elements has not the same quality as the filled areas close to the walls due to the different element size of the initial meshes. Nevertheless, it is expected that these areas will not have a significant impact on the CFD solutions.

2.2 The algorithm in three dimensions

As stated above the algorithm in 3D maintains the main structure made of the hole-cutting procedure and remeshing. Nevertheless the third dimension introduces some issues to be carefully identified and solved. As a sample for the development of the algorithm a NASA CRM mesh has been employed. The mesh in Fig. 2.15, courtesy of DLR, has been employed for Chimera scheme testing in the $C^2A^2S^2E^2$ project [21].

Figure 2.15: $C^2A^2S^2E^2$ mesh employed for Chimera simulation[21].

We summarize here which are the main differences between the simple 2D geometry and this one:

- Shared boundary. The meshes in this case are added to refine poorly meshed zones. These meshes are placed in proximity of the wall and share a large part of the wall area with other meshes. This may happen of course also in the 2 case but in this case the problem is much more difficult to solve.

- Since the wall boundaries are shared the wall distance criterion to assign the priority to each mesh cannot be automated in the same way. We will then assign manually the priority to one mesh on the others when we need to degenerate the wall meshes.
- When we create the hull to be filled, part of the wall surface mesh is removed and the hull remains open. Before proceeding to the regeneration step the hull must be closed regenerating the wall surface mesh in order to close any hole in the hull boundary.
- The problem is split in two. We have to design two different algorithms to deal with the two meshes.
- Three dimensional cells are more complex. The triangulation in presence of hexahedral cells requires additional care.

2.2.1 The wall mesh

As explained in the previous sections the degeneration step consists in a subtraction of cells from the mesh involved. This step produces a hull that has to be filled with a new mesh. The strategy chosen is still the constrained triangulation. This algorithm requires that the hull boundaries are closed. Still, the procedure involves two meshes at a time. Hence, we pick as example the merging of the full airplane and wing-fuselage junction meshes (Fig. 2.16).

We marked in full red the wall of the wing-fuselage junction mesh. Assuming that our degeneration criterion is the same as in the 2D case, the wall distance, we will not be able to apply it on the wall itself. For this reason we have considered as high priority the wall of the wing-fuselage junction. The full-airplane wall mesh is then degenerated in correspondence of the high priority wall mesh as shown in Fig.2.17.

The case of 3D surface mesh is not comparable to the pure two-dimensional case, since it presents some additional issue and it is really difficult to find general purpose

Figure 2.16: Full airplane and wing-fuselage junction meshes.

Figure 2.17: Wall mesh degeneration.

algorithm suitable for the specific case. For these reasons an ad hoc procedure has been defined for the circumstance.

The degeneration of the lower priority mesh has been done by mean of the AABB tree closest primitive search routine available in *CGAL*. For each point of the low-priority mesh the closest high-priority cell is found. If the distance does not exceed a tolerance defined by the user the low-priority point is eliminated.

Having the routines of the *CGAL* library the procedure is not complex. Assuming that the geometry on which the meshes are built is the same, it can fail for really a small tolerance requested or for a considerable difference of cell sizes in correspondence of high level of curvature. Furthermore, such a small tolerance is not really needed and a high value of tolerance can in the worst case introduce a wider gap between the high-priority mesh and the degenerated low-priority mesh with a small impact on the final outcome.

2.2.1.1 The zipper mesh

The next step is to mend together the two surface meshes. This has to be done accordingly to the airplane geometry. Then we had to design an algorithm to triangulate the space between the hems of the two surface meshes. One more time, we used constrained triangulation routine, of which implementation is available in *CGAL*. This, however poses some challenges:

- The 2D triangulation is designed to work on a single plane.
- The geometry of the airplane is described by the airplane wall mesh but this is strongly anisotropic and fine.

The first idea is to use one of the two surfaces as the "true" surface defining the wall. The low-priority surface in particular is suitable because it is possible to use last row of eliminated cell. These cells are defined by the mesh and they are not "dummy" cells as the extruded cells from high-priority mesh would be. Furthermore these cells are surely crossing the high-priority mesh border and we have a univocal correspondence between each border point and one of these cells (Fig. 2.18).

By mean of these cells is possible to define almost all the triangulation planes (Fig. 2.19). At this point we could project the border on the high priority grid on this plane but, not only we considered the process laborious, but also it would need the latter triangulation of the high-priority border cells. Unfortunately, to avoid such inconvenience,

the triangulation planes defined by the low-priority eliminated cells are not sufficient for all nodes.

Figure 2.18: The last row of cells belonging to the eliminated cells is retrieved. On the right we can notice that the high-priority border segments cross all these elements

For this reason we developed an algorithm to choose the plane of triangulation based on logical criteria. First step is to order the segments and points in the same direction. After that the high-priority segments are traversed. It must be checked the retrieved eliminated row element correspondent to the two nodes of the segment. If the points correspond to the same element, the segment will be used to define a polygon with normal equal to the eliminated low-priority cell and assembled with all segment lying in it and the correspondent low-priority border points. This case is shown in the Fig. 2.20, where segments lying in element A create the triangulation plane in grey on the left, with the same normal of element A .

If the points correspond to different elements A and B the triangulation plane will have the normal of the high-priority element correspondent to the segment. The triangulation plane will be a polygon assembled with the high-priority segment points and all the points belonging to the low priority eliminated cells layer situated between A and B . In order to make the traversal agile, the plane allocation is led in this way:

- If the segment points are lying on the same element A only the first point is taken in consideration. It will be allocated to the triangulation plane defined by A , which will be created if not already existing.
- If the segment is cross-element and the first and second point lie respectively on element A and B three actions will be performed.

Figure 2.19: The strip of cells regenerated defines the planes of triangulation to merge the the two surface meshes. The red and blue are the hems of the two meshes separated by the gap.

1. First point will be allocated to triangulation plane defined by A , (which will be created if not already existing).
2. Polygon lying on triangulation plane defined by element A will be closed with the low priority border points belonging to element A .
3. The two point of the segment traversed are used to define the polygon lying on a triangulation plane with same normal of high-priority border cell correspondent to the segment. The polygon is closed with all the points belonging to the low priority eliminated cells layer situated between A and B .

After all the segments are traversed, all the rotation matrices are calculated to rotate the points on a two dimensional system. At this point all the polygons on the triangulation planes are triangulated with the constrained Delaunay triangulation *CGAL* routine. All the triangulation planes assembled by three points are directly transformed in a triangle and the ones containing only two points are ignored. The zipper mesh obtained is shown in Fig. 2.21.

Figure 2.20: A scheme of the gap separating the two borders. On the left the set of segment lying in the same A element belonging to the layer retrieved from the low-priority eliminated cells form, together with the correspondent low-priority border points, a triangulation plane with normal of the A element (in grey). On the right a cross-element segment. The triangulation plane (in white) has the normal of the hi priority cell and contains the points of the segment and the low-priority points belonging to the elements between A and B

Figure 2.21: Zipper mesh joining the two meshes.

2.2.1.2 Hole-cutting volume interface

This step is really similar to the procedure adopted in the two-dimensional case. As previously explained the steps are still the following, with one addition:

- Wall piercing.
- Mid-surface identification.
- Auxiliary generation.
- Distance field identification.
- Iso surface detection.
- Hull boundary adjustment.

We proceed showing the application of the steps in 3D but we explain thoroughly the last step, which is characteristic of the three dimensional case and necessary for the removal of hanging nodes and abutting faces.

In the Fig. 2.22 it is shown the piercing procedure with xyz -rays. The goal is still the same as in the two-dimensional case. The intersection points will be used to build the octree to obtain the auxiliary mesh and the Voronoi diagram to refine it. The faces crossed by the rays (Fig. 2.23) will be used for the Voronoi filter.

We have already explained the reasons why we need the Voronoi diagram so we proceed showing how to reproduce this step in three dimensions. The Fig. 2.24 shows the 3D Delaunay triangulation obtained with the points outcome of the intersection between the wall and the xyz -rays. Furthermore the facets correspondent to each node is represented in yellow. As in the 2D case, where the filter is built accordingly to the angle between the Delaunay diagram edges and the edges of the wall where the Delaunay nodes lay, in this case the check will be done on the 3D angle between the Delaunay edges and the wall facets. The Fig. 2.25 shows the outcome of the filter. Once calculated the points

Figure 2.22: Wall piercing in 3D.

Figure 2.23: Facets hit by the rays are recovered to be used for the Voronoi filter.

coming from the intersection between xyz -rays and wall and the filtered Voronoi facets, it is possible to build the 3D auxiliary mesh, which is shown in the Fig. 2.26.

Figure 2.24: Delaunay triangulation performed on the intersection points.

We proceed showing the steps of the 3D algorithm on the couple airplane/wing-wake meshes. This because the the outcome of the interface definition is more relevant and evident. After the auxiliary mesh definition the wall distance is mapped on the two grids. The scalar map for the two auxiliary meshes is shown in the Fig. 2.27. On the left is shown the wall distance map for the full airplane, whereas the right picture shows the distance mapped on the auxiliary mesh of the wing-wake. Once calculated the wall distance we proceed to the interpolation. The difference of the two mapped wall distance defines then the scalar field on which we obtain the interface by mean of the marching cube method. It is possible to observe the resultant surface in the Fig. 2.28, which shows where the voxels defining the auxiliary mesh are cut by the triangles of the interface. In the image in the bottom we have depicted in red the wall portion covered by the smallest mesh, the one covering the wing-wake. We notice that the interface, that ideally should cut the mesh exactly in between of the red and the blue wall, is not totally accurate. This is an effect driven by the lack of refinement of the auxiliary mesh but it is, in reality, not a big issue, as long as this method is used only to blank the 3D cells. It will produce a bigger hull, which is in some cases even beneficial to leave to the mesh generator space to produce a

Figure 2.25: The filtered Voronoi diagram shown in red.

better mesh. On the other hand it would be a bigger problem if this interface would be used to degenerate the wall mesh. A big space between the hems would be much more inaccurate if we use the method explained in the previous paragraph. That is why we have not calibrated the method for the 2D surface mesh, where we have employed another method (as previously mentioned). However we are confident that with few correction

Figure 2.26: The refined auxiliary mesh in 3D.

the same interface could be used for the two purposes.

Figure 2.27: The wall distance mapped on the auxiliary meshes for the full airplane (on the left) and for the wing-wake (on the right).

Figure 2.28: The interface obtained with the marching cube method for the scalar field calculated as the difference of the wall distance of the wing-wake auxiliary mesh and the wall distance of the full airplane auxiliary mesh (interpolated on the first auxiliary mesh).

At this stage, we have obtained the Hole-cutting interface that identifies the region where the two CFD meshes will be cut (in Fig. 2.29). To be precise, the cells belonging to

Figure 2.29: The cutting interface coming from the marching cube stage. Thanks to marching cubes method we can smooth with tetrahedra the degeneration volume, which otherwise would be defined by only voxels.

the background CFD mesh (the full airplane mesh) that lay inside of the area defined by the interface will be removed. The opposite happens for the wing-wake overlapped CFD mesh. Whatever lay outside of the hole-cutting area is removed. This will generate a

hull separating the two meshes and this space has to be eventually meshed. The outcome of this process is represented in the Fig. 2.30 as a combination of three different groups of cells associated with three different colors. The one depicted in red is the group of cells coming from the inner mesh boundaries. Since the wall mesh (in shiny red) is not degenerated in the zone over this wall, the wall cells are not "uncovered" and do not take part to the hull building. Different case for the wall cells of the outer mesh (the full airplane CFD mesh) which are part of the hull, as the interface lies over the wall zone (the blue zone in Fig. 2.28). The cells in transparent green are coming from the degeneration of the 3D cells of the full airplane CFD mesh. The third group of cells is in blue and represents the 2D cells regenerated with our algorithm explained above.

2.2.1.3 Boundary adjustment

In the 2D case the hole-cutting process would be closed. In 3D still a last step is necessary to prepare the mesh for the triangulation. Once defined the hull boundaries we have to proceed to the meshing. The state-of-the-art tools to mesh a given hull in 3D with an almost certain outcome and with a good level of automation are not many. As explained above, we have chosen the constrained triangulation. To be more precise we have identified as a good tool for this task *TETGEN*, a well maintain library for 3D triangulation, widely employed for academic and industrial applications. It is developed in the Weierstrass Institute by Hang Si and his research group [66] and it comes with GNU Affero General Public License. The issue coming from the usage of this tool is simple to explain. Giving a look to the Fig. 2.31 we can notice that the boundary cells of the inner mesh (and a big part of the outer), which compose the hull surface, are quadrilateral.

Of course the outcome of the 3D triangulation will be a fully tetrahedral mesh, and consequently, a fully triangular cell surface mesh. Without any additional treatment we would end up with hanging faces between tetrahedral and hexahedral meshes. The solver used for this work is still *DLR-TAU* with finite volume scheme and cell vertex formulation. This scheme does not allow us to use meshes with such characteristic and the pre-processor prevents us from feeding the solver with them checking their validity right before generating the dual mesh employed for the calculations. The solution might

Figure 2.30: The degeneration of the two CFD meshes produces an empty hull between the two. The hull is then defined by three clusters of cells. In red the wing-wake mesh boundary cells, in green the full airplane boundary cells and in blue the 2D cells reconstructed in the previous steps.

seem easy but we will see that some expedients are needed for such real case meshes. The tetrahedral mesh cannot be converted in hexahedral mesh, so we have focused our efforts

Figure 2.31: Boundary cells of the inner mesh that compose the hull surface are quadrilateral.

on the manipulation of the original CFD meshes. The first attempt we made is simple and it is illustrated in Fig. 2.32. The hexahedral cell is split in six pyramids and the boundary face is further split in two tetrahedra. This is theoretically sufficient to ensure the communication between the cells of the different meshes. The problem coming from such splitting is that the cells generated in such way are extremely bad shaped, especially in the boundary layer area. One possible further step may be the split of the boundary pyramid in four tetrahedra (still in Fig. 2.32). This expedient brings some benefit but still some cell do not pass the pre-processor quality check. After a thorough examination of the mesh we have realized that these cell were the pyramidal cells generated in the boundary layer cells.

The reason of the failed check in the *TAU* pre-processor is the following (Fig. 2.33). The cell center of the pyramids is calculated and then used to split the pyramids in one smaller pyramid and four tetrahedron. The small pyramid is split in eight small tetrahedra built on the cell center the basis face center and the edge centers. For each small cell generated the *TAU* pre-processor checks if the volume is positive. If even one of

Figure 2.32: Two strategies for hexahedral cells transformation. The dark cell is the one on the CFD original mesh boundary.

these cells of even only one pyramid has negative volume the full CFD mesh is rejected. This because *DRL-TAU* has to ensure that the center of the cell (as *DLR-TAU* calculates it) is inside the cell, as the center of the cells are used to calculate the fluxes among the neighbors in the finite volume calculations. In order to better understand how this can be possible we can give a look to the Fig. 2.34. The picture is meant to schematize a boundary layer cell. These cells are extremely stretched and are designed to pass the quality check in the hexahedron shape. Once split, the pyramids generated are even more compressed and they are considered as negative volume by the pre-processor. This is not theoretically possible for a perfect pyramid but it happens for these cells since the pyramid bases are not perfectly planar. This is a problem coming from surface of curvature, which is not approximated to a planar surface, and probably from a double-precision error that becomes preponderant when the pyramid height becomes extremely small.

Figure 2.33: *DLR-TAU* quality check for pyramidal cells.

Figure 2.34: Problematic cells are the boundary layer cells where the curvature of the surface is prevalent on the height.

For the reasons presented above, some further adjustments are needed. The idea is to find space to improve the quality of the defective pyramid or to remove it. There are three main type of transformations depending on the position of the defective pyramid:

1. On the boundary
2. The cell is specular to the boundary cell (according to our splitting scheme)
3. Inside the mesh

The first two cases are treated in a really similar way. The Fig. 2.35 represents a case of hexahedron split as shown before, the defective cell is darkened. The strategy to

improve the quality of the mesh is common for both cases(defective cell on the boundary and specular to a boundary cell) and it is simple. The pyramid laying on the boundary is removed. Once removed, its specular size is increased moving the central node of the hexahedron on the boundary. In some cases, as shown in the Fig. 2.35, it might be necessary to further move the node beyond the boundary by a small factor. The new boundary is then built with the triangles of the four neighbor pyramids.

Figure 2.35: Hexahedron split in pyramids. The defective cell is in black.

A different solution is needed in case of defective pyramids lying inside of the volume. in this case removing the cell would be impossible and we need to substitute it. The solution we came out with is the following. The idea is to transform the defective pyramid and its neighbor pyramid with four higher quality tetrahedra, as shown in the Fig. 2.37.

Of course, this strategy might is rare cases "infect" some cells inside the volume. This because it may be necessary to split some more hexahedra. This is however generally not an issue. This because the boundary layer might be split on the large face side corresponding to the wall, and treated removing the boundary pyramid, or it can be split leaving uncovered the small sides, and in this case most of the cells are already

Figure 2.36: Case of defective cell on the boundary and specular to a boundary cell. The defective cell is in black.

Figure 2.37: Case of defective cell inside the volume. Two neighbor pyramids are transformed in four tetrahedra.

transformed in hexahedra. The Fig. 2.38 illustrates what said. The strategy for the case of cells on the wall (in red) has no side effects. If the defective cells are inside the volume the neighbor cells have to be split and if these have defective cells some more neighbor cells have to be split and so on. The cells where it might happen are shown in the Fig. 2.38. However it is clear that generally this happens in zones where almost all the cells are already split and this propagation is not threat.

Figure 2.38: Position of defective cells in a cut mesh.

After these treatment the cell passes the *DLR-TAU* quality check. The last step to run the flow solver is to triangulate the hull.

Chapter 3

Test cases

In this section we show the results of two tests run with the DLR-TAU finite volume Navier-Stokes solver [38] on *GARTEUR A310* and *NACA0012* airfoils.

3.1 *GARTEUR A310*: Multiple objects case setup

Three simulations of flow around *GARTEUR A310* airfoil in take off configuration have been made. In the first one, an unstructured single mesh generated for the three components using the CENTAUR mesh generator [1] (see Fig. 3.1.a) has been considered. The second simulation has been run on a set of overlapping grids generated singularly for slat, flap and wing with CENTAUR (see Fig. 2.1). Chimera approach and the new hole cutting procedure available in TAU have been applied to these overlapping grids. Finally, these overlapping meshes have been incrementally merged adding the components one by one with the new proposed methodology (see Fig. 2.14). From the zoom of the boundary layers between slat and wing in Fig. 2.2 we can observe that the complexity of the geometry is an obstacle for the mesh generator since the growth of the boundary layers is limited by the space between the slat and wing.

Tables 3.1 and 3.2 summarize the sizes of the meshes and the operating conditions of the simulations, respectively. It can be noticed that, although the single CENTAUR

mesh has a similar number of points than the two other meshes, it exhibits a different topology. All simulations have been run with Reynolds stress equation model (RSM).

Figure 3.1: CENTAUR mesh generated for the three components: (a) whole mesh, (b) zoom of the boundary layers between the slat and wing.

Mesh	Number of nodes	
	<i>GARTEUR A310</i>	<i>NACA0012</i>
Single CENTAUR mesh	330784	195010
<i>Chimera</i> (after hole cutting)	353371	198928
<i>Incremental</i>	345740	200848

Table 3.1: Test cases mesh dimensions.

3.2 *NACA0012: Feature driven case setup*

Applications of the Chimera approach are not limited to problems involving objects in relative motion. Another important application of this scheme is also the “customization” of different parts of the mesh. In some cases, it is desirable to increase the resolution of the mesh and to have a different mesh topology in certain flow areas. This may happen for example when shock waves or vortex areas (*e.g.* trailing wing vortex) are present in the flow. The traditional approach is to use mesh adaptation to better resolve these flow features, but sometimes, specially for unstructured meshes, one aims not only to

Operating conditions		
	<i>GARTEUR A310</i>	<i>NACA0012</i>
Re	0.605×10^7	0.390×10^7
Mach	0.224	0.751
AoA	12.19°	1.99°

Table 3.2: Operating conditions of the simulations.

increase the mesh resolution but also to use a better quality elements (*e.g.* to use regular hexahedron instead of tetrahedron). In this last case, is usual to overlap block structured meshes with the original unstructured mesh.

In this section we describe the application of the proposed merging algorithm to the situation described above. As a proof of concept we apply the algorithm to increase the resolution in the proximity of a shock wave on a *NACA0012* airfoil at transonic flow conditions (see Table 3.2). Under the operating flow conditions imposed in the simulation, a shock wave appears on the upper surface of the airfoil. The background mesh (Fig. 3.2) for this test case has been generated with CENTAUR.

Furthermore, to show the great flexibility of the proposed methodology, the construction of the auxiliary Cartesian mesh is not based on geometric features of the original mesh, as usual in the Chimera approach but on detected flow physical features. Specifically, the shock wave contours map have been extracted from the values of the variable $U/a \cdot \nabla p / \|\nabla p\|$, being U the velocity vector, a the speed of sound and p the pressure. The outlines corresponding to ± 0.5 in the shock wave zone are used to drive the auxiliary mesh generation. The output of the shock wave feature detection tool is shown in Fig. 3.3.

While in the high lift test case the hole cutting interface has been defined along the zero value of the difference between wall distances mapped on the two auxiliary grids; in the current test case, the hole cutting interface is defined by the zero value of the function $(d_{wing} - d_{shock}) - 2.5 \cdot d_{wing} \cdot d_{shock}$. The objective is to reduce the influence of the wing wall distance in zones far from the wing. Fig. 3.4 shows the hole cutting area on the auxiliary Cartesian mesh obtained with this criterion.

Figure 3.2: *NACA0012* airfoil CEN TAUR mesh.

This region has been meshed by overlapping the original *NACA0012* mesh with a block structured mesh with prescribed element size (see Fig. 3.5). The two meshes are then assembled in a single one following the proposed cutting and remeshing algorithm. In this case, the transition from one mesh to the other one has been smoothed by placing points in the hole area (Fig. 3.6). To assess the proposed methodology, three simulations have been run with TAU using respectively the background CEN TAUR mesh (Fig. 3.2), the overlapping meshes (Fig. 3.5) applying Chimera approach and, finally, the incrementally merged single mesh (Fig. 3.6).

Figure 3.3: Shock outline of the *NACA0012* profile.

3.2.1 Results

The aim of this section is to assess the meshes incrementally generated with the developed methodology and to confirm some valuable features of the algorithm. All simulations have been run until convergence of the lift and drag coefficients. The converged values of both coefficients for the *GARTEUR A310* test case are summarized in the Table 3.3 along with the relative error of the lift coefficient to the experimental value ($CL_{exp} = 2.983$) [52].

Although from this test is not possible to determine if the error is systematic, it is possible to see that the assembled mesh has an error on the prediction which does not exceed 2%. Furthermore, even without mesh smoothing, the convergence is not affected by the bad quality cells generated with the constrained triangulation, as it was expected.

It is well known that the prediction of drag for high lift configuration is much more

Figure 3.4: Hole cutting area around shock outlines of the *NACA0012* profile.

Lift and drag coefficients (<i>GARTEUR A310</i>)			
	CL	$err(\%)$	CD
CENTAUR	2.975	0.2648	0.0484
<i>Chimera</i>	2.943	1.351	0.0550
<i>Incremental</i>	2.925	1.954	0.061

Table 3.3: Converged lift and drag coefficients and relative error for the *GARTEUR A310*.

challenging than lift. The drag of a three-element configuration is strongly dependent on the pressure distribution on the slat, especially at the leading edge area. The minimum pressure distribution level is also influenced by the cove flow separation and reattachment (recirculation bubble at the slat cove) and slight variations of the separation areas influences strongly in the slat pressure distribution. This could explain the larger differences on the prediction of the pressure distribution on the slat among the three mesh approaches (Fig. 3.7a) and consequently the larger difference in the computed drag values. Note that the pressure differences on the wing (Fig. 3.7b) and on the flap (Fig. 3.7c) are much lower because in this area the main component cove separation is so not critical.

Figure 3.5: Overlapping meshes around *NACA0012* airfoil.

The pressure coefficients for the NACA0012 airfoil are plotted in Fig. 3.8 in which results of the numerical simulations are compared to the experimental ([53]). It is important to point out the low discrepancy in the behavior among the computational flows. As in the case of GARTEUR A310 wing there are no substantial differences between the results of the Chimera approach and the single incremental merged mesh. The two simulations have been performed on the same set of meshes, whereas the three components mesh has a different topology. So, the pressure distribution calculated from the incremental merged mesh is comparable to the other two meshes. The proposed methodology provides an alternative to the Chimera approach, preserving similar performances but with the additional computational advantage of solving the CFD equations on a single mesh.

Figure 3.6: Mesh incrementally generated for *NACA0012* airfoil.

Figure 3.7: Pressure coefficients on the *GARTEUR A310* airfoil: (a) slat, (b) wing, (c) flap

Figure 3.8: Pressure coefficients of the simulations on the *NACA0012* airfoil.

3.3 NASA CRM: 3D case

The same tests performed for the 2D cases have been done for the three-dimensional case. The test case chosen is the NASA CRM shown in the Sect. 2.2, in the algorithm explanation in 3D. The test has been done merging the full background and the wing wake meshes. Unfortunately for the 3D case the results are not as satisfying as in the 2D cases. The computations diverge after few iterations. This result does not change under different simulation settings. Looking at the Fig. 3.9, the source of divergence on the calculation is evident. In the newly created cells the values of the flow variables are clearly not homogeneous. The communication between the two degenerated cells fails in this cells and the instability quickly propagates to the entire computational domain. The cause is as clear. The *TETGEN* algorithms used are not designed to deal with extremely anisotropic hull boundaries nor to generate anisotropic tetrahedra. The result of the triangulation is shown at the bottom in the Fig. 3.9. The Y-constant slice of the wing makes clear the bad quality of the newly generated mesh. The mesh generation tries as much as possible to generate high quality cells according to a shape criterion. The problem in this case is the communication between cells. Extremely small cells have extremely big neighbor cells and the mesh is far from being structured, which means that the number of neighbors of each cell can be extremely variable. This means that a triangulation algorithm must be designed exactly for CFD purpose. The quality criterion of classical triangulation formulation does not fulfill the CFD mesh requirements. For this purpose many authors have investigated the topic to obtain an automatic mesh generator mixing the Delaunay triangulation desirable features with the object functions to get CFD conforming meshes.

Figure 3.9: CP contour plot over the CRM wing surface and in the Y-slice of the wing.

Chapter 4

Conclusions

In the present work an new component in the CFD workflow has been presented and discussed. Such algorithm, placed in between of the mesh generation and the simulation steps, has been designed in order to allow the generation of meshes where each component or zone of the computational domain is meshed separately from the others. The method has been conceived to work in both two-dimensional and three-dimensional cases and to produce a single mesh suitable for CFD calculations with the finite volumes vertex-centered scheme implemented in the *DLR-TAU* solver. The motivation is to provide an alternative to the Chimera approach, preserving similar performances but with the additional computational advantage of solving the CFD equations on a single mesh.

In Chapter 2 we have explained step by step the algorithm and the motivations of the chosen procedure. As anticipated in the abstract, the algorithm is made of two macro steps. The degeneration phase and the regeneration of the meshes. The process has been conceived in order to fulfill the solver requirements and to maximize the efficiency. All the calculation are made "reusable". For this reason has been considered advantageous to implement the construction of an auxiliary mesh relative to the geometry of the computational domain and independent on its mesh. The new proposed method is aimed to be applied on meshes without any limitation of components. All the calculations performed on the previous stages to the triangulation are only slightly affected by the discretization

of the domain. Once the core steps of the algorithm have been performed, the meshes can be refined or even substituted with a low impact on the remaining calculations. These auxiliary meshes are employed as the basis of the subsequent calculations intended to define the hole cutting interface for each CFD mesh. The regeneration of the meshes has been performed with the Delaunay triangulation, for its properties, such robust implementation and high level of automation.

In Chapter 3 we discussed the application of the method on real cases in the two-dimensional and three-dimensional spaces. The first stage, the interface detection and mesh degeneration, has demonstrated to be suitable and transferable from the 2D to the 3D with the addition of a small step. The method can be considered flexible and simple of the mesh generation process and, especially capable to adapt the mesh to moving objects. Furthermore, the algorithm is robust, efficient and highly parallelizable, in accordance with the industrial demands. On the other hand the current implementation of the method has not yet reached an optimal level of maturity. It is not fully automated and it still requires some level of user interaction. Problems come in the reconstruction phase. Whereas in the 2D case the Delaunay triangulation is a perfectly suitable tool for mesh reconstruction, in 3D it cannot be considered the right choice. The reason is that this mesh generation algorithm has no state-of-the-art implementation capable to deal with the anisotropy requested for CFD meshes (e.g. in the boundary layers). In fact, this is a field under development and research. In the zone outcome of the Delaunay triangulation layer the calculations diverge extremely easily producing no outcome to evaluate. The algorithm is particularly interesting if applied in 3D. For this reason, as a future development, it would be surely needed an investigation on the right hull filling algorithm to create the mesh to join different components.

Chapter 5

Appendix

5.1 The Delaunay triangulation

The Delaunay triangulation is probably the most employed and oldest method for meshing along with the simple cartesian mesh. The advantage over the second mentioned method is surely that the Delaunay triangulation falls in the category of the unstructured mesh, which makes it more suitable for real world applications.

For clarity we present it in formal notation and we present some concepts as defined by Cheng, Dey and Shewchuk [19]. We introduce then some fundamental notion regarding the triangulation.

The first is the *affine combination* of points. An affine combination is a point that can be defined as $p = \sum_{i=1}^k w_i x_i$ for a set of points $X = (x_1, x_2, \dots, x_k)$ in $\mathbb{R}^d$ and a set of scalar weights w_i with the property $\sum_{i=1}^k w_i = 1$.

If this condition is not satisfied the point is called *affinely independent*. In $\mathbb{R}^d$ it is not possible to have more than $d + 1$ affinely independent points. The set of all affine combinations of points in X is called *affine hull*. To have a practical idea of what said it is useful to resort to the Fig. 5.1.

Figure 5.1: affine hulls and convex hulls

The convex hull of a set T of $k + 1$ affinely independent points is a k -simplex and we can denote it as σ_T . The *simplicial complex* K is a set of simplices where:

- The face of every simplex K is also in K .
- If two simplices $\sigma_T, \sigma_D \in K$, then $\sigma_T \cap \sigma_D = \sigma_T$

The *underlying space* of K is the pointwise union of its simplices. A *triangulation* τ of a set of finite points S in $\mathbb{R}^d$ is the simplicial complex τ where:

- S is the set of vertices of τ .
- The union of all simplices of τ is the convex hull of S .

It follows that a simplicial complex is a triangulation of S if each of its vertices is a point of S and if its underlying space is the convex hull of S .

The *Delaunay triangulation* (in Fig. 5.2) is a specific kind of triangulation discovered and described by Boris Nikolaevich Delaunay in the 1934 [23]. The Delaunay triangulation presented in the paper is a 2D triangulation made of triangles with a peculiar characteristic. No points of S lies in the circle circumscribing the triangles. As generalization in $\mathbb{R}^d$ we define *Delaunay* a triangulation when it is entirely made of Delaunay simplices. No point of S lies in the d -hypersphere circumscribing each simplices.

Figure 5.2: An example of Delaunay and non-Delaunay triangulation. The correct Delaunay triangulation on the left is made of triangles with a d -hypersphere not including any point.

The Delaunay triangulation presents some interesting features and several applications but, also as we will see, some drawbacks. The first reason for the popularity of the triangulation is that it allows us to work with points clouds. Often medical imaging or detection in any engineering area is in form of points cloud. Of course the points may be provided in chaotic patterns. For this reason the objects with the smallest cardinality (triangles in 2D and tetrahedra in 3D) are the most suitable to reconstruct the desired area or volume.

The Delaunay triangulation has another really attractive feature, which is also one of the reasons we chose it for this work. Although the automatic meshing is still far from being fully automated, the Delaunay triangulation might be considered probably

the meshing algorithm closest to a full automation especially in the 2D case, where this method is by far more mature. The three dimensional case on the other hand has some critical issues and it is less developed than the simpler 2D case. The standard Delaunay triangulation process produces a convex hull. If we need to have a mesh which is conforming to the boundary we need to resort to the *constrained Delaunay triangulation*. In a constrained triangulation some of the edges (generally the boundary edges) are required to become edges of the triangulation. These edges cannot always be Delaunay but still the CDT maintains some desirable properties of the classical Delaunay triangulation as it maximizes the minimum angle, minimizes the largest circumscribing circle, and minimizes the largest.

The CDT though presents even more problems and it is area of development in this topic. In case of highly skewed cells in the boundary its existence is not always guaranteed and it could be needed the insertion of the so-called Steiner points.

Probably the most active area of research related to this topic is the development of the anisotropic mesh generation. Controlling the generation of cell with different size or with desired size criteria is really difficult and an open issue. This made the Delaunay triangulation not suitable for some application as the boundary layer meshing.

A comprehensive survey on the history, applications and most employed algorithms for generation of this data structure can be found in [73]. A plethora of applications can be found in literature. In [30], for example, the authors present an algorithm to generate the Delaunay triangulation for data interpolation. Other sources can be found where the author proposes an implementation and generation of the triangulation like [14], which uses it for medical imaging, [13, 16, 64, 60] for computer graphics and mesh generation in general and even [9, 79] and [63] with applications to fluid dynamics simulations. In the most successfully algorithms employed in the state-of-the-art software we can include the Bowyer-Watson [15, 78] and the Tanemura-Ogawa-Ogita [71]. Also triangulation is particularly advantageous for its capability to be easily optimized. A big branch of the topic is, in fact, the development of optimization algorithms [29, 24, 31, 49, 37, 65, 43] and untangling following deformation [45, 32] As stated before the *CGAL* library has been used for 2D triangulation. An explanation of the algorithm contained is in [14], while

[80, 57, 47] can be used as user references. For the 3D case *TETGEN* has been chosen as meshing tool. The technical documentation with an explanation of the algorithms has been written by Si H. in [66]. Also Si presents in cooperation with Shewchuk [68] and Gärtner [67] algorithms for constrained Delaunay triangulation for meshing of piecewise linear complexes, also implemented in the *TETGEN* library.

5.2 The Voronoi diagram

Another topic strictly related to the Delaunay triangulation is the Voronoi diagram. This is one of the most important and widely employed data structures in computational geometry. Voronoi diagram has been extensively investigated and the mature implementations and sources for deepening can be found in [48],[58], [28], [8] and [42]. In two and three dimensions, the Voronoi diagram is a tessellation, referred to a set of point, made of the regions of space that lies closer to each point than the other points (5.3). We report a formal definition of the Voronoi diagram based on euclidean distance. The diagram built on a set of points $P = p_1, p_2, \dots, p_n$ associates a Voronoi region to each point

$$V(p_i) = \{x \in \mathbb{R}^d : \delta(x, p_i) \leq \delta(x, p_j), \forall j \leq n\}$$

with δ the euclidean distance function. It follows that the region $V(p_i)$ is the intersection of all the $n - 1$ half spaces separating the couples of points. These regions are exactly n and they can be unbounded and they are always convex.

The edges/faces of the Voronoi tessellation are made of the perpendicular bisectors of the lines connecting close points, so each vertex is the circumcenter of the triangle made of three points. For this reason, as we have seen in the previous chapter, the dual diagram of the Voronoi is a Delaunay triangulation.

The 2D and 3D cases are of course the most employed and intuitive. The importance of the Voronoi diagram has been understood since its formalization, as the Delaunay triangulation around one century ago ([76]), but it is gaining more and more popularity due to the increase of the need of computational algorithms for geometry and graphics,

Figure 5.3: An example of Voronoi tessellation. Each vertex is the circumcenter of the triangle made of three points

CAD/CAE and robotics. This does not mean that the investigation on this structure is limited to the third dimension, as the abstraction in the n th dimension can be as significant for many problems and area of research for the scientific community. The applications of this data structure covers a wide range of problems. Knuth in its well known "The Art of Computer Programming" [46] presents an application of the Voronoi algorithm for associative file searching problem known as "the post-office problem" and employed in many algorithms presented in [58]. Applications for cluster analysis can be found in [6] where the Voronoi neighborhood are used as features in dot pattern processing. Finally also applications in robotics, such as collision detection, are present in literature in [62] with the construction of a generalized Voronoi diagram for a set of n circular discs in time $O(n \log^2 n)$, and in [7] where a generalization of Voronoi called power diagrams is presented.

5.3 The marching cube method

This method has been presented by William E. Lorensen and Harvey E. Cline in 1987 for medical data interpretation and still remains an effective strategy to build an iso-surface out of a scalar field. The method was born to work on discrete scalar fields output of medical machines scans, computed tomography, magnetic resonance and single-photon

emission computed tomography. These instruments supply data in discrete fashion and often structured in voxels as output of a series of planar slices or ray casting. From this comes the need of a method capable to recover a surface out of a regularly voxelized 3D space. The idea comes from the need to identify surfaces of determined value in a 3D scalar map. We briefly explain the concept behind and the way we employed it. Once assumed that the scalar field is built of voxels we know that the scalar values are stored in regularly distributed nodes. First step we need to identify the zones where the scalars assume the desired value. This is a simple process. The interesting part of the marching cube method is the second step, the construction of the surface. To accomplish this task the authors suggests to proceed as follows.

The outcome of such process is a set of points for each voxel. Using linear interpolation the number of these points will be one at most. Each of the eight nodes is then marked as outside/inside. This means that each voxel will have $2^8 = 256$ combinations of states. Instead of triangulating the nodes the authors use a lookup table that for each pattern points to a pre-built set of triangles. In the Fig. 5.4 a few of these possible triangulations are shown. The outcome is extremely fast and it does not require special additional treatments.

The output surface will depend on where it is supposed to cross the edges of each voxel. The position of the surface on the edges is calculated interpolating the node values on the edges. In this way not only we find the position of the surface but also we can encode the combination of points obtained to build and quickly consult the lookup table of the possible triangulations. The Fig.5.5 outlines the contact points outcome of the interpolation used to encode the permutation for the cell.

It is necessary to remark here that the marching cubes method has been used in an unconventional way in the present work. In fact, the method should be used on any regular mesh with no hanging nodes or abutting facets. This is because the interpolation could produce non coincident nodes on the surface and, then, open surfaces. In Fig. 5.6 it is represented the outcome of the marching cubes method on non homogeneous cells, which produces holes in the resultant surface.

Figure 5.4: After the identification of the cell nodes and surface-edges contact points the scheme is used to search the correspondent triangulation in the lookup table.

The reason why we can still make a good use of this method is that our goal is to cut the cells with the surface and to use the remaining part of the cell to perform an inclusion test for mesh nodes. For this reason the part of cells result of the cut with the surface is simply triangulated. In this way we obtain a compact volume mesh.

Figure 5.5: Scheme of the cell nodes and surface-edges contact points. The scheme is used to search the correspondent triangulation in the lookup table.

Figure 5.6: The surface outcome of the marching cubes method on meshes with hanging nodes (the auxiliary mesh described in Sect. 2.2). The nodes calculated by interpolation on the edges may not match. This produces holes in the surface.

Figure 5.7: Once identified the voxels cut, the surfaces are triangulated and the their volumes tetrahedralized.

Bibliography

- [1] Centaur software 2017, www.centaurosoft.com.
- [2] The computational geometry algorithms library (CGAL), www.cgal.org.
- [3] Paraview, www.paraview.org.
- [4] The visualization toolkit (VTK), www.vtk.org.
- [5] Abbruzzese, G., Gómez, M., Cordero-Gracia, M. Unstructured 2d grid generation using overset-mesh cutting and single-mesh reconstruction. *Aerospace Science and Technology*, 78:637 – 647, 2018.
- [6] Ahuja, N. Dot pattern processing using voronoi neighborhoods. *IEEE Trans. Pattern Anal. Mach. Intell.*, 4(3):336–343, Mar. 1982.
- [7] Aurenhammer, F. Improved algorithms for discs and balls using power diagrams. *Journal of Algorithms*, 9(2):151 – 161, 1988.
- [8] Aurenhammer, F. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, Sept. 1991.
- [9] Baker, T. J. Mesh adaptation strategies for problems in fluid dynamics. *Finite Elements in Analysis and Design*, 25(3):243 – 273, 1997. Adaptive Meshing, Part 2.
- [10] Batdorf, M., Freitag, L. A., Ollivier-gooch, C. Computational study of the effect of unstructured mesh quality on solution efficiency. In *Presented at the 13 th Annual Computational Fluid Dynamics Meeting, Snowmass Village*, 1997.

- [11] Benek, J. A., Arnold Engineering Development Center. *Chimera: A Grid-embedding Technique : Final Report*. Arnold Engineering Development Center, 1986.
- [12] Boger, D. A. Efficient method for calculating wall proximity. *SIAM Journal on Scientific Computing*, 39:2404–2406, 2001.
- [13] Boissonnat, J.D. Shape reconstruction from planar cross sections. *Computer Vision, Graphics, and Image Processing*, 44(1):1 – 29, 1988.
- [14] Boissonnat, J. D., Devillers, O., Teillaud, M., Yvinec, M. Triangulations in cgal (extended abstract). SCG '00, pages 11–18, New York, NY, USA, 2000. ACM.
- [15] Bowyer, A. Computing dirichlet tessellations. *Comput. J.*, 24:162–166, 02 1981.
- [16] Cavendish, J.C., Frey, W.H., Marin, S.P. Feature-based design and finite element mesh generation for functional surfaces. *Advances in Engineering Software and Workstations*, 13(5):226 – 237, 1991.
- [17] Chan, W. M., Kim, N., Pandya, S. A. Advances in domain connectivity for overset grids using the x-rays approach. *Seventh International Conference on Computational Fluid Dynamics (ICCFD7)*.
- [18] Chan, W. M., Pandya, S. A. Advances in distance-based hole cuts on overset grids. *22nd AIAA Computational Fluid Dynamics Conference Dallas, TX*, 2015.
- [19] Cheng, S. W., Dey, T. K., Shewchuk, J. R. *Delaunay Mesh Generation*. Chapman and Hall / CRC computer and information science series. CRC Press, 2013.
- [20] Crippa, S. Application of novel hybrid mesh generation methodologies for improved unstructured cfd simulations. *28th AIAA Applied Aerodynamics Conference - CFD Drag Prediction Workshop Results, 28 June - 1 July 2010*, 2010.
- [21] Crippa, S. Application of chimera with hexahedral blocks in solar grid. *C²A²S²E, Institute of Aerodynamics and Flow Technology, DLR*, 2011.
- [22] Crippa, S. Application of chimera with hexahedral blocks in solar meshes. *Third Symposium "Simulation of Wing and Nacelle Stall", 21st - 22nd June 2012*, 2012.

-
- [23] Delaunay, B. Sur la sphère vide. *Bulletin de l'Académie des Sciences de l'URSS, Classe des sciences mathématiques et naturelles*, page 793–800, 1934.
- [24] de L'isle, E. B., George, P. L. Optimization of tetrahedral meshes. *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*, 75:97–127, 1995.
- [25] Diskin, B., Thomas, J. L. . Accuracy of Gradient Reconstruction on Grids with High Aspect Ratio. *NIA Report No. 2008-12*, 2008.
- [26] Diskin, B., Thomas, J. L. . Notes on accuracy of finite-volume discretization schemes on irregular grids. *Applied Numerical Mathematics*, 2009.
- [27] Diskin B., Thomas J. L. Mesh effects on accuracy of finite-volume discretization schemes. *AIAA-2012-0609*, 2012.
- [28] Edelsbrunner, H. *Algorithms in Combinatorial Geometry*. Springer-Verlag, Berlin, Heidelberg, 1987.
- [29] Edelsbrunner, H., Shah, N. R. Incremental topological flipping works for regular triangulations. In *Proceedings of the Eighth Annual Symposium on Computational Geometry*, SCG '92, pages 43–52, New York, NY, USA, 1992. ACM.
- [30] Farin, G. Surfaces over dirichlet tessellations. *Computer Aided Geometric Design*, 7(1):281 – 292, 1990.
- [31] Freitag, L. A., Ollivier-Gooch, C. Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering*, 40(21):3979–4002, 1997.
- [32] Freitag, L. A., Plassmann, P. Local optimization-based simplicial mesh untangling and improvement. *International Journal for Numerical Methods in Engineering*, 49(1-2):109–125, 2000.
- [33] Gerhold, T. TAU overview. 2008.

-
- [34] Gremmo, S., Antonik, I., Corso, P., Hirsch, C., Coussement, G. Partial remeshing strategy for cfd simulations when large displacements of moving bodies are taken into account. *24rd International Meshing Roundtable (IMR24)*, 2015.
 - [35] Gremmo, S., Antonik, I., Nigro, R., Hirsch, C., Coussement, G. Partial remeshing strategy for cfd simulations when large displacements are taken into account. *23rd International Meshing Roundtable (IMR23), Research Notes*, 2014.
 - [36] Henshaw, W. Generating composite overlapping grids on cad geometries. *8th International Conference on Numerical Grid Generation in Computational Field Simulations*, 2002.
 - [37] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W. Mesh optimization. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 19–26, New York, NY, USA, 1993. ACM.
 - [38] Institute of Aerodynamics and Flow TechnologyBraunschweig. Technical Documentation of the DLR TAU-Code Release 2011.2.0.
 - [39] , J. L., Dougherty, F. C., Benek, J. A. A chimera grid scheme. *Advances in Grid Generation*, 1983.
 - [40] Kao, K., Liou, M. Advance in overset grid schemes - From Chimera to DRAGON grids. *AIAA Journal*, 33(10):1809–1815, Oct. 1995.
 - [41] Katz, A., Sankaran, V. Mesh quality effects on the accuracy of CFD solutions on unstructured meshes. *J. Comput. Physics*, 230(20):7670–7686, 2011.
 - [42] Klein, R. Simplified abstract voronoi diagrams, 2007.
 - [43] Klingner, B. M., Shewchuk, J. R. Aggressive tetrahedral mesh improvement. In *In Proc. of the 16th Int. Meshing Roundtable*, pages 3–23, 2007.
 - [44] Knupp, P. M. Algebraic mesh quality metrics. *SIAM J. Sci. Comput.*, 23(1):193–218, Jan. 2001.
 - [45] Knupp, P. M. Hexahedral and tetrahedral mesh untangling. *Engineering with Computers*, 17(3):261–268, 2001.

- [46] Knuth, D. E. *The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1998.
- [47] Kruithof, N. 2D periodic triangulations. CGAL Editorial Board, 4.12.1 edition, 2018.
- [48] Lee, D. T. and Preparata, F. P. Computational geometry a survey. *IEEE Trans. Comput.*, 33(12):1072–1101, Dec. 1984.
- [49] Li, T.S., Wong, S.M., Hon, Y.C., Armstrong, C.G., McKeag, R.M. Smoothing by optimisation for a quadrilateral mesh with invalid elements. *Finite Elements in Analysis and Design*, 34(1):37 – 60, 2000.
- [50] Lorensen, W. E., Cline, H. E. Marching cubes: A high resolution 3d surface construction algorithm. *COMPUTER GRAPHICS*, 21(4):163–169, 1987.
- [51] Madrane, A., Heinrich, R., Gerhold, T. Implementation of the chimera method in the unstructured hybrid dlr finite volume tau-code. *6th Overset Composite Grid and Solution Technology Symposium, Ft. Walton Beach, Florida, USA, October 8-10, 2002*, 2002.
- [52] Manie, F., Piccin, O., Ray, J. P. Report of the 2d model m1 in the onera f1 wind tunnel. *GARTEUR High Lift Action Group AD*, 1989.
- [53] McDevitt, J. B., Okuno, A. F., Ames Research Center. *Static and dynamic pressure measurements on a NACA 0012 airfoil in the Ames High Reynolds Number Facility*. NASA technical paper. National Aeronautics and Space Administration, Scientific and Technical Information Branch, 1985.
- [54] Meakin, R. Object x-rays for cutting holes in composite overset structured grids. *15th AIAA Computational Fluid Dynamics Conference*, 2001.
- [55] Noack, R. W., Boger, D. A., Kunz, R. F., Carrica, P. M. Sugar++: An improved general overset grid assembly capability. In *Proceedings of the 47th AIAA Aerospace Science and Exhibit*, 2009.

- [56] Petersson, N. Hole-cutting for three-dimensional overlapping grids. *SIAM Journal on Scientific Computing*, 21(2):646–665, 1999.
- [57] Pion, S., Yvinec, M. 2D triangulation data structure. CGAL Editorial Board, 4.12.1 edition, 2018.
- [58] Preparata, F. P., Shamos, M. I. *Computational Geometry: An Introduction*. Springer-Verlag, Berlin, Heidelberg, 1985.
- [59] Rai, M. M. A conservative treatment of zonal boundaries for euler equation calculations. *Journal of Computational Physics*, 62(2):472 – 503, 1986.
- [60] Schroeder, W., Shephard, M. Geometry-based fully automatic mesh generation and the delaunay triangulation. *International Journal for Numerical Methods in Engineering*, 26:2503 – 2515, 11 1988.
- [61] Schwamborn, D., Gerhold, T., Heinrich, R. . The dlr tau-code: Recent applications in research and industry. *In proceedings of “European Conference on Computational Fluid Dynamics“ ECCOMAS CDF 2006, Delft The Netherland, 2006*, 2006.
- [62] Sharir, M. Intersection and closest-pair problems for a set of planar discs. *SIAM J. Comput.*, 14:448–468, 05 1985.
- [63] Sharov, D., Nakahashi, K. A boundary recovery algorithm for delaunay tetrahedral meshing. *5th international conference on numerical grid generation in computational fluid dynamics and related fields*.
- [64] Shenton, D., Cendes, Z. Three-dimensional finite element mesh generation using delaunay tessellation. *IEEE Transactions on Magnetics*, 21(6):2535–2538, November 1985.
- [65] Shewchuk, J. R. Two discrete optimization algorithms for the topological improvement of tetrahedral meshes. *In Unpublished manuscript*, 2002.
- [66] Si, H. Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw.*, 41(2):11:1–11:36, Feb. 2015.

- [67] Si, H. and Gärtner, K.", editor="Hanks, Byron W. Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *Proceedings of the 14th International Meshing Roundtable*, pages 147–163, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- [68] Si, H. and Shewchuk, J. R. Incrementally constructing and updating constrained delaunay tetrahedralizations with finite-precision coordinates. *Engineering with Computers*, 30(2):253–269, Apr 2014.
- [69] Sørensen, K. Multi-component modeling with the dual mesh connect approach. *Computational Flight Testing*, 123:257–271, 2013.
- [70] Spiering, F. *Development of a Fully Automatic Chimera Hole Cutting Procedure in the DLR TAU Code*, volume 132. Notes on Numerical Fluid Mechanics and Multidisciplinary Design, 2016.
- [71] Tanemura M., Ogawa T. and Ogita N. A new algorithm for three-dimensional voronoi tessellation. *Journal of Computational Physics*, 51(2):191 – 207, 1983.
- [72] Thompson, J. F. A general three-dimensional elliptic grid generation system on a composite block structure. *Computer Methods in Applied Mechanics and Engineering*, 64(1):377 – 411, 1987.
- [73] Thompson J. F., Soni B. K., Weatherill N. P. Handbook of grid generation. *CRC Press*, 1998.
- [74] Toro, E. F. *Riemann solvers and numerical methods for fluid dynamics : a practical introduction*. Springer, Berlin, New York, 1997.
- [75] Tucker, P. G. Differential equation-based wall distance computation for des and rans. *Journal of Computational Physics*, 190(1):229 – 248, 2003.
- [76] Voronoi, G. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. *Journal für die Reine und Angewandte Mathematik*, 1907.
- [77] Wang, G., Schwöppe, A., Heinrich, R. Comparison and evaluation of cell-centered and cell-vertex discretization in the unstructured tau-code for turbulent viscous flows.

- V European Conference on Computational Fluid Dynamics ECCOMAS CFD 2010*, 2010.
- [78] Watson, F. D. Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. *Comput. J.*, 24:167–172, 02 1981.
- [79] Weatherill, N. P., Hassan, O. Efficient three-dimensional grid generation using the Delaunay triangulation. pages 961–968, 1992.
- [80] Yvinec, M. 2D triangulation. CGAL Editorial Board, 4.12.1 edition, 2018.
- [81] Zhang, H., Fan, J., Yuan, W., Zhang, W. Structure overset grid method and its applications to simulation of multi-body separation. *Science China Physics, Mechanics & Astronomy*, 58(9):594–702, Sep 2015.
- [82] Zhao, Y., Yu, B., Yu, G. A new two-dimensional hybrid grid generation method based on improved hole cutting. *International Communications in Heat and Mass Transfer*, 54:103 – 114, 2014.