Vrije Universiteit Brussel

Unstructured Hexahedral Non-conformal Mesh Generation

Konstantin Kovalev

A thesis submitted to obtain the degree of
Doctor in Engineering Sciences

Promotors:

Professor Charles Hirsch
Professor Chris Lacor

Faculty of Engineering
Vrije Universiteit Brussel
Belgium
December 2005

Jury Members :

Prof. Dr. Ir.	Hugo SOL	***president***
Prof. Dr. Ir.	Jean VEREECKEN	***vice-president***
Prof. Dr. Ir.	Steve VANLANDUIT	***secretary***
Prof. Dr. Ir.	Herman DECONINCK	***(VKI)***
Prof. Dr. Ir.	Pierre BECKERS	***(ULg)***
Prof. Dr. Ir.	Charles HIRSCH	***promotor***
Prof. Dr. Ir.	Chris LACOR	***promotor***

Dedicated to

My parents and Maria...

Acknowledgements

In the end of the long educational journey, I would like to express my deepest gratitude to those who voluntarily invested their time, experience, patience, and even love to make this period of my life unforgettable and eventually very fruitful.

First of all, I would like to thank Professor Charles Hirsch, the former Head of the Department of Fluid Mechanics of VUB for the opportunity to realize the present thesis. For the last five years he has always been able to help me find new sources of motivation and to advance in the right direction in my work. His expertise and advices have always been priceless for me. It has been an honour and a pleasure to study under the supervision of such exceptional professional, teaching and personal qualities.

Furthermore, I would like to thank Marc Tombroff, the General Manager of NUMECA International, for giving me the opportunity to perform the work described in the thesis in the dynamic and competitive environment, which represents this company. Collaboration with researchers working for the company was also invaluable.

I would like to thank Dr. Michel Delanaye, who was my technical co-adviser during the first two years of studies. I appreciate his great teaching abilities as well as his exceptional expertise in computational fluid dynamics. I also would like to thank my numerous colleagues all of whom participated in various parts of the completed work: Ir. Özgür Uğraş Baran for his friendship and ideal partnership in many aspects of our work, Ir. Benoît Leonard for his priceless help with the flow solver, and many others: Dr. Alpesh Patel, Dr. Mohamed Mezine, Dr. Lionel Temmerman, Ir. Hervé Stassin, Ir. Kang-Joon Verniers, Dr. Miles Elsden, Dr. Pierpaolo Borelli, Ir. Andrea Lani, Dr. Koen Hillewaert, Dr. David Vanden Abeele. Also, I would like to thank Ir. Michel Pottiez, the Technical Director of the company, for his consistent and professional management, and his wife Jenny D'haes who provided much help as a secretary of the Department of Fluid Mechanics. Many thanks to Professor Chris Lacor for his help with some organizational aspects. Also, my student colleagues from the Department of Fluid Mechanics should not be forgotten. I would like to thank Ir. Sergey Smirnov, Dr. Jan Ramboer, Ir. Bamdad Lessani, Ir. Cristian Dinescu, and many others. And, undoubtfully, the list of all people to whom I am thankful would be infinite.

The completed work would never be possible without every day's help, support and love of my wife Maria. I would like to thank her for highly professional editing of the thesis.

Without her, this work would not only be impossible and unbearable but it would simply be pointless.

I would like to express my exceptional gratitude to my parents, Vladimir Kovalev and Natalia Kovaleva, who supported me with their care, love, and a good advice during the long period of my education.

Finally, I would like to acknowledge European Community for its support throughout the Growth TROPHY project funded by the European Commission under the contract No. G3RD-CT-2001-00510.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Scope and motivation

Mesh generation is a process of spatial subdivision of a generally complex domain into simple-shaped sub-volumes of pre-defined topology. These sub-volumes are usually referred to as mesh *elements* or *cells*. Elements of a mesh are connected to each other, they do not intersect with each other and cover the entire domain.

The problem of generation of computational meshes arose as a pre-requisite for numerical analysis of physical phenomena. When computational power of new computers increased sufficiently for modeling physics using discrete approaches such as finite-difference, finite-volume and finite element methods, spatial discretization emerged as a critical issue. On one hand, all requirements of numerical methods for computational meshes should be met during mesh generation. For example, these requirements may include limitations on types of elements allowed in the mesh, on mesh quality, or on connectivity between mesh elements. On the other hand, meshing tools should be capable of successful treatment of the widest possible class of geometries. These two factors provide a perpetual challenge for creators of mesh generation tools for numerical analysis.

Initially, the family of finite-difference methods was spread widely as the primary tool for numerical analysis. As these methods are based on numerical discretization of partial differential equations describing simulated phenomena, they require *structured* hexahedral meshes to be used. A structured mesh (more commonly referred to as *a grid*) can be defined as a mesh in which an identical pattern of elements is connected to any internal node (for instance, there are 8 cells connected to each internal node in a hexahedral grid). Development of finite-volume and finite-element families of methods significantly broadened the class of meshes used in numerical analysis. Particularly, the *unstructured* methods (methods which generate meshes with generally arbitrary patterns of cells connected to each internal mesh node) quickly became a very attractive subject of interest due to several main reasons.

To begin with, an unstructured mesh is able to meet certain requirements of node distribution and element sizing in a much better way than a structured one. The absence of integral structure makes unstructured meshes generally more flexible. Another reason is that the unstructured methods provide fruitful environment for automation of meshing process. Unlike structured grids, unstructured meshes can always be modified locally: elements can be inserted or removed; connectivity can be modified. Effects of such modifications are always local and are limited to modified elements and their direct neighbors. This feature enables the unstructured methods to choose appropriate actions automatically based on a local analysis of geometry and topology. Therefore, automatic mesh generation for an arbitrary domain is finally brought to scope of practical problems. Besides, unstructured meshes may optionally contain mixed-type elements or non-conformal interfaces between cells. These extensions increase flexibility and robustness of the unstructured methods but require special treatment in flow solvers.

The recent evolution of numerical analysis and design methods, combined with quickly growing range of simulated phenomena and the constantly increasing complexity of involved geometries, provide a high demand for powerful automatic meshing tools. Meshing became a necessary component of industrial design loops and a decrease in its cost is as important as that of other components. Moreover, meshing fills the gap between geometry processing in CAD systems and numerical analysis. The only user input at this stage is the control over meshing process. Therefore, automation is the ultimate direction of evolution of up-to-date mesh generation tools.

Although the problem of automatic *tetrahedral* mesh generation has practically been solved and is successfully employed in the industry, there is still a high degree of interest in fully automatic hexahedral mesh generation. An obvious reason is that hexahedral elements are naturally suitable for numerical analysis for both finite-volume and finite-element methods. In numerical solution of Navier-Stokes equations, a flow is usually parallel to solid boundaries and its details can be captured better by means of hexahedral elements. In finite-element analysis, hexahedral linear elements have been proved to be capable of providing more accurate results than tetrahedral linear elements. Another strong advantage of hexahedra is that for a certain number of nodes, fewer elements are required to mesh a domain as compared to tetrahedra. It means that less memory is used for the mesh and solution data storage and a shorter time is required to compute the solution. Therefore, automatic hexahedral unstructured mesh generation remains to be one of the biggest challenges among industrial demands for numerical design.

1.2 Outline of methods developed in the thesis

In recent years attempts to solve the problem of automated unstructured hexahedral meshing have been focused on the few principal approaches reviewed in the following sections. Among others, the so-called *Overlay* method is considered as one of the most robust automatic tools capable of dealing with geometries of practically arbitrary complexity. This

approach generates a conformal Cartesian non-boundary fitting grid that encompasses an entire domain. The initial mesh is refined via the Octree technique until cell size requirements are met. Afterwards, cells located outside the domain are deleted; the remaining mesh is connected to domain boundaries and its quality is optimized.

The Overlay approach is considered as one of the most promising directions towards full automation of hexahedral meshing process. Certain critical aspects of this approach constitute the subject of this thesis. Namely, the following issues are covered:

- Mesh quality improvement via untangling concave cells.
- Mesh quality improvement via local optimization-based mesh smoothing.
- Generation of high quality viscous mesh near solid walls.
- Mesh deformation.

1.2.1 Untangling concave cells

The term untangling describes a transformation of a cell shape from concave to convex. Generally, the issue of improvement of mesh cell quality represents a combination of two operations applied to mesh cells, one of which is untangling. Its goal is to convert concave cells into convex ones via finding new appropriate positions for their vertices. Accordingly, the second operation is optimization, which is aimed at improving cell quality according to a certain quality measure. The problem of untangling consists in finding *any* convex cell shape, while the goal of optimization is to find a convex cell shape of best quality. The former can be solved via linear programming methods that are generally fast and not demanding.

1.2.2 Optimization of mesh cells

As the untangling process recovers only invalid (concave) cells, it is clearly insufficient for a global mesh quality improvement. Even if all mesh cells are valid, the low quality of mesh cells (for instance, too small dihedral angle or another quality measure) may degrade not only the convergence rate of computations but also the accuracy of final results. Mesh optimization procedure is aimed at direct improvement of mesh cell quality. It is based on repositioning mesh vertices to minimize a certain functional. This functional expresses the deformation energy of a cell with respect to shape of an undeformed cell. Generally, this functional represents a non-linear function of mesh vertex coordinates. Unlike the untangling procedure, solving an optimization problem is usually quite time-consuming.

1.2.3 Viscous mesh generation

Modern methods for viscous computations as well as integrated physical models often require high quality of computational meshes. This includes smoothness of viscous grids as well as control of position of the first mesh point close to wall, depending on the type of a turbulence model [87, 112]. Viscous layer insertion methodology developed in this thesis addresses the above issues. It combines the Laplacian smoothing with the untangling and optimization procedures in order to inflate Euler mesh cells adjacent to solid boundaries (i.e. increase their normal sizes). The inflated cells are refined tangentially until a desired number of anisotropic cell layers is inserted with their vertices being redistributed in normal direction.

1.2.4 Mesh deformation

There exists a wide class of physical phenomena modeling of which requires flow analysis in domains of shapes that vary during computation. This class encompasses multiple coupled problems with moving or deforming interfaces, fluid-structure interaction problems and many others. Solving such problems demands that a computational mesh follows deformation of domain boundaries. Therefore, two main possibilities exist. The first one is to fully re-mesh the domain every time its shape is modified. The second is to deform an existing mesh according to deformation of its boundary. However, in many phenomena mentioned above, relative deformation of mesh boundary is small and full re-meshing appears to be too costly. Instead, deforming an already existing mesh may be quick and reliable provided that appropriate tools are available. Development of such a tool for unstructured hexahedral non-conformal meshes is addressed in the thesis. The proposed methodology preserves mesh topology and modifies only positions of vertices. Displacements of boundary points are propagated to volume mesh points. Optimization loops are incorporated in order to ensure high mesh quality.

1.3 Thesis structure

The present thesis consists of 7 chapters.

Chapter 2 contains a detailed review of recent literature related to topics covered in the thesis and draws conclusions on main contributions of present work. The Chapter is divided into five sections. First, a survey and a comparison of most common state-of-the-art approaches to hexahedral unstructured meshing are given. Following that, methods for isotropic mesh quality improvement are reviewed, with the emphasis on the most recent methods for unstructured meshes. The next section reviews recent methods for generation of high-quality unstructured anisotropic meshes as well as up-to-date criteria of quality of such meshes. In the last section of the Chapter, a survey of mesh deformation techniques and their adaptation to unstructured applications is presented.

Chapter 3 presents a description of the unstructured hexahedral meshing technology as implemented in the new generator. Three sections constitute this chapter. Section 1 provides a general description of the overlay mesh generation technique and framework implemented in the software. Section from 2 to 7 detail the principal algorithms involved in the process. Geometry definition, initial mesh construction, Octree adaptation, boundary fitting, quality improvement and insertion of viscous layers are described. Finally, Section 8 describes the results of application of the developed generator to a set of geometries. Conclusions about the performance of the method are drawn in Section 9.

Chapter 4 describes the approach for isotropic mesh quality improvement suggested in the present thesis. The chapter consists of six sections. Section 1 introduces the problem of quality improvement of isotropic unstructured meshes. Section 2 describes technical details of the "untangling" approach. The problem is formulated and the method for solving it is described. Similarly, Section 3 formulates the problem of mesh optimization and provides a detailed description of the methodology developed to solve it. Section 4 provides details of the approach which combines the "untangling" and optimization techniques. This combined approach enables the both techniques to show their maximum efficiency. Section 5 presents the results of application of the developed techniques to real test problems; the advantages and disadvantages are discussed. Finally, Section 6 reflects conclusions about basic properties of the techniques and suggests recommendations for further developments.

Chapter 5 presents the methodology for generation of high-quality unstructured hexahedral meshes for viscous computations developed herein. The chapter contains six sections. Section 1 gives an introduction to the issue of anisotropic hexahedral meshing. Section 2 provides a general description of viscous layer insertion within the framework of overlay methodology. Section 3 formulates the problem of viscous mesh generation. Section 4 describes the solution methodology. In Section 5, the results of application of the method to a set of test problems are presented and analysed. Finally, in Section 6, conclusions regarding performance of the methodology are drawn and recommendations for future evolution are proposed.

Chapter 6 describes application of the developed hexahedral meshing methodology to the problem of deforming and moving meshes. Five sections constitute this chapter. Section 1 introduces the problem of mesh deformation. Section 2 contains details of formulation of the problem. Section 3 describes the methodology for deformation of unstructured hexahedral meshes presented herein. Section 5 presents and discusses the results of application of the mesh deformation technique to test problems. Section 6 provides conclusions about efficiency of the method and gives recommendations regarding future work.

Chapter 7 discusses a real industrial problem to which the mesh deformation technique described in Chapter 6 was applied. This Chapter contains a description of application of the mesh deformation technique to the problem of aquaplaning of a car tire. This phenomenon was studied using the *Fluid-structure interaction* (FSI) methodology. Results of this study are presented in comparison with experimental data.

Finally, Chapter 8 finalizes the achievements of the work and provides general recommen-

dations regarding future developments.

Chapter 2

Literature Review

2.1 Unstructured hexahedral mesh generation

The interest in automatic unstructured hexahedral mesh generation has increased significantly in the last two or three decades. There are several reasons for this growth. To begin with, up-to-date industrial design loops require optimization of certain stages in order to decrease the overall turnaround time, while manual mesh generation tools often appear to be excessively time-consuming. Secondly, the continuously increasing computational power of modern computers nowadays allows for implementation of techniques and algorithms that were unavailable earlier due to excessive computational cost. Finally, the increasing reliability of numerical modeling tools induces a gradual substitution of experimental investigations with computational analysis at certain design stages. Obviously, under these circumstances, creation of powerful automatic meshing tools becomes a *must*.

The modern trends in automatic unstructured mesh generation include two principal directions - tetrahedral and hexahedral methods. Initially, the tetrahedral methods [9,16–19,49, 50] gained primary attention due to their robustness and the high level of automation. However, automatic meshing of a domain with hexahedral elements has always been the highest challenge due to its attractive qualities. Firstly, hexahedral elements can be naturally oriented along a wall thus providing optimal conditions for flow analysis via *Finite-Volume Method* (FVM). Furthermore, tangential anisotropic refinement of wall-adjacent hexahedra is capable of producing semi-structured wall-aligned layers of highly stretched cells, which is beneficial for high quality viscous computations. Besides that, the number of hexahedra is several times lower than the number of tetrahedra necessary to mesh the same domain with the same number of nodes. This may result in a significant decrease in flow analysis time.

One of the families of approaches to unstructured hexahedral meshing is based on the idea of automatic domain decomposition into blocks of simplified topology that can be meshed according to a certain pre-defined pattern. One of the first techniques on automatic decomposition of a 2-D domain was proposed by Boehmann *et al* [15]. A domain is

decomposed via the quad-tree subdivision based on local sizes of geometrical features. The resulting blocks are meshed with quadrilateral elements such that the final mesh conforms to domain boundaries. Similar approach was suggested by Talbert and Parkinson [114]. In this method, an initial domain is decomposed into simple polygons and these polygons are meshed according to a certain pattern. The *medial axis* subdivision technique had first been introduced by Tam and Armstrong [115]. A medial axis is the set of lines and curves swept by the center of the maximum circle that can be fitted into the area of interest. Axes are swept by the center as the circle rolls through the area (see Figure 2.1). Similar approach in 3-D is referred to as the *medial surface* subdivision [4, 79, 102, 103]. In this approach, planes and surfaces are swept by the maximum sphere center instead of lines and curves. While the 2-D medial axis approach has proved its relatively high efficiency in automatic domain decomposition into meshable regions, the 3-D medial surface methodology encounters robustness problems in the generation of medial surfaces and meshing of regions bounded by medial surfaces.

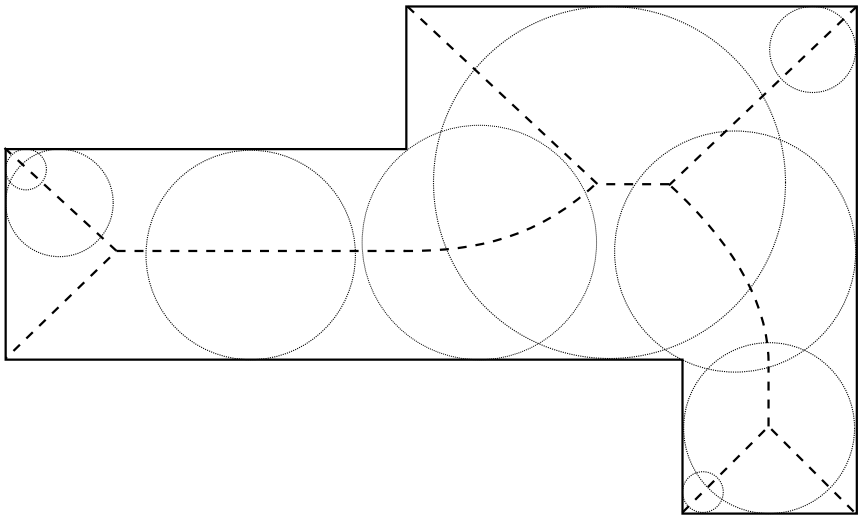


Figure 2.1: Medial axis - maximum circle rolling throughout the domain.

Another group of methods for quadrilateral and hexahedral unstructured meshing is referred to as the *advancing front* techniques. In the 2-D version of this technique, dealing with quadrilateral elements called *paving* [13, 27, 122], rows of such elements are added along domain boundaries (see Figure 2.2). Rows originating from different boundaries are matched to result in a conformal mesh. One of the most critical aspects is meshing the remaining void successfully when fronts from different boundaries converge. The method had been extended to 3-D surface meshing problems by Cass *et al* [27]. Similar approach in 3-D is referred to as *plastering* [12, 24, 113]. In this method, layers of hexahedral elements are generated along boundaries (see Figure 2.3). As in paving, fronts built on different surfaces are connected together to preserve mesh conformity. However, the problem of meshing the void which remains when different fronts converge to each other is generally

not yet solved. While paving had successfully been proved to be robust and reliable, its three-dimensional counterpart lacks robustness in many classes of problems.

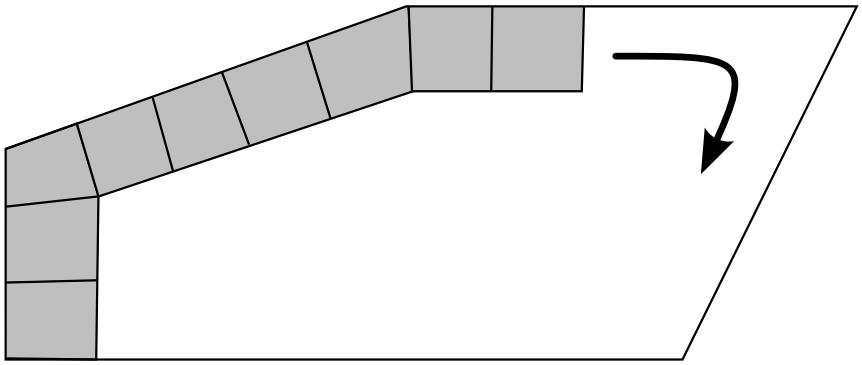


Figure 2.2: Paving.

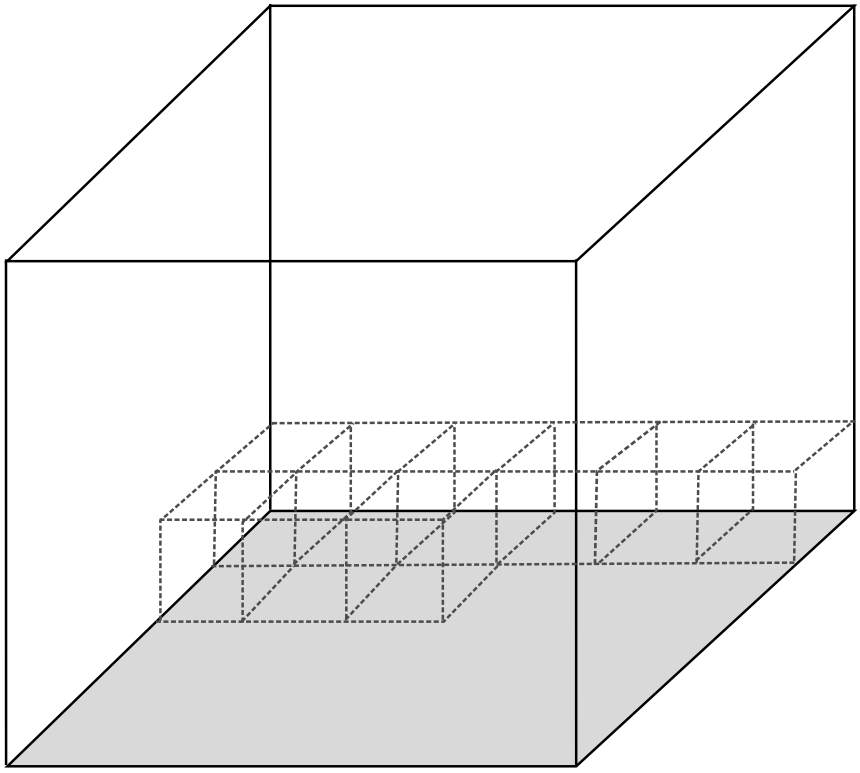


Figure 2.3: Plastering.

In attempts to overcome the difficulties posed by plastering, an alternative algorithm called *whisker weaving* was developed [22, 40, 118]. The algorithm employs the so-called *Spatial Twist Continuum* (STC) or *the dual* of a volume mesh [91]. The idea behind Whisker weaving can be described as follows. Let the dual of a surface quadrilateral mesh be

generated. Let this dual be extruded into the volume bounded by this surface. This results in generation of a set of surfaces (*twist planes*) which intersect the surface mesh along the lines of its dual (see Figure 2.4). Once a valid set of twist planes is found, hexahedral elements can be generated wherever the three twist planes converge. As the intersection of twist planes is important in topological sense only, there is no need to compute any intersections. The method is relatively young and must still prove itself as a reliable meshing tool for many classes of problems.

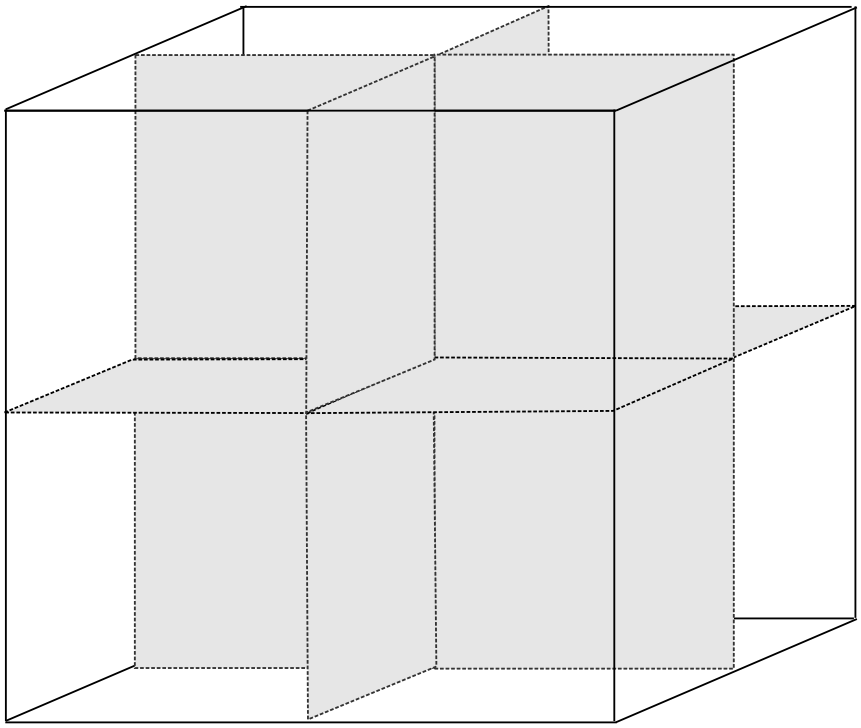


Figure 2.4: STC - Spatial Twist Continuum.

The next technique described here is considered to be the most robust approach to automatic hexahedral meshing up-to-date. This is the so-called *overlay* or *grid-based* approach [36, 105–108, 119, 126]. In this approach, a Cartesian grid that encompasses the entire domain of interest is generated. This grid is refined non-conformally based on characteristic sizes of geometrical features using an Octree technique. After that, all cells generated by refinement and located outside the domain are removed from the mesh. Surface of the remaining grid is connected to domain boundaries and layers of regular elements are inserted along the boundaries for quality reasons (see Figure 2.5). While being robust, this method tends to create highly distorted or even inverted elements. Therefore, powerful quality improvement tools must be integrated into those methods that employ the grid-based approach. Another well-known problem of this technique is that resulting meshes are extremely dependent upon orientation of the initial overlaid grid with respect to domain geometry (see Figure 2.6).

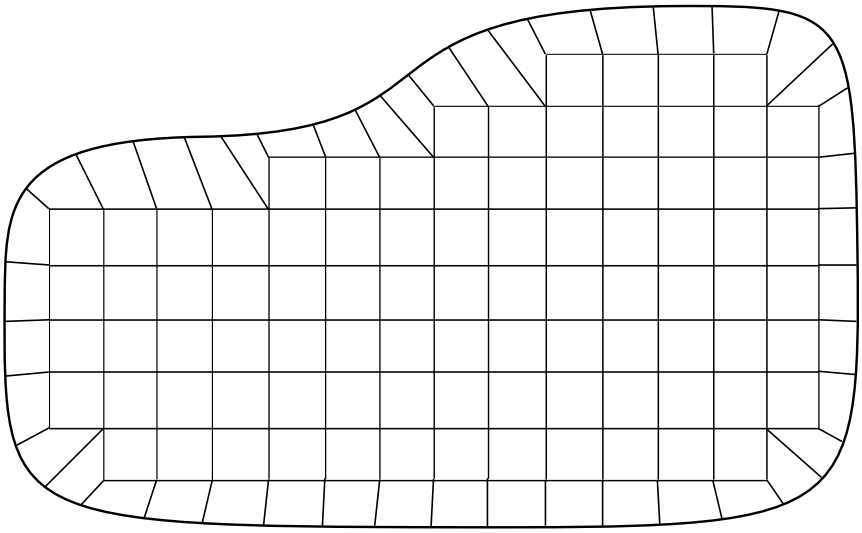


Figure 2.5: Grid-based approach.

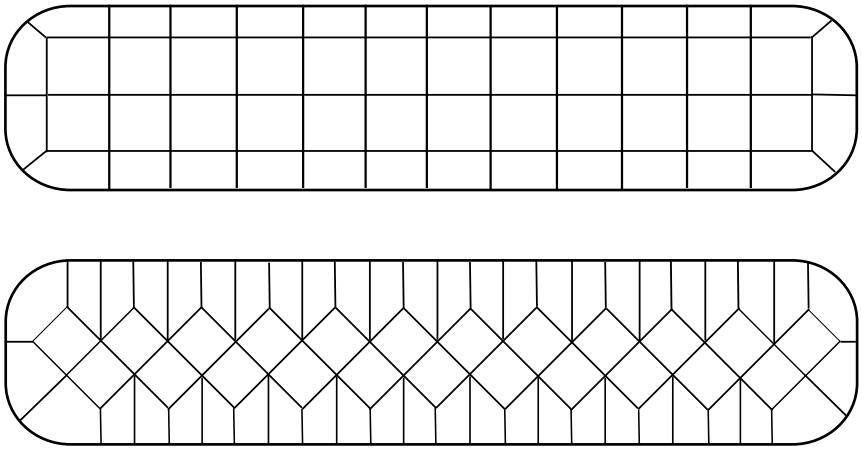


Figure 2.6: Dependency of final mesh upon orientation.

A few words can be devoted to the so-called *indirect* methods [84, 116]. In these methods, a conformal tetrahedral or hybrid hex-dominant mesh is first generated in the domain, after which the mesh is transformed into a fully hexahedral one via subdivision of each non-hexahedral element into hexahedra (see Figure 2.7).

The present work has been performed within framework of an approach that employs the grid-based technique [36].

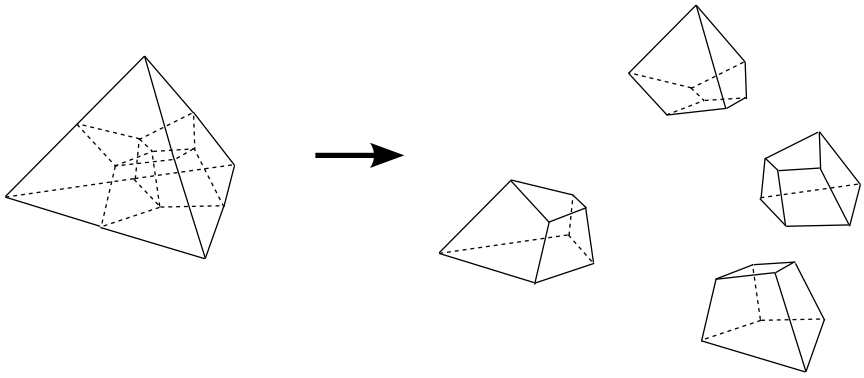


Figure 2.7: A tetrahedron subdivided into four hexahedra.

2.2 Mesh quality improvement

It is common for unstructured mesh generation methods to produce elements which are of poor quality or even invalid (concave or inverted). In general, such elements degenerate accuracy of computations. Hence, invalid elements should not be allowed to exist in final meshes, while quality of valid elements should be improved as much as possible.

Recent industrial trends require mesh generation tools to be as automated as possible in order to be competitive. Most automated unstructured mesh generation procedures usually do not apply thorough analysis and prediction of mesh quality. Therefore, obtaining an unstructured mesh with well-shaped elements requires posteriori procedures that directly address the issue of mesh quality.

Generally speaking, there exist two main possibilities to improve quality of mesh elements: modification of mesh connectivity to allow improvement of quality of elements involved (*topological cleanup*) and direct modification of nodal positions (*smoothing*). The two following sections provide a detailed description of state-of-the-art techniques of these two directions. It should be noted that application of topological cleanup to hexahedral unstructured meshes is still a topic under investigation. Therefore, the emphasis will be placed on the variety of smoothing methods widely employed in unstructured meshing nowadays.

2.2.1 Topological cleanup

The term mesh *connectivity* denotes a system of edges, faces and cells which are formed based on a given set of vertices. It is also referred to as mesh *topology*. First of all, it is important to understand how topological modifications can affect mesh quality. Improvement of mesh quality via node movement is possible only up to a certain quality limit. This limit depends on topology of the mesh. Figure 2.8 shows an example in which mesh quality cannot be improved unless its topology is modified. Namely, substitution of a five

cell pattern by a four cell one results in an increase of the minimum angle of the mesh up to 90°.

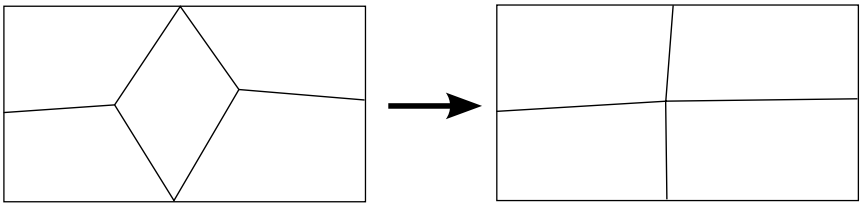


Figure 2.8: Mesh quality improvement via modification of its connectivity.

Topological modifications are applied most successfully in triangular and tetrahedral mesh generation systems [5,6,9,16,17,20,45,49–51,110,121]. The so-called edge *swapping* or *reconnection* is effectively used to improve triangular mesh topology (see Figure 2.9). Similar operations are available in tetrahedral meshing procedures. They have been successfully used in many tetrahedral mesh generation systems as, for instance, in [45]. As shown in Figure 2.10, tetrahedral reconnectable sets of elements are somewhat more complicated; nevertheless, they can easily be classified and detected.

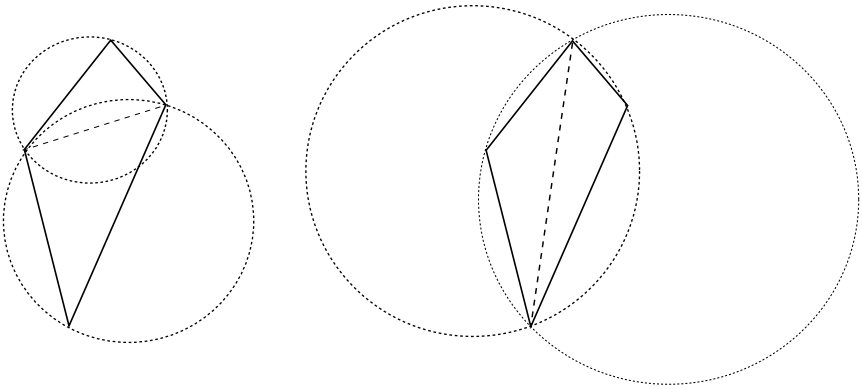


Figure 2.9: Delaunay and non-Delaunay triangulations.

There exist various criteria for concluding whether an edge or a face must be reconnected or not. The most popular one is the so-called *Delaunay condition* extensively employed for generation of *Delaunay triangulations* [5, 6, 9, 16, 17, 37, 39, 50, 110, 121]. According to this condition, the *circumcircle* of a triangular cell or the *circumsphere* of a tetrahedral cell cannot contain any mesh nodes within itself. This criterion is illustrated in Figure 2.9. The condition provides a precise answer whether an edge must be reconnected or not. Delaunay condition does not address the issue of element quality directly. However, it can be shown [110] that quality of any *simplicial* (triangular or tetrahedral) mesh that satisfies this condition is not lower than a certain threshold.

Alternatively, control of nodal *degree* or *valence* (the number of edges connected to a node) can be used as a reconnection criterion [47]. Its optimal value is defined as the valence of

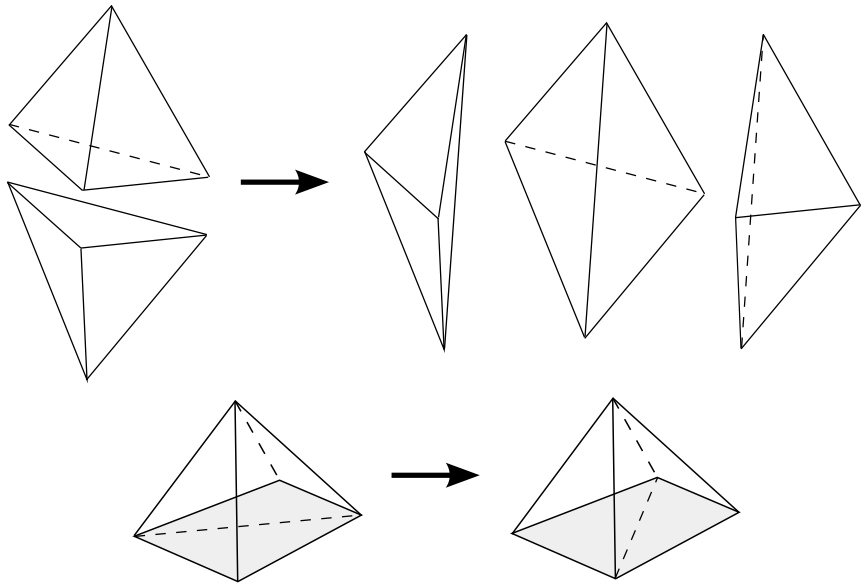


Figure 2.10: Tetrahedral reconnectable sets.

a node in a mesh which consists of ideally shaped elements, namely equilateral triangles or tetrahedra (Figure 2.11). Non-optimal valence of the node indicates that the quality of surrounding elements cannot be optimal. Hence, reconnection is performed if and only if it results in valence improvement of the involved nodes.

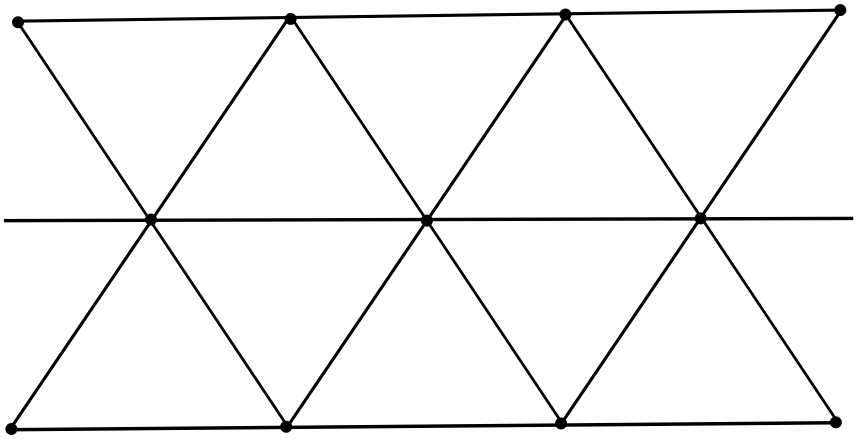


Figure 2.11: Mesh relaxation.

Obviously, these examples do not limit the variety of criteria for topological modifications, and many others can be employed. It is often beneficial to use criteria customized for a particular problem of interest, or to combine several different criteria together in order to obtain better results.

In contrast with simplicial meshes, adapting topological cleanup to quadrilateral and hex-

ahedral meshes turns out to be a difficult task. Among the few attempts to tackle it, the 2-D cleanup procedure for quadrilateral meshes by Kinney [69] shows practical results. However, the 3-D case appears to be much more complicated. Only some classification and feasibility conditions have been established so far [11]. Performance and all possible practical benefits of hexahedral cleanup are still to be investigated, but that task is beyond scope of present work.

2.2.2 Smoothing

Smoothing can be defined as relocation of mesh nodes at constant connectivity. Despite the recent evolution of methodology of unstructured mesh quality improvement, smoothing still remains the most common approach to quality improvement of non-simplicial meshes. There exist multiple ways of defining new positions for relocated nodes. These can be sorted into the following categories:

- Laplacian smoothing and its variations.
- Optimization-based smoothing.
- Physics-based smoothing.
- Combining different techniques.
- Untangling.

Laplacian smoothing represents the most common family of methods among those mentioned above. The simplest form of Laplacian smoothing places each mesh node successively at the average position of nodes connected to it. This was implemented and extended by many research groups [39, 41, 45, 49, 50, 54, 64].

The technique works well in convex regions, but it may produce poorly-shaped or even inverted elements in concave ones. Typically, the mesh surrounding concave geometrical items is pulled outwards. This often leads to geometry-overlaying meshes. A simple illustration of this behavior is depicted in Figure 2.12. Due to excessive distortion, such meshes are usually quite difficult to recover via posteriori quality improvement tools. Therefore, application of this approach in its original form to meshes generated for arbitrary domains is generally not efficient. Additional controls are necessary to prevent elements from being distorted during smoothing.

For instance, the *constrained Laplacian smoothing* restricts nodal displacement to a certain limit in order to avoid element distortion. A simple form of this approach limits nodal movement equally in all directions: if the computed node displacement is smaller than a certain pre-defined threshold, this displacement is applied directly; otherwise, the node is moved by the threshold distance in the same direction. In a more advanced implementation,

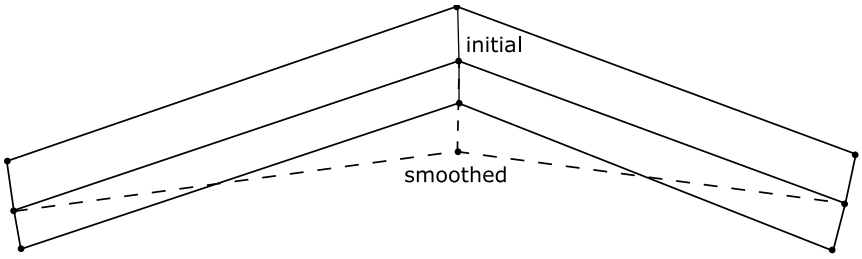


Figure 2.12: Geometry-overlaid meshes produced by a simple Laplacian smoothing.

quality of elements affected by node movement is computed using both old and new node locations. The node is then moved only if the quality of all affected elements does not decrease after the motion. This method is referred to as the *"smart" Laplacian smoothing* [26, 41].

Another example of a modified Laplacian smoother can be found in [90]. Here, the advantage is taken from local action of Laplacian smoothing applied to unstructured meshes. Topology of a mesh subject to smoothing is analyzed and the most appropriate smoothing schemes are applied based on the result of the analysis. For instance, if exactly eight hexahedral cells surround a node of a hexahedral mesh (as in structured grids), then the so-called *equipotential (Winslow) smoothing*, an approach for smoothing of structured grids, is applied. Earlier, Knupp [70] developed a 2-D version of this smoother applied to quadrilateral unstructured meshes.

When applied to unstructured hexahedral meshes, a Laplacian smoother can be involved in a quality improvement process, but it cannot be a standalone quality improvement tool due to limited efficiency.

The *optimization-based smoothing* [25, 26, 30, 41, 43, 44, 46, 58, 59, 61, 63, 95] moves mesh nodes to minimize a certain cell quality metric. Nodal positions are modified based on analysis of variations of local mesh quality, unlike in heuristic approach used in Laplacian smoothing. The optimization-based approach is more reliable, especially in concave regions of computational domain. However, the computational cost of this approach is obviously higher than that of the Laplacian methods.

Essentially, the difference between various versions of this family of methods consists in choice of the distortion metric. There exist numerous ways of defining the metric that expresses the distortion of mesh elements. De Cogny *et al* [32] proposed one of the first optimization-based smoothing approaches. The method was applied to improve quality of tetrahedral elements near boundary. The distortion metric used in this approach represents a scaled ratio of an element's volume to its face areas. In some other approaches [30, 61, 63] a mesh cell is considered as a solid object and its distortion is measured by computing the energy needed to deform a reference element (tetrahedron or hexahedron) such that it has the final shape of the given element. The lower this energy is, the closer the given element is to an ideal one. Therefore, minimizing this deformation energy for a set of elements will

result in a global quality improvement for this set.

Recently, many new and challenging ideas in the field of distortion metrics have been developed and introduced into state-of-the-art mesh generation procedures.

Parthasarathy and Kodiyalam [95] suggest using the element aspect ratio as the function to be minimized. In their approach, a global optimization problem is solved, and an additional constraint is enforced onto volumes of elements (areas in 2-D), which are not allowed to drop below a certain threshold.

Canann *et al* [25] has implemented another optimization-based smoothing algorithm that employs Oddy's metric for isoparametric elements [93]. This metric was constructed for hexahedral elements but it can be modified for optimization of tetrahedral meshes as well. However, the metric was found to be excessively restrictive on element's aspect ratio distortion and, on the contrary, insufficiently restrictive on element's angle distortion.

The two latter approaches employ global optimization and, therefore, are impractical for large meshes. Local successive smoothing mode is shown to be more feasible.

Freitag *et al* [42, 45] suggested using the minimum angle of a triangle or the minimum dihedral angle, the minimum solid angle or the minimum aspect ratio of a tetrahedron as quality metrics for optimization-based mesh smoothing algorithms. These quality metrics are generally non-smooth functions of vertex coordinates, and the corresponding solution procedures for non-smooth functions are employed. The approach demonstrates good efficiency on simplicial meshes and is suitable for parallelization.

Calvo and Idelsohn [23] developed a node-based quality metric for all-hexahedral meshes. Each hexahedron attached to a mesh node has exactly three edges that reference this node. Three unit vectors emanating from the node along these edges form a *trihedron*. Its quality measure contributes to the cumulative quality measure associated with this node.

Knupp developed a new approach to quality metrics for tetrahedral elements based on tetrahedral Jacobian matrix norm [71, 72]. He continued his research in this direction and together with Freitag [43, 44] developed an approach that uses the so-called condition number as a distortion metric for tetrahedral elements. This quality measure is based on analysis of shape transformation between an ideal and a physical tetrahedra. As mentioned above, the condition number is based on Jacobian matrix norm of the transformation. This measure does not depend on element orientation and size, but only on shape distortion with respect to an ideal element. The ideal element can be chosen based on a certain set of requirements to element shape that must be satisfied in an optimized mesh. Later, Knupp had extended this approach to hexahedral framework [74]. Decomposition of a hexahedron into tetrahedra is used and all the constituting tetrahedra are optimized.

Generally more computationally expensive, the optimization-based smoothing methods are preferred over the Laplacian methods because the former do not create inverted or non-convex elements. Combining these methods may result in efficient and fast quality improvement tools. This possibility is discussed further in this section.

The *physics-based smoothing* is based on the idea that a well-shaped mesh can be viewed as an analogue of certain mechanical systems that can be found in nature. Some authors have developed techniques that smooth unstructured meshes using principles which drive dynamics of such systems in nature.

A classical example of one of such techniques is the so-called *spring analogy* approach [10,82]. In this method, edges of a mesh are considered as springs with stiffnesses depending on their target sizes. A mesh node is repositioned to a new location to bring the set of surrounding springs to an equilibrium. All mesh nodes are successively repositioned until the entire spring system reaches equilibrium. Forces occurring in springs can be unidirectional or bi-directional. In the first case, magnitude of a force depends on spring's length but its direction is always the same. In the second one, equilibrium spring length can be defined. A force changes its direction depending on the actual edge length with respect to its equilibrium length.

Another group of methods which has been developed recently and which employs the analogy with mechanical spring systems uses the so-called *torsional springs* [33,38]. While regular linear springs resist changes in internodal distances, the torsional springs resist changes in the angle between edges incident to same node. When the angle between such edges deflects from equilibrium, a corresponding torsional spring generates torque directed to bring the system back to equilibrium (see Figure 2.13).

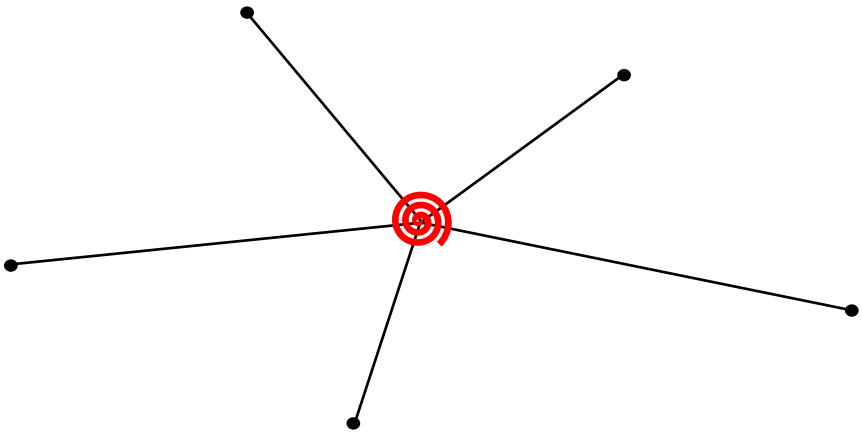
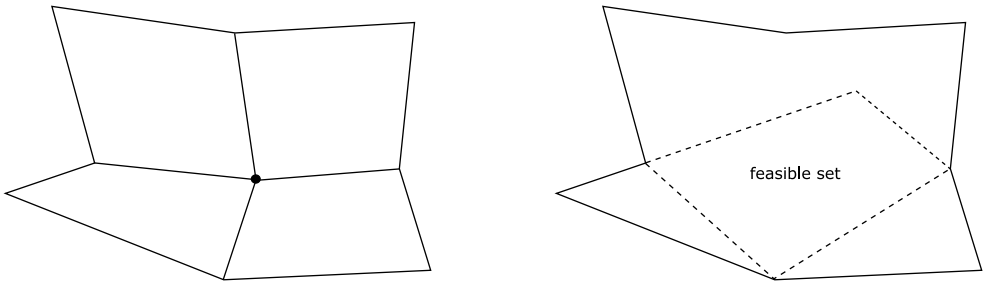


Figure 2.13: Torsional spring analogy.

The optimization method that uses deformation energy as the quality metric [30, 61, 63] can also be considered as a physics-based smoothing approach from a certain point of view.

The *combined approaches* usually represent a blend of optimization-based and Laplacian smoothing methods. Development of such methods is aimed at combining the speed of Laplacian approach with the efficiency of optimization-based methods. In these methods, it is important to develop a system of criteria to effectively distinguish whether a Laplacian or an optimization-based smoothing method should be applied to a mesh node. Examples of combined approaches can be found in [26, 41].

The *untangling* has recently received greater attention in research activities in the area of unstructured mesh quality improvement. The term untangling refers to the process of elimination of tangled (concave) elements from a mesh. It is applied to those mesh nodes, which represent concave corners of tangled cells. Current position of such a node is clearly unsatisfactory as it creates at least one concave (i.e. invalid) mesh element. Therefore, a new position which corrects all tangled cells adjacent to the node while preserving validity of the rest should be found. The process of untangling derives this position directly from the concavity/convexity criteria applied to each affected cell. In other words, *the feasible set* or *the kernel* (a complete set of points satisfying convexity condition for all surrounding cells) is found and the node is placed somewhere within this region (see Figure 2.14). For instance, Freitag and Plassman [46] solve the problem of maximization of minimum area/volume of elements attached to the node being moved. Knupp [73] uses untangling based on optimization of an objective function constructed using determinants of Jacobian matrices of elements surrounding the node.



(a) Cells around a node. (b) The kernel of a node.

Figure 2.14: Untangling.

A new combination of Laplacian and optimization-based smoothing methods with an untangling approach constitutes one of the topics of the present thesis.

2.3 Viscous mesh generation

Up-to-date techniques for viscous flow analysis require high quality of computational meshes in regions where viscous effects are essential. These requirements include smoothness of cell sizes of viscous grids, control of position of the first mesh point close to wall (depending on type of turbulence model [87, 112]), limitation of stretching ratio h_{i+1}/h_i that must range within certain values (in the work by Spalart [112] it ranges between 1.2 and 1.3).

Ever since viscous computations became possible, various viscous mesh generation approaches have been developed. In case of boundary-conforming structured grids, the only

difference between viscous and inviscid regions is the grid spacing in normal-to-wall direction. Unfortunately, structured grids tend to contain too many cells in regions where it is not necessary, which is typically due to the limited ability of generators to produce grids with strongly non-uniform cell size distribution. In order to generate more cells in a certain region, new grid lines should be introduced along the entire computational domain in corresponding direction, which usually results in overpopulation of cells in regions where it is not desired. Block-structured methods can resolve this drawback to some degree but they still provide limited flexibility within one block. Therefore, up-to-date unstructured methods require new approaches for viscous meshing.

Essentially, families of approaches that address the issue of high quality anisotropic mesh generation can be classified as follows [48]:

- Direct generation of anisotropic meshes by combining isotropic meshing methods with domain transformation.
- Transformation of initial isotropic mesh by node repositioning and local mesh modifications.
- Building anisotropic elements by special methods (advancing layer methods) and isotropic meshing of the rest of the domain.

The first group of methods related to direct generation [18,19,28,29,50] had been attempted together with the Delaunay and advancing front approaches to triangular and tetrahedral (simplicial) unstructured mesh generation. Due to their nature, these approaches produce isotropically graded meshes by default. Certain modifications of algorithms are required in order to allow for generation of highly stretched elements.

Mavriplis [86] generated a metric field consisting of stretching vectors specified at each point. This metric is used to map an existing computational domain to a control domain, where an isotropic mesh is generated via the Delaunay method. Backward conversion of this mesh into original space results in an anisotropic mesh in the domain.

Borouchaki *et al* [18,19,50] generalized the idea of anisotropic mesh generation via Delaunay method based on the metric specification. Castro-Diaz *et al* [28,29] extended this approach by recognizing that the mesh orthogonality near the wall as well as the fixed first cell size are of prime importance for successful viscous computations. The metric was modified in the near-wall region to take these criteria into account.

This family of methods, however, generates purely simplicial meshes. This restriction poses a problem with element quality, as presence of low angles is unavoidable. Besides, these meshes contain too many elements with respect to hexahedral or hybrid meshes with prismatic layers.

The approach of modification of initial isotropic mesh is well represented in work by Hassan *et al* [55]. This method is based on the advancing front technique. Layers of cells are

generated along an advancing front using isotropic criteria. After generation of another layer, its elements are compressed towards boundary via node repositioning to result in anisotropic elements along the wall. The procedure is repeated until the required number of anisotropic layers is generated along the boundary. After that, the rest of the domain is meshed using a standard advancing front technique. Although this method shows good results in 2-D, its 3-D version is prone to problems. One of the problems is the fact that the method tends to generate more points on the surface of the first layer than on the boundary, thus resulting in degraded mesh quality after node repositioning.

The third group of methods for viscous mesh generation consists in generation of layers of anisotropic cells along boundaries prior to volume isotropic meshing. The layers are generated based on certain criteria. As the next step, the rest of the domain is meshed isotropically using one of the available techniques. Methods of this family are often referred to as the advancing layer methods [100].

Kallinderis *et al* [65–67] have developed a method that generates layers of prisms around a body. The height of prisms increases with distance from the wall according to user-specification. After specified layers are generated, the remaining domain is meshed using a combination of Octree and advancing front techniques. Sharov and Nakahashi [109] suggest a similar approach but with some modifications to produce higher quality cells. Their method generates purely tetrahedral meshes. Generally, use of prisms produces less elements. Problems with element quality near sharp corners appear to be common for this family of methods.

Lohner [80] also uses a similar technique. In his method, layers of tetrahedronized (subdivided into tetrahedra according to a predefined pattern) prisms are generated along a surface, after which an advancing front method is used to mesh the rest of the domain. This method is equipped with powerful quality control and improvement tools. However, in his later work [81], Lohner advocates use of anisotropic refinement of isotropic mesh using the Delaunay criterion. Inability of the advancing layer methods to deal with complex geometrical models is the reason for focusing on a new technique.

Pirzadeh [100] proposed a similar approach called the *Advancing Layer Method* (ALM), in which layers of anisotropic prisms are generated along boundaries. A certain procedure is applied to provide consistent diagonalization of prisms. It is necessary that prism diagonals are consistently related to each other in order to allow for subdivision of a prism into tetrahedra. An example of indivisible prism is shown in Figure 2.8.

Finally, Garimella provided a generalization of the advancing layer methods applied to the problem of anisotropic tetrahedral mesh generation in his PhD thesis [48].

The issue of generation of unstructured hexahedral meshes for viscous flows has not yet been addressed widely in literature. Quite simply, hexahedral methods for unstructured automated meshing are relatively young and still pose many problems related to isotropic meshing.

Research presented herein addresses the problem of anisotropic unstructured hexahedral mesh generation for viscous flow applications.

2.4 Deforming meshes

Numerous problems of computational fluid dynamics dealing with moving and deforming meshes became of great interest in recent years. The need to fit a mesh into new borders is frequently encountered in various problems that involve domains with moving boundaries. There exists a wide range of methods developed for successful solving of such problems. Most of them can be split up into two major groups:

- Methods for meshes with variable connectivity address the problem of mesh deformation by means of simple interpolation of boundary motion followed by mesh refining/coarsening. The latter is applied for the sake of preserving mesh quality as well as nodal density necessary for capturing certain physical phenomena. Methods of this family are not generally applicable to hexahedral meshes due to invariant connectivity of such meshes.
- Methods for meshes with constant connectivity employ more sophisticated interpolation algorithms. These algorithms must be capable of preserving mesh quality during interpolation process as no coarsening is generally allowed for this type of meshes. These methods are common within structured and hexahedral frameworks.

Methods for moving or deforming triangular or tetrahedral meshes [7, 8] are inapplicable to similar problems that require other types of meshes. The reason is that methods for tetrahedral mesh generation extensively employ operations of mesh refinement and coarsening via inserting and removing mesh nodes and retriangulating respective voids with valid triangles or tetrahedra. These operations allow using basic interpolation algorithms for repositioning of mesh nodes without enforcing strong quality control. Instead, refinement and coarsening are applied to a mesh with displaced nodes to eliminate inverted elements. Particularly, any internal mesh node being a vertex of at least one inverted element is removed from the mesh until no inverted elements remain. After that, additional refinement can be applied in certain regions in order to satisfy nodal density requirements. However, as operations of mesh refinement and coarsening are limited to triangular and tetrahedral frameworks, generally they are not applicable to unstructured hexahedral meshes.

Another wide range of methods for moving and deforming meshes is based on requirement to preserve both the mesh connectivity and set of nodes. Such limitations deny any possibility to apply partial re-meshing. The general approach to the problem of deformation of meshes with constant connectivity includes two stages. The first one consists of interpolation of deformation from domain boundaries into volume. At the second step, mesh quality recovery is performed: shapes of elements that became invalid or excessively distorted undergoes improvement.

Several conventional mesh deformation approaches are reviewed by Wick [124]. Some of them are described in this section along with a few front-end methods developed in recent years.

In the general framework, a prescribed boundary deformation can be extended into the volume mesh in a smooth manner, thus providing a displacement vector for each internal mesh node:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} u \\ v \\ w \end{pmatrix} \tag{2.1}$$

Here, (X, Y, Z) are the initial undeformed positions; (x, y, z) are the final deformed positions. This expression is applied to each mesh node once distribution of displacements (u, v, w) over the meshed volume is known.

In the *Least-squares formulation*, such a distribution of nodal displacements must be found that preserves angles between mesh edges. Such a distribution corresponds to orthogonal mapping. Conformal mapping between the deformed and the undeformed spaces (X, Y) and (x, y) is given by the Cauchy-Riemann equations:

$$\frac{\partial x}{\partial X} - \frac{\partial y}{\partial Y} = 0, \ \frac{\partial x}{\partial Y} + \frac{\partial y}{\partial X} = 0 \tag{2.2}$$

However, proper boundary conditions for these equations cannot be established, as boundary deformation is prescribed for a general case. Therefore, the exact solution cannot be found. Instead, a least-squares problem can be formulated to satisfy the equations as precisely as possible. In this case, the following functional must be minimized in order to obtain a mesh that best obeys the initial requirements:

$$I = \int \left((u_X - v_Y)^2 + (u_Y + v_X)^2 \right) \, dX \, dY \tag{2.3}$$

This can be done by means of any standard optimization approach described in linear algebra.

Alternatively, the *solid-body elasticity approach* introduced by Lynch and O'Neill [83] more than 20 years ago, considers a mesh as a solid object with linear elasticity. Mesh deformation is described by a functional that reflects the strain energy saved in a body:

$$I = \int \left(u_X^2 + (u_Y + v_X)^2 + v_Y^2 \right) \, dX \, dY \tag{2.4}$$

The minimum of this energy corresponds to the position of partial equidistribution of strain induced by boundary motion. A detailed description of this approach can be found in [123]. This approach can be considered as a particular case of a more general structural analogy [62, 125].

The *spring analogy model* is by far the most popular method for solving the problem of mesh deformation. According to this approach, connectivity between mesh vertices is

treated as a spring system. Namely, each mesh edge is represented by a mechanical spring with a certain stiffness. Therefore, the energy of mesh deformation is contributed by each edge-spring. The functional which controls deformation can be expressed as follows:

$$I = \sum K_{ij} \left((u_i - u_j)^2 + (v_i - v_j)^2 \right) \tag{2.5}$$

Here, K_{ij} denotes a spring's stiffness. The shorter one of the mesh edges is, the stiffer the corresponding spring must be. The general relation between spring stiffness K_{ij} and length L_{ij} of an edge connecting vertices i,j can be expressed as follows:

$$K_{ij} = L_{ij}^M, \; M < 0 \tag{2.6}$$

The original work by Batina [10] proposes to use $M = -1$. However, if deformation is large enough to produce self-intersecting meshes, M should be less than -1 in order to prevent nodes from crossing edges thus producing invalid elements [14].

The *torsional spring analogy* [33,38,68] is another alternative approach to mesh deformation or smoothing. The method uses torsional springs to control angles between edges incident to same mesh node. These approaches were mentioned in Section 2.2.2. Description of algorithmic details can be found in [38].

Among front-end developments which emerged in this area in the recent years, the following few approaches deserve attention.

Helenbrook [56] proposed to apply the so-called *biharmonic operator* to the problem of mesh deformation. The idea behind this method is as follows. The conventional way to propagate deformation from mesh boundary to volume using a differential operator, is to apply the Laplacian operator to coordinates of mesh nodes:

$$\bigtriangledown^2 X = 0 \tag{2.7}$$

The boundary condition for the resulting system of second order differential equations can either be the coordinates of boundary nodes that experienced deformation (the *Dirichlet* condition) or the first derivative of these coordinates in direction normal to boundary (the *Neumann* condition). The latter corresponds to specified mesh cell sizes in normal direction. However, it is impossible to respect both conditions using a second order operator. It means that only displacements of boundary points can be obeyed. Biharmonic operator is the fourth degree differential operator similar to the Laplacian:

$$\bigtriangledown^4 X = 0 \tag{2.8}$$

With this operator, both the new coordinates of boundary nodes and normal cell sizes can be respected. Some results of application of this method to mesh deformation problems are discussed in details by Helenbrook [56]. The main disadvantage of this approach is

that the solution for the coordinates is not necessarily bounded by boundary values and for very large deformations the mesh may become self-overlaid.

Quaternions. Recent investigations in the fields of fundamental and computational mathematics provide new and powerful tools that can be applied to the problem of mesh deformation. One of such frontline methods is application of quaternions for propagation of a boundary deformation into a volume mesh. Quaternions [104] represent generalized complex numbers (*hypercomplex* numbers) composed of one real and three imaginary terms:

$$Q = q_0 + q_1 \cdot i + q_2 \cdot j + q_3 \cdot k \tag{2.9}$$

where i, j, k are such that

$$\begin{gathered} ii = jj = kk = -1 \\ ij = -ji = k \\ jk = -kj = i \\ ki = -ik = j \end{gathered}$$

Quaternions represent a very powerful mathematical tool for description of rotation in mechanics, computer graphics and other areas. The three imaginary components represent a vector along the axis of rotation and the real component represents the angle of rotation.

A combination of quaternions and translation vectors can be used to denote an elementary mesh deformation without losses of information about rotational component. In conventional methods for mesh deformation, boundary deformation is described only by distribution of displacements of boundary points. However, any general deformation can be decomposed into a combination of elementary deformations such as translation, rotation, skew, twist etc. Usually, modification of shape of mesh elements such as skew or twist is negligible and, therefore, can be omitted. On the contrary, loss of information concerning rotation may often result in a significant decrease of quality of deformed meshes. Quaternions enable to preserve information not only about translation of mesh elements but also about their rotation. General information about properties of quaternions, as well as details of their application to mesh deformation can be found in [104].

Martineau and Georgala [85] proposed a method for deformation of unstructured generalized meshes for viscous computations. This approach extends the standard techniques for deformation of isotropic meshes and represents another form of the spring analogy approach. The latter was improved to deal with anisotropic meshes. Particularly, a Laplacian smoothing procedure is modified to successfully smooth meshes with highly stretched elements; an initial solution generation procedure which is based on distance functions is modified to account for interference of deformations induced by different boundaries; rotational components are accounted when computing volume mesh deformation; a dual mesh is used in order to avoid complications related to presence of hanging nodes. For more details, one should address the original work.

In the present work, a combined methodology based on the two latter approaches [85, 104] is developed to address the problem of efficient mesh deformation within unstructured hexahedral non-conformal framework.

2.5 Conclusions

Nowadays, large scale use of unstructured hexahedral meshes for *Computational Fluid Dynamics* (CFD) applications is usually limited by quality of these meshes. Obviously, hexahedral meshes are a good choice for accurate numerical analysis. However, taking the full advantage of them requires powerful mesh generation tools capable of producing meshes with sufficiently high quality of elements.

Up-to-date tools for hexahedral unstructured meshing which are available in the industry do not provide reliable framework for automated hexahedral mesh generation. Various software packages are usually capable of successful mesh generation for a certain (sometimes relatively wide) class of problems. However, meshing domains of truly arbitrary geometry still represents a great challenge. Particularly, mesh quality remains one of the most critical issues. No optimal approach for posteriori quality improvement of hexahedral meshes generated for arbitrary geometry has been created so far.

Computational meshes for CFD can be divided into two main classes. Meshes for inviscid flow analysis are used for discretization and solution of systems of *Euler* equations, while *Navier-Stokes* equations are discretized and solved using meshes for viscous computations. Whereas Euler meshes are isotropic and their cell sizes do not depend on certain direction, viscous meshes include anisotropic regions where highly stretched cells are generated along boundaries for accuracy reasons. These two types of computational meshes require different criteria for quality estimation and, consequently, different methods for its improvement. For inviscid meshes such methods address shapes of elements, while for viscous meshes node distribution near boundaries should be controlled as well. Besides, computational time consumed by quality improvement procedures is to be minimized in both cases.

In the present work, a tool suite for improvement of quality of unstructured hexahedral meshes for both inviscid and viscous computations has been developed. It includes a combined untangling and optimization tool for inviscid meshes, as well as a procedure for generation of viscous meshes of high quality. The latter uses an optimized Euler mesh as the starting point.

Furthermore, a procedure for deformation of unstructured hexahedral meshes has been introduced. A combined approach based on work by Martineau and Georgala [85] is combined with an idea by Samareh [104] on using quaternions for mesh deformation and reinforced by the quality improvement methodology developed herein.

Chapter 3

Unstructured Hexahedral Non-conformal Mesh Generator

3.1 Introduction

Development of fast and reliable mesh generation tools is a significant step for *CFD* (Computational Fluid Dynamics) on the way to becoming routinely used and appreciated in industrial design environment. Industrial designers constantly deal with modeling of complex flows within domains of complicated geometry. While turbulent flows and real physics modeling still remain research topics, creation of reliable numerical tools capable of accurately solving Navier-Stokes equations in complex domains is necessary. The primary objective is to decrease time spent on a CFD simulation for a new geometry as much as possible, provided that most of the time is spent for the flow modeling itself. Therefore, the time dedicated to mesh generation should be optimized.

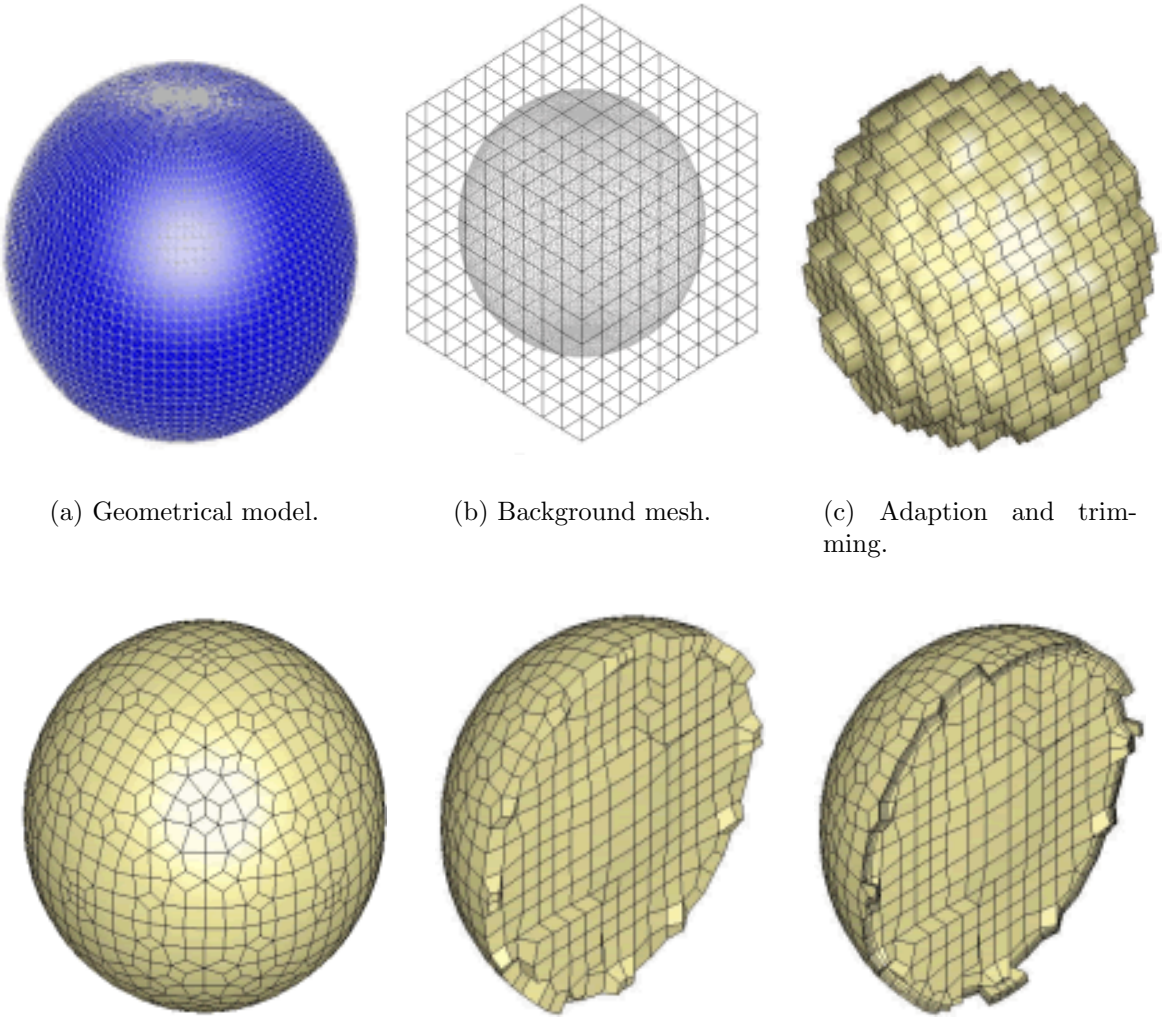
One of the most important aspects of meshing process is importing a *CAD* (Computer-Aided Design) model into mesh generation tools. Industrial CAD models provided for mesh generation are usually poorly defined. Arising issues are related to non-matching *NURBS* (Non-uniform Rational B-Spline) patches resulting in either oveplaps or holes. Generic faults in geometry definition are also possible. Most of these flaws can be attributed either to surface modeling paradigm or to multiple translations of CAD models between various file formats instrumented with different tolerances. A popular solution of this problem is to employ sophisticated CAD repair systems that enable to identify and correct flaws in CAD models. A perfectly clean geometry definition is required by most unstructured mesh generators relying on creation of a surface mesh or *triangulation* prior to generation of volumic mesh. The method described herein represents an attempt to avoid the latter requirement. It employs the grid-based approach [105–108, 126] in which a volume mesh is generated first, while the surface mesh is obtained as a by-product of volume mesh generation. Therefore, small flaws in CAD models can be ignored as long as their characteristic sizes are smaller than the minimum element size of a surface mesh.

Another reason why this approach is chosen is the fact that it allows for an automated generation of boundary-conforming fully hexahedral meshes. Grids of this type still lack popularity due to inherent difficulties of meshing within highly complex geometries. However, they potentially offer a higher accuracy of solution than tetrahedral or hybrid prismatic tetrahedral meshes when using classical numerical methods. Besides, hexahedral elements are the best choice for resolving highly sheared flows such as boundary layers.

The presented approach has been implement in the unstructured mesh generation software package called HEXPRESS^{TM} and developed by NUMECA, International, s.a. [35,36,119]. The approach consists of five main steps. First, a geometrical model of the domain to be meshed (Figure 3.1a) is converted into native format which contains both topological and geometrical information about the domain. Following that, a Cartesian background mesh that encompasses the entire model is generated (Figure 3.1b). It is further refined non-conformally via an Octree technique until cells sufficiently small for capturing relevant flow details are populated throughout the domain. At this point, the mesh is not yet boundary-conforming but distribution of cell sizes already satisfies requirements of the problem. At the next step, all cells of the octree-refined mesh that fall outside the domain or intersect its boundary are removed (Figure 3.1c). Surface of the remaining staircase mesh is projected onto domain boundaries; all topological items such as ridges and corners are captured and regular cell layers are inserted. This step is followed by mesh quality improvement (optimization) procedures. Figure 3.1e shows a mesh cross-section revealing optimized cell shapes inside a volume mesh. Optionally, layers of highly stretched cells can be inserted along solid boundaries if viscous effects in the boundary layer are to be captured during simulation (Figure 3.1f). Following sections present a detailed description of these steps as well as examples of meshes obtained by means of this method. Conclusions regarding general performance of the approach bring the Chapter to closure.

3.2 Domain definition

Nowadays, CAD systems are widely employed in the industry for modeling of computational domains which are to be meshed. There exist numerous file formats for description of CAD models. Many of them evolve following progress of respective software packages. In order to separate CAD model definition and mesh generation process, a native file format for definition of the domain to be meshed by the present generator is employed. This format represents a segregated definition of topological features and geometry (see Figure 3.2). Topology of a CAD model can be viewed as its skeleton. It is defined as a set of oriented surfaces, also referred to as topological faces. Each surface is composed of a list of topological edges (ridges). Edges form loops which can possibly be multiply connected. A collection of topological edges forming a topological face is oriented in such a way that the surface of a CAD model remains on the left hand side when a loop is traversed in counterclockwise direction. Orifices in a topological face are described by a clockwise list of topological edges. A topological edge usually has a topological vertex (corner) on both



(a) Geometrical model. (b) Background mesh. (c) Adaption and trimming.

(d) Boundary fitting. (e) Optimization. (f) Viscous layers.

Figure 3.1: Grid-based approach.

ends. It is assumed to be oriented from its origin to its destination. A cyclic topological edge represents a particular type since its origin and destination are one and the same. It may either be a single topological vertex or no vertex at all.

In many cases, the topology definition corresponds directly to respective entities of a CAD model. However, this paradigm allows for the possibility of simplifying the geometry by removing those details which may be redundant for flow simulation as well as cumbersome for grid generation.

The actual geometry of a CAD model includes support of all topological entities. The lowest dimensional entity, a topological vertex, is supported by a single point given by

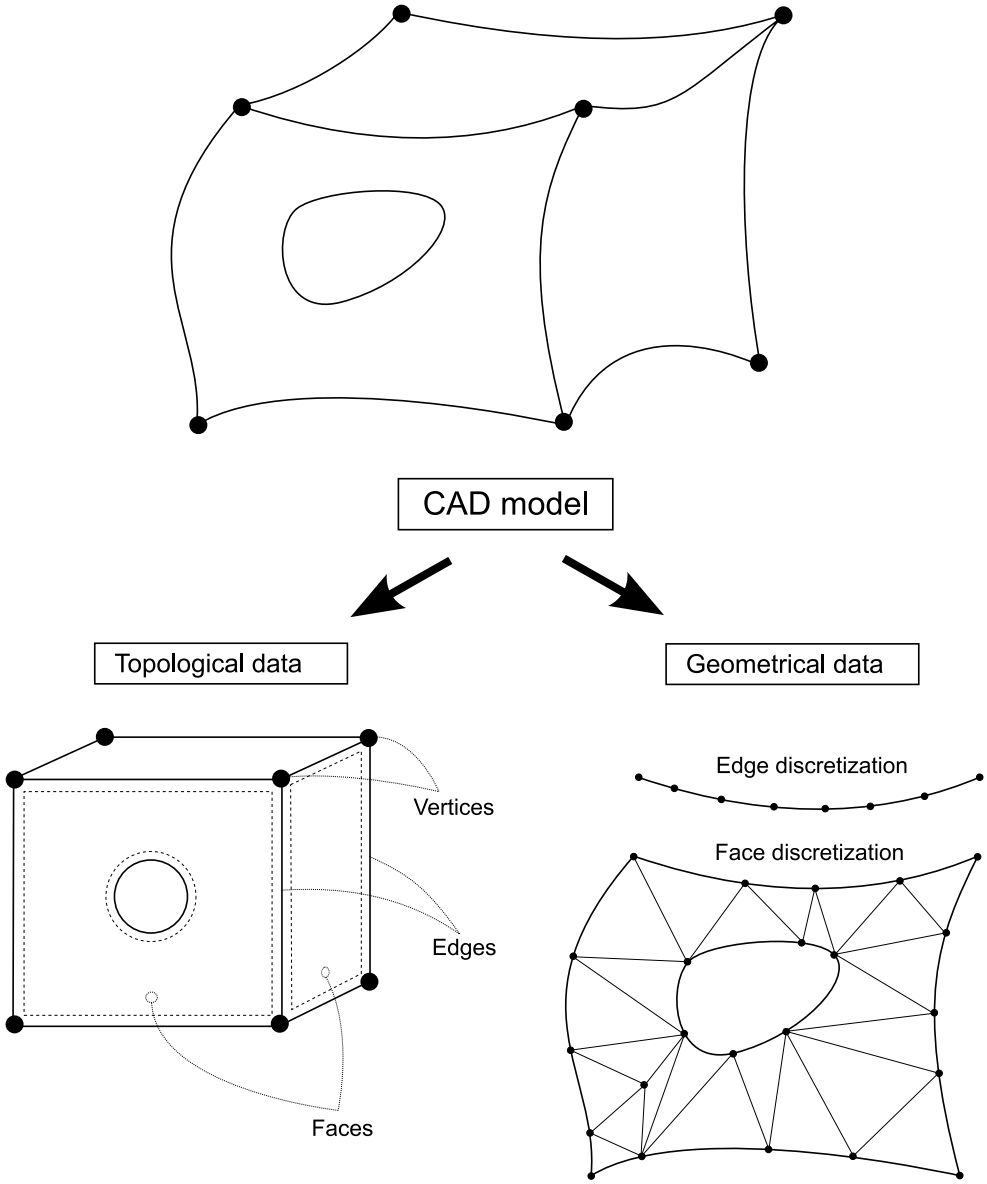


Figure 3.2: CAD model definition.

its three Cartesian coordinates. Topological edges lean on geometric curves represented by a list of connected line segments. Indexing of discretization points is local to each curve. Coordinates of points are provided. Similarly, each topological face is supported by individual triangulation of a locally indexed set of points. Since both edge and surface discretizations employ local indexing, no limitations exist on conformity of neighboring discretizations.

The only objective of triangulation supporting a topological face is to represent the geometry with sufficient accuracy. As the methodology implemented in the generator extensively

employs projections onto domain boundaries, direct use of the NURBS definition for this purpose would lead to excessive computing time. However, it is always possible to project points of the final surface mesh on the actual NURBS definition at the end of the mesh generation procedure.

An example of such a domain is depicted in Figure 3.3. The domain represents a three-pipe joint configuration. This example will be used for further illustration of algorithms involved in the mesh generation process described in this Chapter.

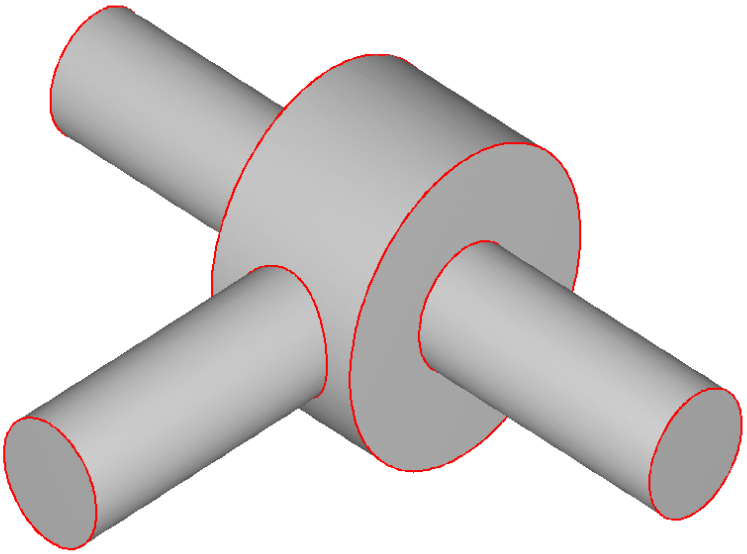


Figure 3.3: Three-pipe joint.

3.3 Initial mesh

According to the grid-based approach, the first step of a mesh generation process is construction of an initial background mesh. For this purpose, a Cartesian box bounding the domain of interest is identified and optionally subdivided into a set of identical cells by refinement along the axes x, y, z. Figure 3.4 illustrates this operation.

Alternatively, any hexahedral mesh fully encompassing the domain can be used. The choice of the initial mesh is large. It can be a structured mesh with orthogonal or non-orthogonal cells, curvilinear or straight, or it can also be an unstructured collection of hexahedra. As highlighted by Tchon *et al* [119], this mesh should be very easy to generate and should fully encompass the respective computational domain.

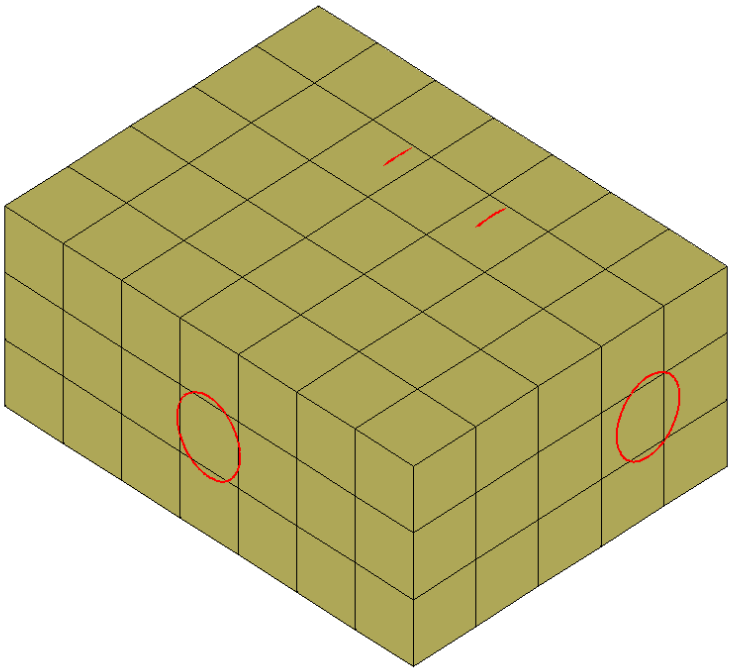


Figure 3.4: Initial mesh generation.

The default behaviour of the generator is to start from an initial mesh involving a single hexahedral cell corresponding to the bounding box of the computational domain.

3.4 Adaptation

At this step, the initial mesh is automatically adapted to particular geometrical features by successive subdivisions of its cells. The goal is to refine the mesh until necessary nodal density and cell sizing are obtained. Further adaptation of the mesh can be performed optionally during flow simulation, depending on some indicators measuring the quality of obtained solution. The available refinement patterns are depicted in Figure 3.5.

The process of refinement typically involves several levels. At each level, mesh cells are either preserved or refined according to one of the patterns in Figure 3.5. All cells intersecting the domain boundary are identified and their sizes are compared to target cell sizes that must be satisfied. The latter can either be specified by user or computed based on characteristic length scales of underlying geometry. The difference in refinement levels allowed between neighboring cells is limited to one. This criterion guarantees smooth cell size gradation between regions with different cell sizes.

A cell is subdivided if at least one of refinement criteria is satisfied (see Figure 3.6). The first criterion involves target cell sizes specified by user for each topological surface, near

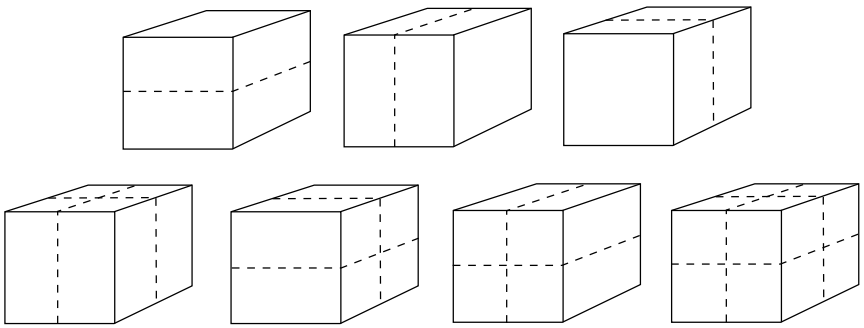
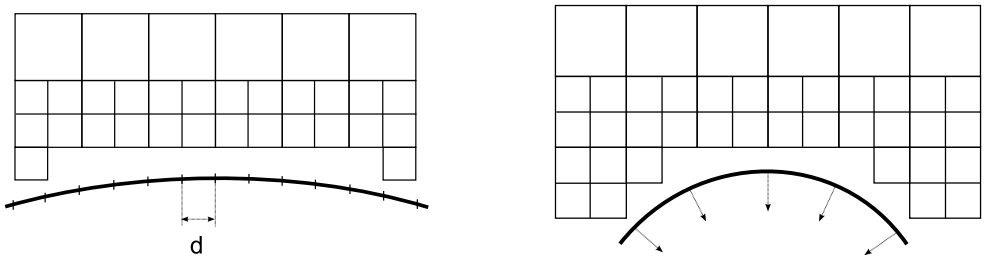


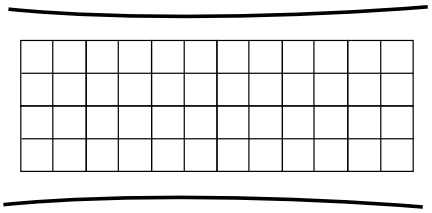
Figure 3.5: Hexahedral anisotropic refinement patterns.

which certain cell sizing is required. During the refinement process, cells intersecting respective surfaces are subdivided until their sizes reach specified values. The other criterion is variation of normals of surface triangles intersecting mesh cells. Cells are refined if this variation exceeds a certain threshold value. Additional refinements are forced in narrow gaps present in the domain. A certain minimum number of cells is generated across such gaps.



(a) Target sizes.

(b) High curvature.



(c) Narrow gaps.

Figure 3.6: The three refinement criteria.

After the refinement process is completed, the final operation of the adaptation procedure is removal of cells that intersect or fall outside the domain boundary. This process is also referred to as *trimming*. Intersecting cells are identified and marked first. After that, cells

located inside the domain are marked by adjacency starting from the cell that contains the so-called *seed-point*. The latter represents an arbitrary user-specified position inside the domain. It is employed specifically for trimming purposes. Finally, both the intersecting and located outside the domain boundary cells are deleted from the mesh. Example of a refined and trimmed mesh is depicted in Figure 3.7.

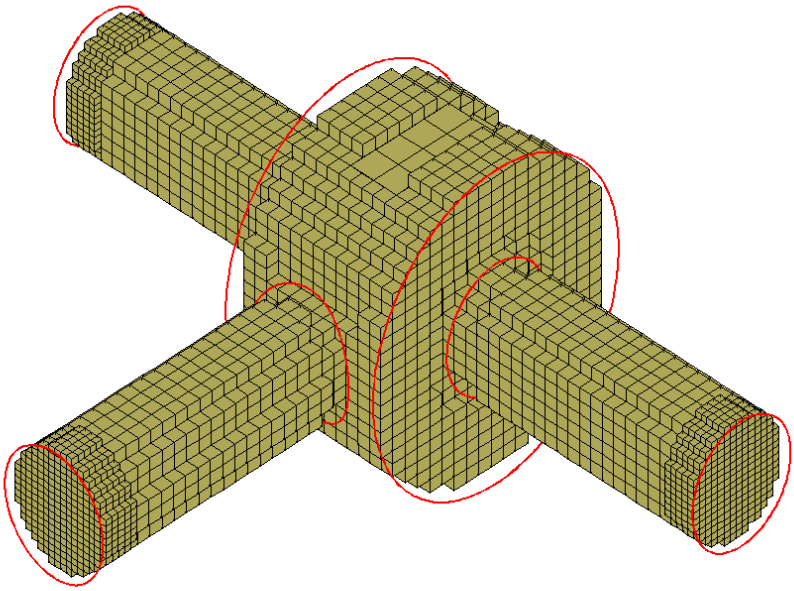


Figure 3.7: A sample adapted mesh.

3.5 Boundary fitting

After the initial mesh is refined and trimmed, the staircase boundary of the volume mesh is snapped to geometry surface via direct projection of surface mesh points onto domain boundary. Prior to this operation, a Laplacian-like smoothing procedure is applied to smooth out the staircase surface mesh in order to facilitate projection and improve quality of the mesh resulting from surface snapping. For this purpose, the surface fairing method by Taubin [117] which includes a finely tuned anti-dissipative step is applied. The method allows avoiding shrinking effects typical for Laplacian smoothing. Projections of surface mesh points onto the surface are computed using the closest distance criterion. However, a location generated by this criterion is accepted only if the difference between the surface normal computed at this location and the normal of the smoothed surface mesh at the respective node is not excessive. This technique prevents a “snap-through” in thin regions

such as trailing edges of wings or blades. Namely, erroneous snapping may project surface mesh points through solid onto its opposite side. After all surface mesh nodes are projected successfully, a connection table between the volume mesh and the geometry is created.

This snapping procedure generally does not guarantee that important geometry features such as corners and ridges are preserved. However, it is important to recover these features for accuracy of physics simulation. To address this issue, special algorithms for low-dimensional feature recovery are applied.

First step is to recover topological vertices. For this purpose, a node of the surface mesh is associated with each topological vertex (see Figure 3.8). This node is placed at the exact location of the associated corner. The closest distance test is usually not sufficient to ensure the resulting good quality mesh in neighborhood of the corner. Several conditions should be satisfied in order for a node to be chosen to capture the respective corner. The mesh node should have a degree (the number of connected edges) not lower than that of the corner (the number of connected ridges) to be associated with; the node should be close enough to the corner; adjacent surface mesh faces should be snapped onto respective topological faces adjacent to the corner. The closest mesh node is chosen as the final alternative if no candidates meeting the requirements are found.

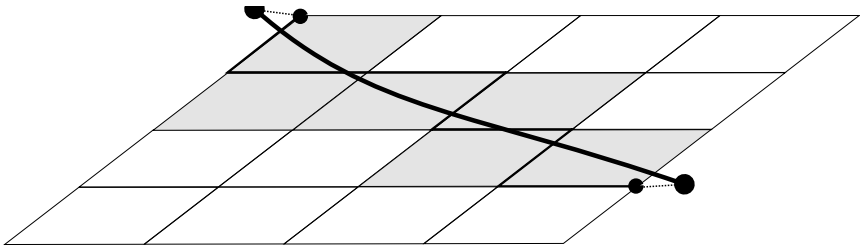


Figure 3.8: Capturing topological vertices and edges.

Explicit capturing of topological edges in a surface mesh consists in associating a list of mesh edges connected by mesh nodes and projecting those nodes onto curves supporting the topological edges. For this to be accomplished, a list of surface mesh faces with nodes projected on neighboring topological faces is compiled (see Figure 3.8). A path joining mesh nodes already associated with both end-corners of a topological edge is sought. Smoothing is applied locally to avoid wrinkling during the projection step.

Some degenerated cells are usually created during both the projection and edge capturing steps. These typically produce angles close to one hundred eighty degrees for certain specific configurations. Figure 3.9 illustrates an example of such behaviour. If two edges of the same surface mesh face are projected onto the same topological edge, degenerate faces appear unavoidably.

To address this issue, a sophisticated buffer insertion strategy, which represents a generalization of the method proposed by Mitchell and Tautges [88], is applied. The methodology is illustrated in Figure 3.10. Buffer layers of cells are created by duplicating surface mesh

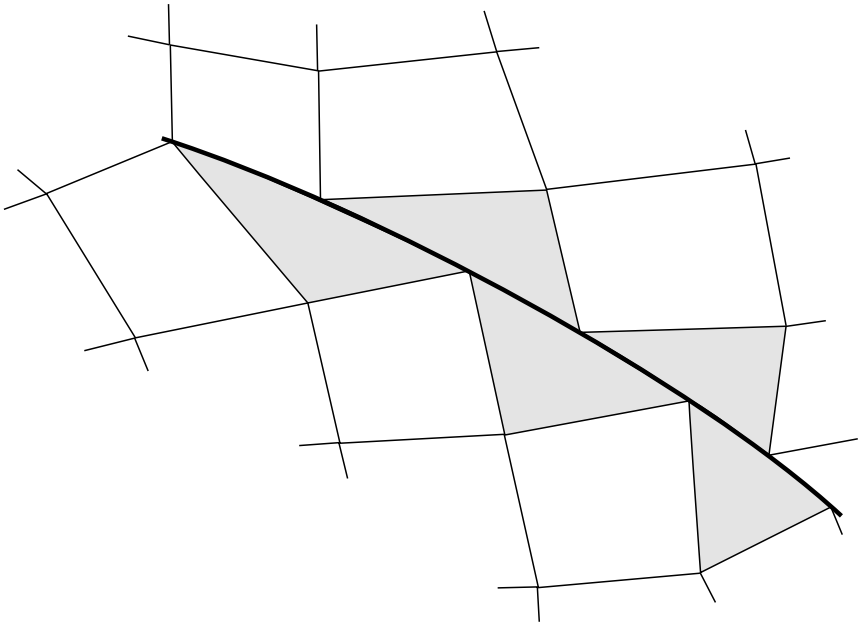


Figure 3.9: Degenerate cells.

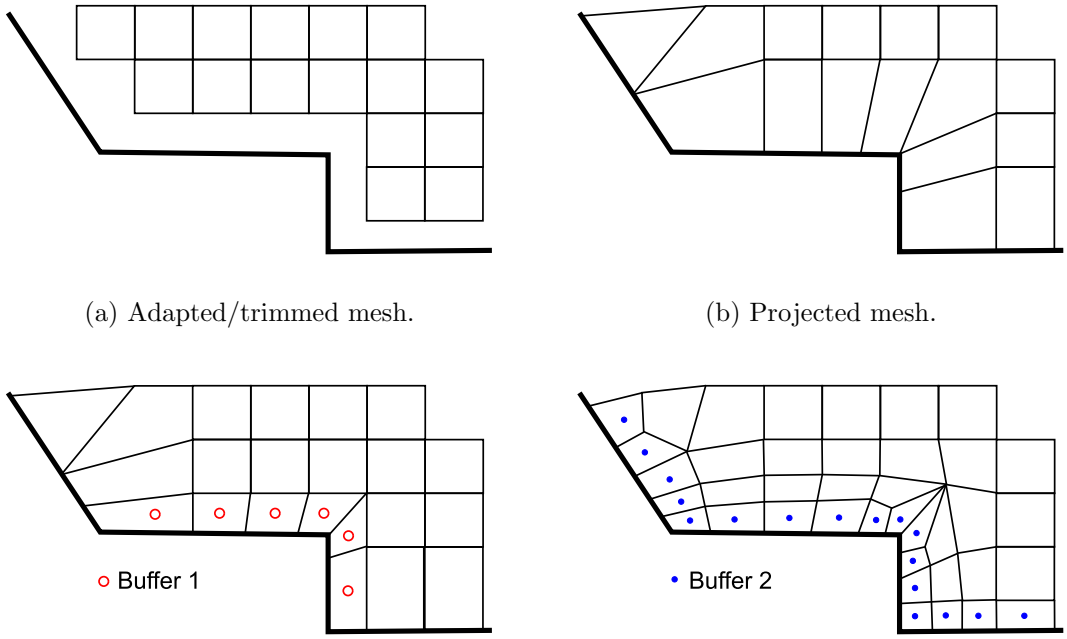
faces and reconnecting them to the volume mesh while creating necessary additional mesh entities - edges, faces and cells. A layer of new cells is inflated using Laplacian smoothing (Buffer 1 in Figure 3.10c). This first step removes a portion of degenerate volume cells and surface faces attached to topological edges. However, degenerate cells and faces along non-convex topological edges still remain in the mesh. To remove those, the second step introduces additional isolated buffer layers on each topological face (Buffer 2 in Figure 3.10d). This step is performed by a procedure similar to the one described above. It removes remaining degenerate cells from the mesh. As a result, two layers of quasi-orthogonal cells are produced by this two-fold buffer insertion strategy.

Example of a mesh resulting from the boundary fitting step is presented in Figure 3.11.

3.6 Mesh quality improvement

The addition of buffer layers usually greatly improves the quality of a mesh obtained after the boundary fitting procedure. However, requirements to mesh quality posed by most flow solvers oblige to further improvement. In particular, a number of nearly flat or twisted cells can still be present in the mesh. Obviously, Laplacian smoothing is in general not sufficient to treat such problematic configurations. For this purpose, a mesh optimization technique [60] has been implemented and has been proved to be efficient.

The optimization method is based on a mechanical model which is meant to minimize deformation between an ideal reference cell and an actual cell (see Figure 3.12). Deformation



(a) Adapted/trimmed mesh.

(b) Projected mesh.

(c) Mesh after buffer 1 insertion.

(d) Mesh after buffer 2 insertion.

Figure 3.10: Buffer insertion strategy.

is measured by the functional σ representing the energy of deformation of a hyper-elastic isotropic homogeneous material. In three dimensions, the functional can be expressed as follows:

$$\sigma_{3D} = \sum \int \int \int \sigma^e_{3D} \cdot d\xi_1 \cdot d\xi_2 \cdot d\xi_3, \quad \xi_i = \xi^\star_i / h_i \tag{3.1}$$

where summation is performed over all mesh cells; h_i represents target cell size in direction i, and $\xi^\star_i$ is a non-normalized coordinate in this direction. Cellular contribution is given by:

$$\sigma^e_{3D} = (h_1 h_2 h_3)^\gamma \left[C \cdot (I_1 + I_2 - 6 \cdot J) + K \cdot (J-1)^2 \right] \tag{3.2}$$

where C, γ and K are constants chosen to be equal to 1. I_1, I_2 and J represent invariants of a cell deformation tensor of (J denotes the Jacobian determinant).

The first term of (3.2) reflects the default of orthogonality of an actual cell while the second one reflects cell flatness. Detailed expressions for these invariants can be found in [60, 62]. The optimization algorithm minimizes the functional using a weighted steepest descent technique which locally updates coordinates of nodes in a Jacobi procedure. A sample optimized mesh is depicted in Figure 3.13.

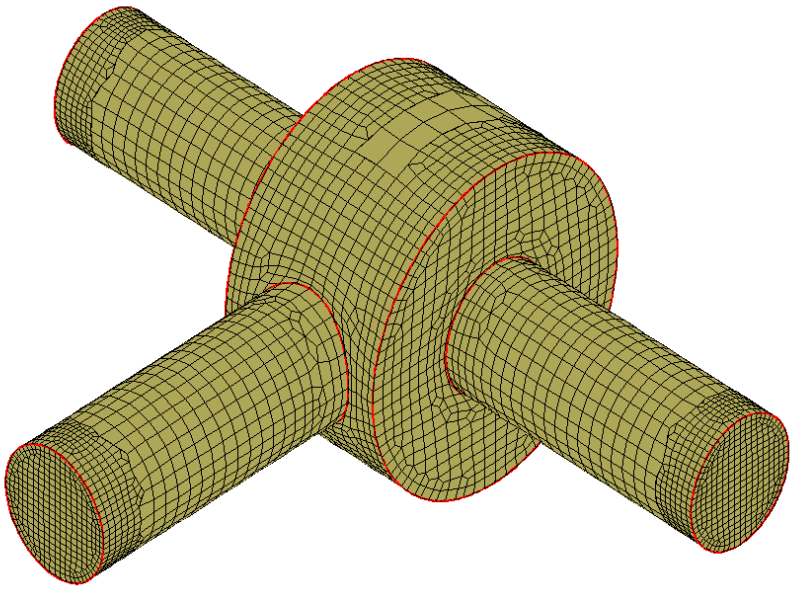


Figure 3.11: Boundary-fitted mesh for a three-pipe joint configuration.

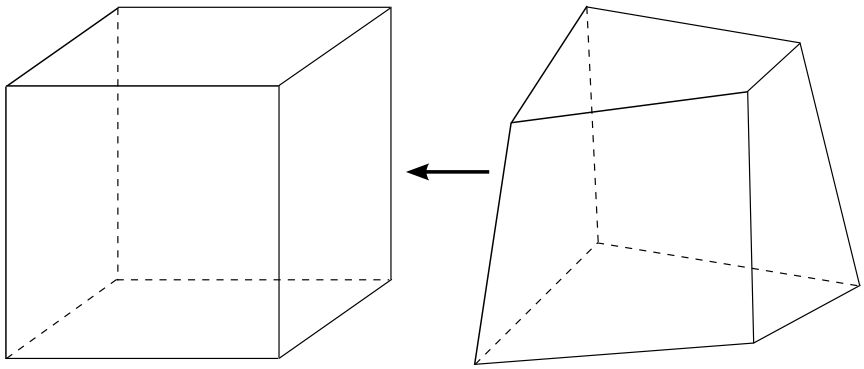


Figure 3.12: Optimization.

The optimization technique is capable of removing overlapping and degenerate cells but is an order of magnitude more expensive than the Laplacian smoothing. Development of a new fast and reliable mesh quality improvement approach is addressed in Chapter 4 of this thesis.

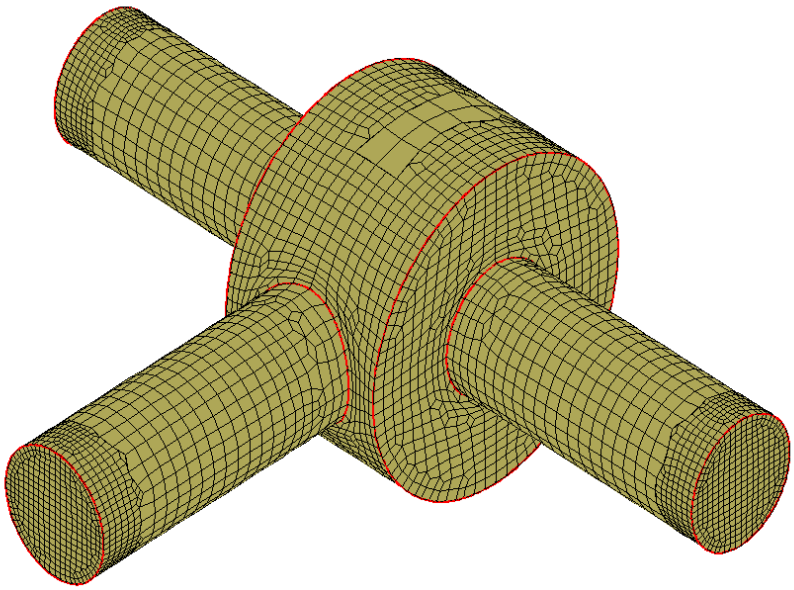


Figure 3.13: Optimized mesh for a three-pipe-joint configuration.

3.7 Viscous layer insertion

Several layers of high-aspect ratio hexahedral cells can be added to a mesh close to surface geometry for correct resolution of boundary layers in high-Reynolds number flow simulations. A method using tangential refinement of cells adjacent to solid boundaries in normal-to-wall direction is applied (see Figure 3.14). The method is capable of generating layers of highly stretched cells along solid boundaries as well as of satisfying user-specified first-cell size and stretching ratio. However, such simplified procedure often produces meshes with excessive cell size discontinuities in normal-to-wall direction between anisotropic and isotropic regions. Figure 3.15 presents an example of such a mesh.

Chapter 5 presents a new powerful approach for generation of high-quality meshes for viscous computations developed in the thesis. The new method directly addresses the problem of cell size discontinuities between isotropic and anisotropic regions.

3.8 Results

The first example demonstrates performance of the mesh generator for a turbomachinery application. A fully hexahedral non-conformal mesh is generated inside a generic volute. The CAD model contains four cyclic topological edges and four topological faces. The

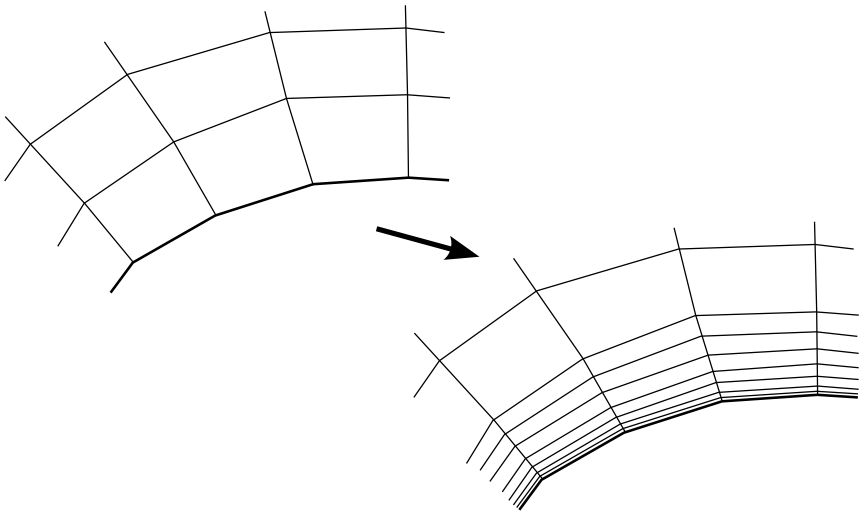


Figure 3.14: Viscous layer insertion.

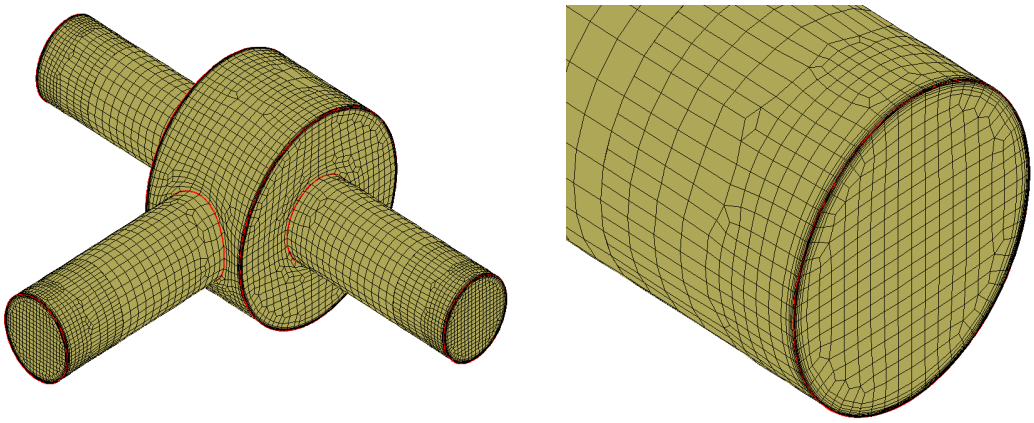


Figure 3.15: Viscous mesh for a three-pipe-joint configuration.

initial mesh consists of thirty two, twenty five and eight cells in the three respective Cartesian directions; the total number of cells is 6,400. After three refinements and a boundary fitting, the mesh contains 136,155 cells. Buffer insertion increases cell count to 229,331 cells. Figure 3.16 shows the resulting surface mesh with all topological edges captured successfully. All three refinement criteria (target sizes, distances between opposite walls, and high curvature) were implied in this case.

The second example shows performance of the generator when used for an external aerodynamics application. The computational domain around a generic business jet geometry contains fourty eight topological edges and twenty five topological faces. This domain consists of the airplane model and the external bounding box. The initial mesh consists of only one cell. Eleven refinements are performed, after which the mesh is snapped onto the

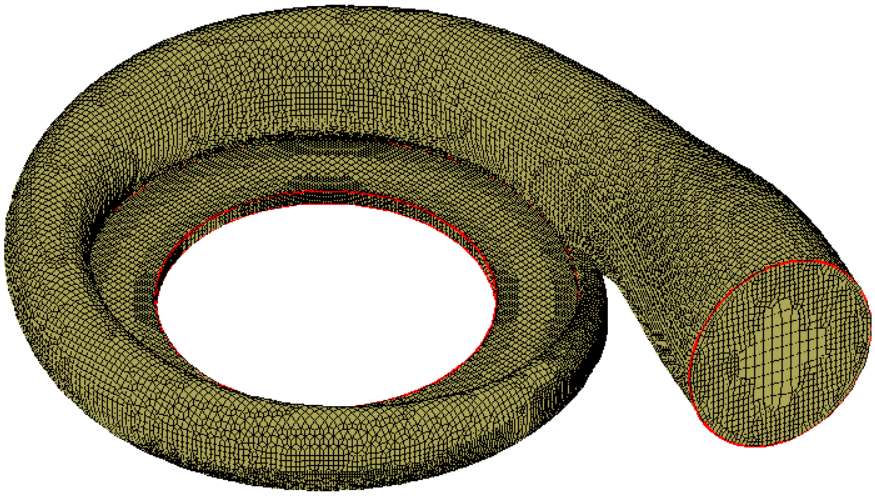


Figure 3.16: Mesh within a generic volute.

geometry. At this point, the mesh contains 352,933 cells. Inserting buffer layers results in a 424,514 cell mesh. Figures 3.17 and 3.18 depict two fragments of the surface mesh. This example exhibits a certain level of complexity, which consists in presence of very sharp trailing edges of wings, ailerons, and nacelles. Successful meshing of this test case indicates good performance of the generator.

In the next example, a mesh is generated for another external aircraft configuration. An unstructured hexahedral mesh is generated around the wing-body-nacelle configuration DLR-F6. The respective computational domain contains fifty topological edges and twenty two topological faces. The initial mesh contains twelve, five and ten cells in each respective Cartesian direction, i.e. six hundred cells in total. After thirteen refinements are performed, the mesh is successfully projected onto the geometry and all topological entities are captured. The resulting mesh contains 536,206 cells. The next operation, buffer insertion, increases cell count by approximately thirty percent to result in a 710,249 cell mesh. At this point, mesh quality is improved by means of optimization. The latter is followed by insertion of layers of highly stretched cells along solid boundaries (i.e., the aircraft surface). Parameters of the layers are: the first cell size is 10^{-3}, the stretching ratio is 1.2, and the number of layers is seven. The resulting viscous mesh contains 1,834,477 cells. Figure 3.19 shows a general view of the surface mesh on the DLR-F6 configuration. A close-up view of the nacelle mesh is depicted in Figure 3.20. This test case represents a reference geometry for CFD community, and a successful mesh generation is an important validation milestone for the mesh generator.

Finally, the last example represents external geometry around a fragment of a generic

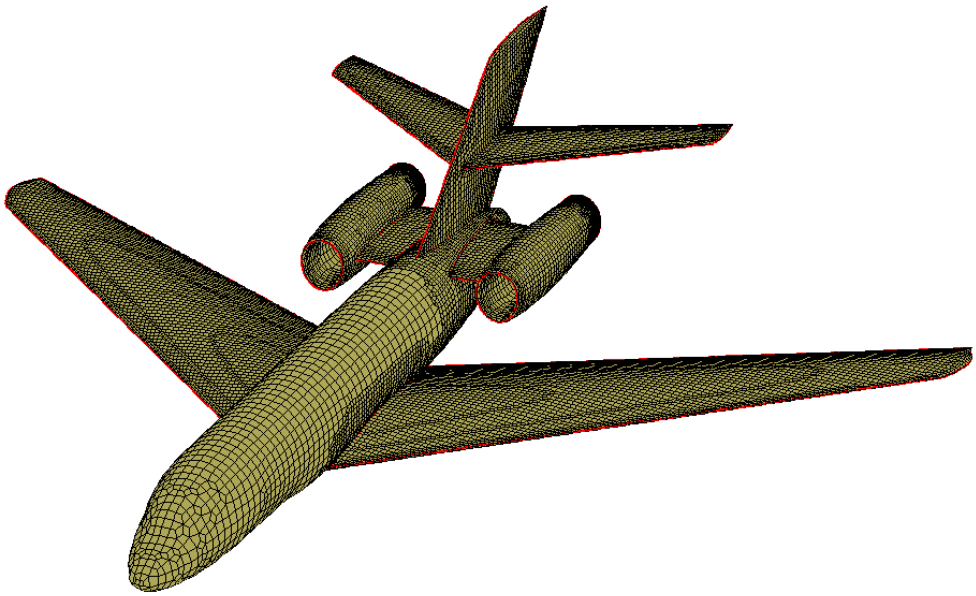


Figure 3.17: Surface mesh on a business jet; Fragment 1.

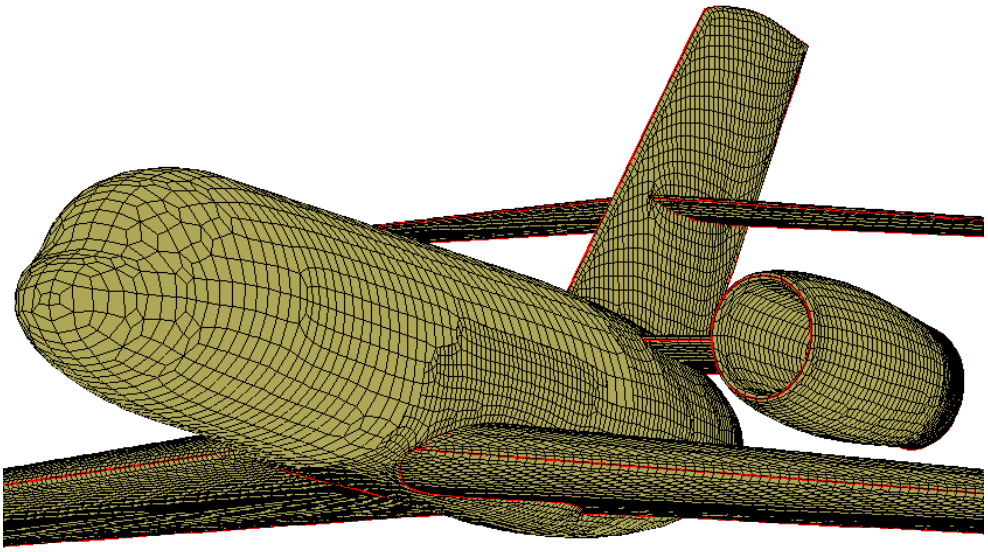


Figure 3.18: Surface mesh on a business jet; Fragment 2.

Formula One car. The computational domain is more complicated than in the examples above. It contains ninety nine edges and fourty one faces. The initial mesh contains eleven, nine and ten cells in the respective Cartesian directions. After nine refinements are performed, the mesh is trimmed and body-fitted successfully. The snapped mesh

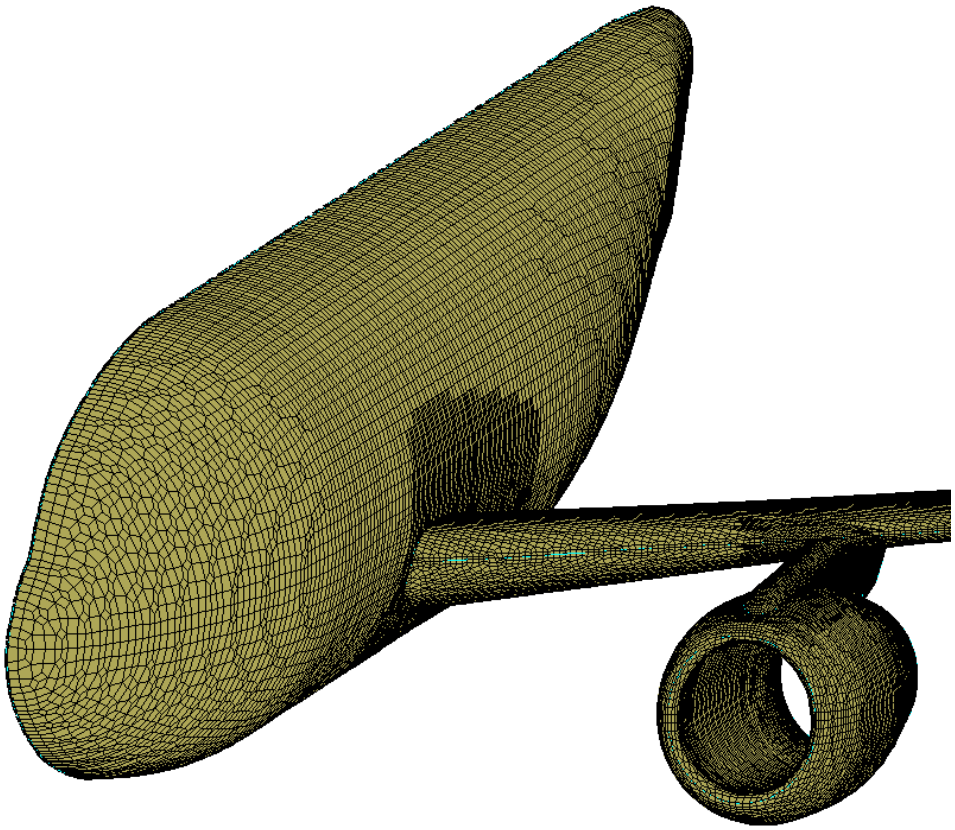


Figure 3.19: Surface mesh on a wing-body-nacelle DLR-F6 configuration.

contains 598,339 cells prior to buffer insertion. The latter operation increases this number to 830,386 cells. A general view of the surface mesh generated on the car body is presented in Figure 3.21. Figure 3.22 depicts a close-up view of the front wing of the car. This geometry is rather complicated and contains many elements which typically pose problems for automatic mesh generation. These are the trailing edges and very small (close to zero degrees) and very large (close to one hundred and eighty degrees) dihedral angles between topological surfaces. Nevertheless, the presented mesh generator demonstrates good performance by meshing the case successfully.

3.9 Conclusions

Capabilities of fully hexahedral mesh generation methodology have been demonstrated on several configurations. A challenging goal for an unstructured grid generator is to provide an engineer with a fully automatic way of discretizing a computational domain to enhance the speed of simulation of physics in complex geometries. To begin with, the present

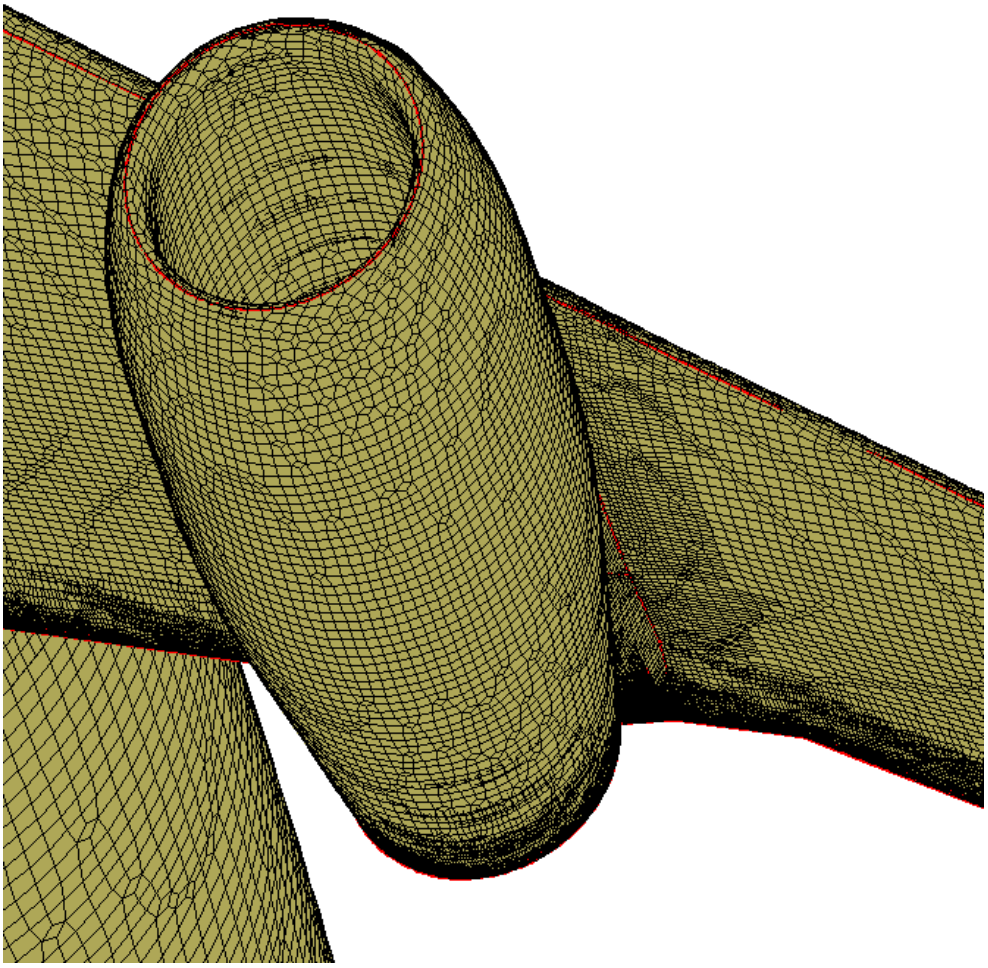


Figure 3.20: A close view of a nacelle surface mesh of the DLR-F6 configuration.

technique is very tolerant to quality of triangulations used for geometry description. It simplifies the potentially cumbersome translation of a CAD model into a native data set of the mesh generator. Secondly, the mesh generator automatically adapts the initial mesh in such a way that the final one is graded according to typical length scales of surrounding geometry with minimal user intervention. Furthermore, unlike other related Cartesian-based methods, the present technique automatically fits a mesh to geometry and removes degenerate cells and faces. It is also important, that the method does not require existence of a surface mesh with length scales compatible with physics simulation prior to volume mesh generation. Instead, it is directly obtained as a by-product of the boundary fitting step. Also, the method implements sophisticated algorithms to preserve geometry features of the domain of interest. This obviously becomes a benefit at the physics simulation stage, as the effects of corners and ridges on simulation results can be significant.

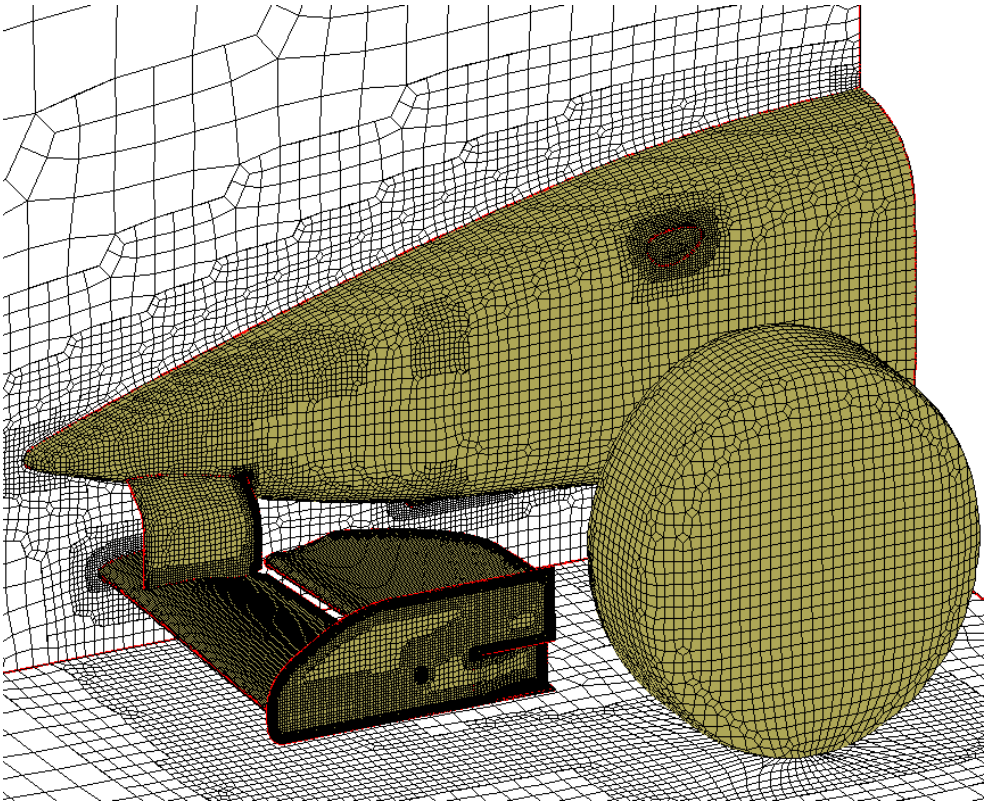


Figure 3.21: Surface mesh on a fragment of a generic Formula One car.

However, further investigations are required in order to enhance performance of the mesh generator. Primarily, the optimization and viscous layer insertion algorithms are to be addressed. That is the reason why the present thesis addresses these two issues in Chapters 4 and 5, respectively.

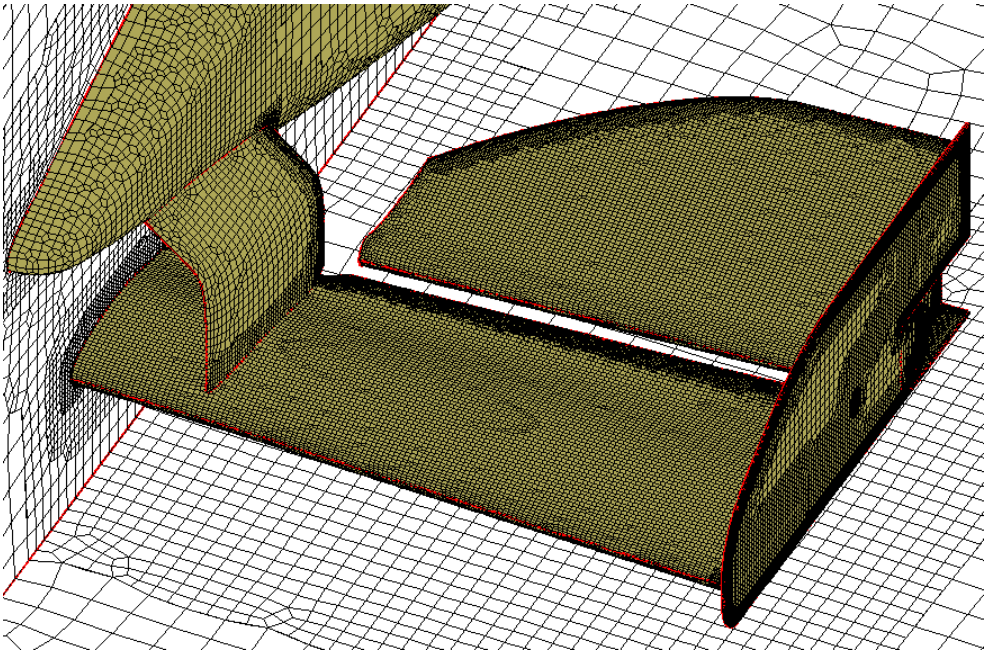


Figure 3.22: Surface mesh on the front wing of the Formula One car.

Chapter 4

Isotropic Mesh Quality Improvement

4.1 Introduction

Automatic generation of unstructured meshes sometimes produces meshes with poorly shaped and inverted elements. The case is even worse for hexahedral meshing because of high flexibility of a hexahedron to become extremely distorted, and its intrinsic difficulties when generating such meshes for complex configurations. Presence of invalid mesh elements in a mesh may lead to a major breakdown of the simulation algorithm. Therefore, development of a posteriori quality improvement tools is of prime importance.

Meshes consisting solely of simplicial elements (triangles or tetrahedra) are most likely subject to quality improvement. This is due to presence of linear mapping of a uniform triangle or tetrahedron onto an arbitrary triangular or tetrahedral element. Poorly-shaped cells can be detected by evaluating the Jacobian of the corresponding mapping. Linearity of such mappings is extremely important because the Jacobian of linear mapping is constant. Therefore, evaluating the mapping Jacobian unambiguously answers the question whether or not mesh element is inverted, in other words, has a non-positive volume.

However, the situation for computational meshes consisting of non-simplicial elements such as quadrilaterals or hexahedra is different. Particularly, no linear mapping exists for a hexahedral element to be transformed into a reference element. Instead, the mapping between a unit cube and a hexahedral cell is trilinear and its Jacobian exhibits a rather complicated behavior throughout the cell. Also, no convexity relations can be effectively exploited because faces of a hexahedron are generally not planar and can be folded even when all eight corners are convex. This makes the task of quality improvement much more complicated for hexahedral meshes as compared to simplicial and even quadrilateral meshes. For the latter, simple convexity relations can be easily established.

An effective method for untangling and optimization of hexahedral unstructured non-conformal meshes is presented in the thesis [34, 75]. The method has been developed as a part of the new mesh generator presented in Chapter 3. It is able to untangle invalid (i.e.,

concave) cells and optimize valid but poorly-shaped cells resulting from grid generation process. The ultimate goal is to obtain a mesh with all convex cells.

The grid generation process employed in the generator is based on overlay approach and includes several stages. First, an initial non-boundary-conforming mesh is created and refined based on geometry particularities. Cells that fall outside or intersect the domain are removed from the volume mesh. Next, surface of the resulting staircase mesh is projected onto the domain boundary and layers of buffer cells are inserted between the volume mesh and the corresponding surface mesh in order to obtain a body-conforming mesh. Concave and poorly shaped cells which are usually concentrated near the boundary may appear during the projection step. In the final stage, new untangling and optimization tools are applied to transform these cells into convex ones and to recover a mesh of high quality. Optionally, layers of high aspect ratio cells for viscous flow computations may be inserted.

The grid generator is coupled with a new flow solver, which intensively adapts the mesh based on local solution error estimation in order to obtain a mesh optimized for a particular flow solution. Refinement of a concave cell may result in new cells with negative volumes. The latter are unacceptable for reasons of robustness and accuracy of the flow solver. That is why an automatic optimization procedure is necessary. It combines the untangling and optimization techniques with Laplacian smoothing to result in an efficient automated tool for improvement of quality of unstructured hexahedral meshes.

The upcoming sections are organized as follows. To begin with, methods for untangling and optimization are considered separately with the focus on their individual algorithmic details. After that, an approach combining both techniques is described in details. Finally, performance of the combined approach is analysed, conclusions about its efficiency are drawn and recommendations for future development are suggested.

4.2 Mesh “untangling”

4.2.1 Problem statement

The problem of untangling of an unstructured hexahedral mesh can be formulated as follows:

Problem. *There exists an unstructured hexahedral non-conformal mesh with a fixed boundary. Let $\{V_I\}$ be the set of internal non-hanging nodes of the mesh, $\{V_B\}$ - the set of non-hanging boundary nodes. The nodes of $\{V_B\}$ are not allowed to move. A new set of positions for nodes of $\{V_I\}$ must be found, such that no cells in the mesh are concave, while topology of the mesh is preserved.*

A node is called *hanging* if it results from refining a mesh edge into two edges or a face into four faces, while the parent edge or face still remain in the mesh. Geometric position of a

hanging node is determined by positions of vertices of the corresponding edge or face (see Figure 4.1). The four red nodes are edge-based. They are located at corresponding edge centers. The black node is face-based. Hanging nodes of this type are placed at centers of faces to which they belong.

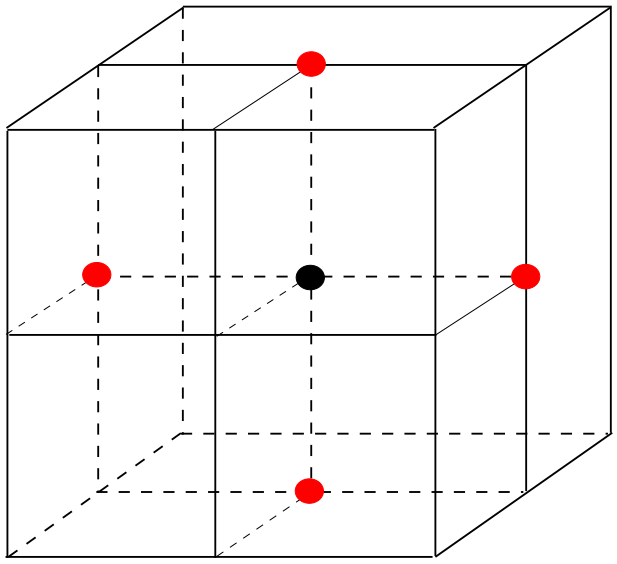
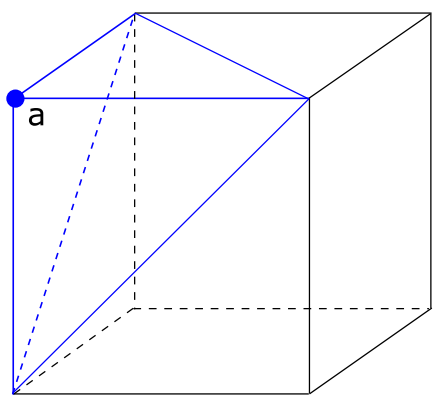
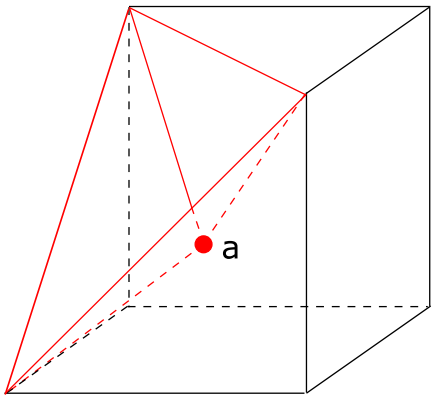


Figure 4.1: Different types of hanging nodes.



(a) A convex element.



(b) Aconcave element.

Figure 4.2: Convex and concave hexahedral elements.

A cell is called *concave* if at least one of the tetrahedra constituting the cell has negative volume (see Figure 4.2). Similarly, a cell is called concave *with respect to a node* if at least one of the tetrahedra constituting the cell and referencing this node has negative volume.

Due to multiple topological and geometrical limitations and dependencies, the problem of global mesh untangling is too complicated to be solved using implicit methods. Instead, the problem is decomposed into a series of consecutive untangling operations applied to sub-meshes possessing only one internal node. The set of cells surrounding a non-hanging mesh node N (see Figure 4.3) is called *the ball* of node N: $Ball(N)$. Each cell of the ball $Ball(N)$ can be convex or concave depending on the location of the node N as shown in Figure 4.4. In this Figure, dashed lines form the boundary between regions of "convex" and "concave" positions for node N.

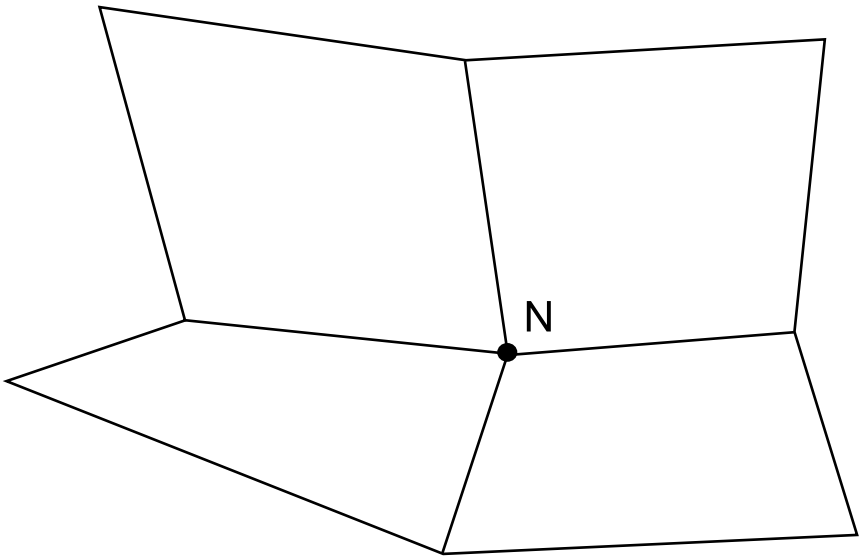


Figure 4.3: Ball of cells surrounding node N.

The problem of global mesh untangling is solved iteratively. Each ball of cells in the mesh is successively untangled in every iteration. Iterations are repeated until either no concave cells remain in the mesh or no concave cells can be untangled anymore by moving any internal nodes. Finally, the problem of local untangling can be formulated as follows:

Problem. *There exists an arbitrary non-hanging node N belonging to $\{V_I\}$ and the respective ball of surrounding cells $Ball(N)$. A new position for node N must be found, such that no cells in $Ball(N)$ are concave with respect to N.*

The *Kernel* or the *feasible set* of node N can be defined as a sub-region of $Ball(N)$, such that all cells in $Ball(N)$ are non-concave with respect to node N if and only if N is positioned at any point within its kernel $Kernel(Ball(N))$. A two-dimensional illustration (Figure 4.5) shows that the kernel of node N is the intersection of feasible sets of node N with respect to each cell in the ball. The kernel is bounded by straight lines and always represents a convex polyhedron. Placing node N at any location inside $Kernel(Ball(N))$ guarantees convexity of $Ball(N)$ provided that the kernel exists, otherwise the untangling problem has no solution.

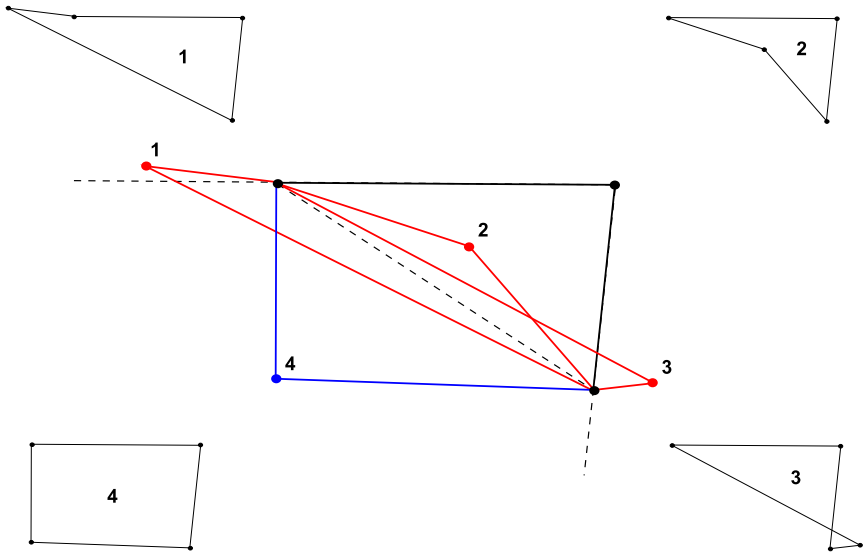


Figure 4.4: Node position defines whether a cell is convex or concave.

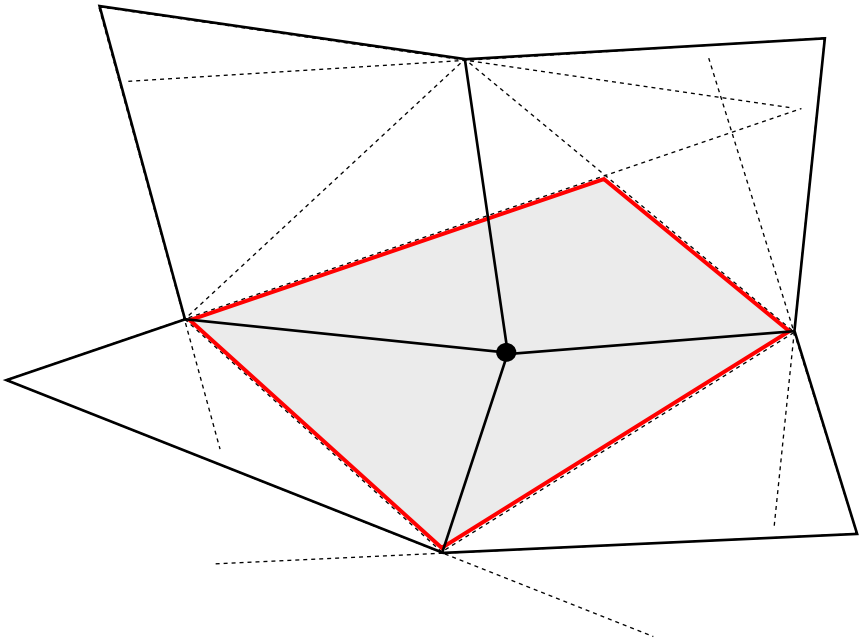


Figure 4.5: The kernel of a node.

4.2.2 Decomposition of a hexahedron into tetrahedra

For the sake of clarity, all previous illustrations for untangling are shown in 2-D. There is one-to-one correspondence between convexity relations for a quadrilateral and its decomposition into triangles. A corner of a quadrilateral is convex if and only if the respective corner triangle is non-inverted (see Figure 4.6).

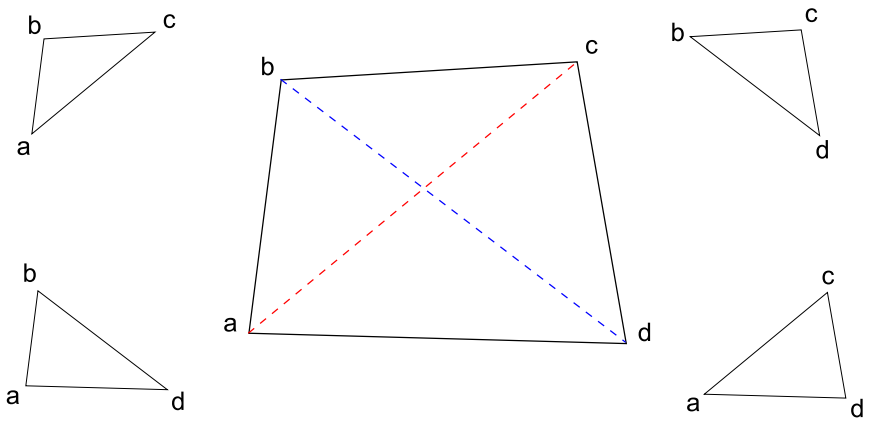


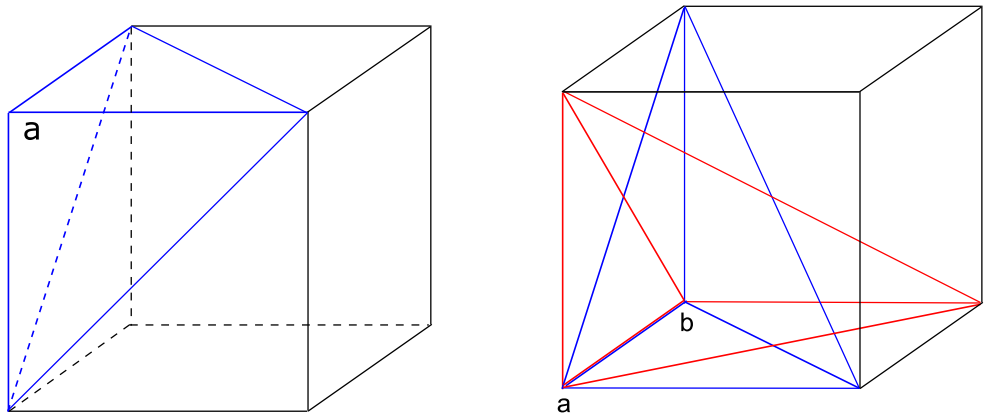
Figure 4.6: Decomposition of a quadrilateral into triangles.

However, the situation in 3-D is not as straightforward. In conventional approach, convexity of a hexahedral element is established by evaluating volumes of eight corner tetrahedra constituting this element. If at least one such tetrahedron is inverted, the respective hexahedron is considered to be concave. Vice versa, the hexahedron is considered to be convex if and only if all tetrahedra are non-inverted.

Volumes of tetrahedra constituting a hexahedral cell represent the Jacobian matrix determinants of the transformation between a real and a reference tetrahedra. Volume of a tetrahedron is a linear function of coordinates of each vertex. A hexahedron can be decomposed into several generally non-equal tetrahedra. In case of the eight corner decomposition (further referred to as the *8-tet* decomposition, see Figure 4.7a), each tetrahedron corresponds to one corner. A Jacobian equal to the tetrahedron volume is assigned to the respective corner vertex.

However, ensuring a positive Jacobian at cell corners does not guarantee Jacobian positiveness throughout the cell. In general, no Jacobian positiveness can be guaranteed for an entire cell at all. Instead, the problem can be solved discretely by ensuring Jacobian positiveness at certain locations within an element. A thorough analysis of the issue of discrete analogue of Jacobian positiveness is presented by Ivanenko [58, 59] and Ushakova [120]. It was shown that the 8-tet decomposition may not be sufficient to ensure positiveness. There exists a small probability of strong distortion of a hexahedron while all eight corner tetrahedra are positive. In order to address this class of distorted hexahedra, an alternative scheme for decomposition of a hexahedron into twenty four tetrahedra (further referred to as the *24-tet* decomposition, see Figure 4.7b) was proposed by Ivanenko [58].

According to this decomposition, two tetrahedra are assigned to each cell edge. For a twelve-edge cell, twenty four tetrahedra represent a complete decomposition. For a unit hexahedron (i.e., having volume equal to one), the total volume of twenty four constituting tetrahedra is equal to four. Therefore, the tetrahedra cover the volume of a hexahedron four-fold. These two alternative decompositions (8-tet and 24-tet) represent the two respective criteria for defining whether a hexahedron is convex. No conventional term of



(a) 8-tet decomposition. (b) 24-tet decomposition.

Figure 4.7: Decomposition of a hexahedron into tetrahedra.

"convexity" is applicable to hexahedra due to their complicated nature.

4.2.3 Simplex method

The problem of local untangling is solved herein by means of the so-called *simplex method* [101]. The method is generally used for solving linear optimization problems and can be described as follows. A linear function to be optimized on a set of n independent variables is given and a series of linear constraints on these variables is specified. In a general form, this problem can be formulated as follows:

For N independent variables x_1, $x_2, \ldots,$ x_N, maximize function

$$w = a_{01} \cdot x_1 + a_{02} \cdot x_2 + \ldots + a_{0N} \cdot x_N \tag{4.1}$$

subject to primary constraints

$$x_1 \geq 0,\ x_2 \geq 0, \ldots,\ x_N \geq 0 \tag{4.2}$$

and subject to $M = m_1 + m_2 + m_3$ additional constraints, m_1 of them being of form

$$a_{i1} \cdot x_1 + a_{i2} \cdot x_2 + \ldots + a_{iN} \cdot x_N \leq b_i, \quad b \geq 0, \quad i = 1, \ldots, m_1 \tag{4.3}$$

m_2 of them being of form

$$a_{i1} \cdot x_1 + a_{i2} \cdot x_2 + \ldots + a_{iN} \cdot x_N \geq b_i, \quad b_i \geq 0, \quad i = m_1 + 1, \ldots, m_2 \tag{4.4}$$

and m_3 of them being of form

$$a_{i1} \cdot x_1 + a_{i2} \cdot x_2 + \ldots + a_{iN} \cdot x_N = b_i, \quad b_i \geq 0, \quad i = m_2 + 1, \ldots, m_3 \tag{4.5}$$

Each of the additional constraints represents a straight line in n dimensions. Position of the found solution should satisfy every constraint. A non-strict constraint ($i = 1, \ldots, m_1, m_1 + 1, \ldots, m_2$) implies that the solution is positioned on one side of the line in order to satisfy this constraint. A strict constraint ($i = m_2 + 1, \ldots, m_3$) implies that the solution is positioned exactly on the line. The simplex method is capable of finding the solution if it exists. The solution is always a point on the boundary of the region limited by constraints. This is due to the fact that a linear function defined on a bounded domain can reach its maximum on the boundary only.

If the solution does not exist, there may be two possible reasons why. The first one is that the region bounded by linear constraints may not necessarily be enclosed. Therefore, it is possible that the function reaches its maximum at infinity. The second reason is that the region bounded by constraints may be empty.

Figure 4.8 illustrates the basics of the solution procedure. For simplicity, let us assume that the solution exists and the kernel is closed, i.e. no solution at infinity is possible. The constraints bound the region (i.e. the feasible set) where the solution is to be found. The inclined plane represents the target function to be maximized. When the feasible set is extruded along the z-axis, it intersects the target function plane to result in a polygon in this plane. A position with the maximum value of z-coordinate in this polygon is the linear program solution.

4.2.4 Solution methodology

The solution of a local untangling problem must be the new position of node N, such that no concave cells remain in $Ball(N)$. In order to set up the simplex problem properly, a series of linear constraints must be established and a target function must be defined. Each cell in $Ball(N)$ provides a number of non-strict linear constraints (Figure 4.5). In this 2-D illustration, each quadrilateral can be subdivided into two triangles by each of the two diagonals. Four resulting triangles must have positive volumes in order to guarantee cell convexity. The condition of volume positiveness for a triangle can be expressed in terms of coordinates $(x, y)_{0,1,2}$ of its vertices as follows:

$$\begin{vmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{vmatrix} \geq 0 \tag{4.6}$$

or

$$(x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0) \geq 0 \tag{4.7}$$

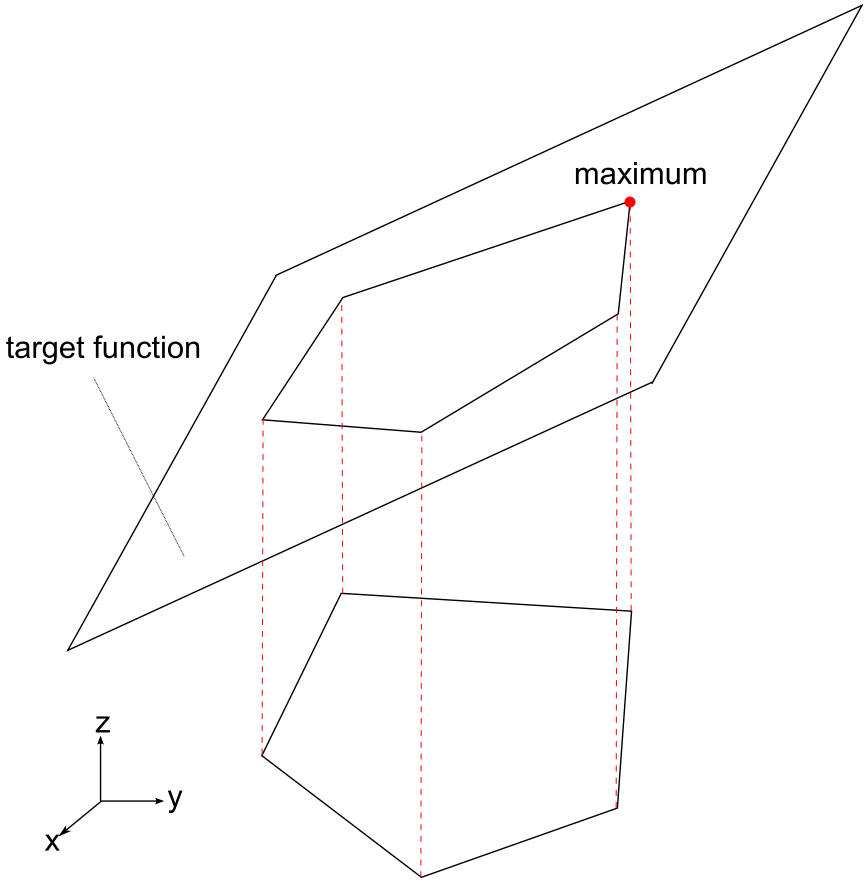


Figure 4.8: Linear programming and simplex method.

This expression is linear with respect to any of vertex coordinates. Node N is referenced by three of the four triangles constituting the quadrilateral. Thus, each cell of $Ball(N)$ contributes exactly three linear constraints. In 3-D the number of constraints per cell depends on the pattern used to decompose a cell into tetrahedra. Each tetrahedron always contributes one constraint. For the 8-tet decomposition (Section 4.2.2) the number of constraints is eight; for the 24-tet decomposition it is twenty four, respectively.

As we are interested only in finding *any* point within the kernel, the target function can be chosen almost arbitrarily. Its gradient must be non-zero; otherwise, no solution will be found. Unfortunately, solutions provided by the simplex method are always located on kernel boundary, which is unacceptable. In this case, at least one cell in the ball remains in undetermined, or "locked" state between concave and convex. If this situation emerges due to kernel degeneracy, it indicates that no solution exists. In all other cases, when the kernel is not degenerate, the point located strictly within the kernel is always possible to find. This is achieved via combining solutions resulting from different target functions as described below.

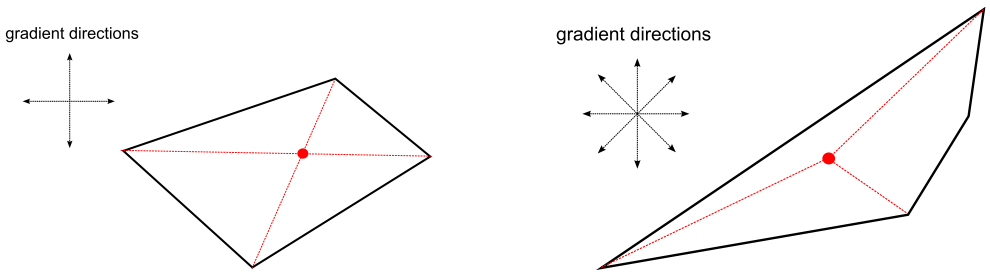
For the sake of clarity, the combining procedure is explained using 2-D framework. The

following set of target functions is employed:

$$\begin{cases} w_0 = +x \\ w_1 = -x \\ w_2 = +y \\ w_3 = -y \end{cases} \tag{4.8}$$

Linear optimization of functions of this set results in four opposite positions on the kernel boundary: with maximum x, minimum x, maximum y and minimum y. These positions are averaged to result in a new point. For a general kernel geometry, it is located strictly within the kernel (see Figure 4.9a). However, it is sometimes possible that the averaged position still appears on the kernel boundary. A few examples of kernel shapes that lead to such breakdown are illustrated in Figure 4.10. In this case, an extended set of target functions is employed. It includes four basic functions (4.8) and four additional ones:

$$\begin{cases} w_4 = +x + y \\ w_5 = -x - y \\ w_6 = +y - x \\ w_7 = -y + x \end{cases} \tag{4.9}$$



(a) The basic set. (b) The extended set.

Figure 4.9: Identification of a strictly internal point within a convex polygon.

It is easy to see that using the extended set of target functions guarantees that an internal kernel point will be found provided that the kernel is not degenerate (Figure 4.9b). For the sake of efficiency, the averaging technique is applied adaptively. The basic set of target functions is used for each quadrilateral by default. The extended set is employed only if averaging fails with the basic set. Thus, the basic set is used for the majority of kernels and the extended set is used only when necessary.

The above procedure is naturally propagated to 3-D without any difficulties. A 3-D kernel represents a convex polyhedron; the simplex method and the averaging technique are ap-

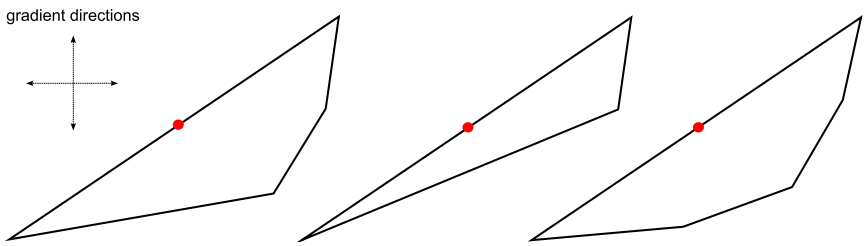


Figure 4.10: Examples of the averaging technique failure.

plied in the same manner as in 2-D. Expression similar to (4.6) can be established for the condition of volume positiveness of a tetrahedron:

$$\begin{vmatrix} x_1 - x_0 & x_2 - x_0 & x_3 - x_0 \\ y_1 - y_0 & y_2 - y_0 & y_3 - y_0 \\ z_1 - z_0 & z_2 - z_0 & z_3 - z_0 \end{vmatrix} \geq 0 \tag{4.10}$$

Similarly to (4.8), the basic set of six target functions used in 3-D is established:

$$\begin{cases} w_0 = +x & w_3 = -y \\ w_1 = -x & w_4 = +z \\ w_2 = +y & w_5 = -z \end{cases} \tag{4.11}$$

Respectively, the extended set of twenty target functions used in 3-D is as follows:

$$\begin{cases} & & w_{18} = +x + y + z \\ w_6 = +x + y & w_{12} = +z - y & w_{19} = +x + y - z \\ w_7 = -x - y & w_{13} = -z + y & w_{20} = +x - y + z \\ w_8 = +y - x & w_{14} = +z + x & w_{21} = +x - y - z \\ w_9 = -y + x & w_{15} = -z - x & w_{22} = -x + y + z \\ w_{10} = +y + z & w_{16} = +x - z & w_{23} = -x + y - z \\ w_{11} = -y - z & w_{17} = -x + z & w_{24} = -x - y + z \\ & & w_{25} = -x - y - z \end{cases} \tag{4.12}$$

As in 2-D, this set of target functions is employed when the basic set (4.11) results in a solution located on kernel boundary. Algorithm 1 finalizes the untangling procedure described in this section.

Algorithm 1 Local untangling of node N.

Begin the untangling *(node N, decomposition mode)*
Identify $Ball(N)$ - the set of surrounding cells
Decompose each cell in $Ball(N)$ into tetrahedra using one decomposition pattern (8-tet or 24-tet)
Apply linear optimization:
  Compute the set of linear constraints for linear optimization problem
  Setup the problem of linear optimization using the basic set of target functions
  Find the respective set of positions on the kernel boundary using the simplex method
Compute the internal kernel position P_I by averaging the found boundary points
If P_I is still on the kernel boundary:
  Reapply linear optimization using the extended set of target functions and recompute P_I
  If the recomputed P_I is still on the kernel boundary:
    Skip node N (no successful untangling is possible)
Move node N to P_I
End.

4.3 Mesh optimization

4.3.1 Problem statement

The problem of optimization of an unstructured hexahedral mesh can be formulated as follows:

Problem. *There exists an unstructured hexahedral non-conformal mesh with a fixed boundary. Let $\{V_I\}$ be the set of internal non-hanging nodes of the mesh, $\{V_B\}$ - the set of non-hanging boundary nodes. The nodes from $\{V_B\}$ are not allowed to move. It is assumed that the mesh contains no concave elements. A new set of positions for nodes of $\{V_I\}$ must be found, such that the mesh quality measured based on a certain quality metric increases globally, while topology of the mesh is preserved.*

Along with the problem of global mesh untangling, the problem of global mesh optimization is too complicated to be solved using implicit methods. Instead, it is decomposed into consecutive optimizations of sub-meshes possessing only one internal node. Such a sub-mesh associated with an internal mesh node is referred to as the $Ball(N)$. Figure 4.3 shows an example of such a ball.

The problem of global mesh optimization is solved iteratively. Each ball of cells in the mesh is successively optimized in every iteration. Iterations are repeated until specified exit conditions are satisfied. Finally, the problem of local optimization can be formulated as follows:

Problem. *There exists an arbitrary non-hanging node N belonging to $\{V_I\}$ and the re-*

spective ball of surrounding cells $Ball(N)$. A new position for node N must be found, such that the quality measure of cells in $Ball(N)$ increases.

The kernel or the feasible set of node N, as defined in Section 4.2.1, is a sub-region of $Ball(N)$, such that all cells in $Ball(N)$ are non-concave with respect to node N if and only if node N is positioned strictly within its kernel (Figure 4.5). The kernel is bounded by straight lines and always represents a convex polyhedron. Placing node N at any location inside its kernel guarantees convexity of $Ball(N)$ provided that the kernel exists. However, quality measure of cells in $Ball(N)$ naturally varies across $Kernel(Ball(N))$. Finding such position of node N in which the quality measure reaches its local maximum solves the problem of local optimization. The local optimization problem is solved successfully for each internal non-hanging mesh node during a number of iterations resulting in global mesh quality improvement.

4.3.2 Quality measure

Among many possible quality metrics available in literature, structural analogy [62] was chosen to be employed for the present optimization procedure. Combined with mesh untangling, this optimization procedure does not have to deal with any degenerate situations, i.e. non-convex cells. While the untangling procedure is supposed to resolve concave cells in the mesh, optimization is aimed at improving quality of convex cells. For this purpose, the functional from the structural analogy can be employed effectively.

Let us assume that trilinear mapping exists between the physical and reference hexahedral cells. The mapping can be expressed as:

$$\begin{aligned} x &= x(\xi,\eta,\zeta) \\ y &= y(\xi,\eta,\zeta) \\ z &= z(\xi,\eta,\zeta) \end{aligned} \tag{4.13}$$

In this case, density of deformation energy produced at an arbitrary position (x, y, z) can be expressed as follows:

$$I = \int\int\int \left[\frac{x_\xi^2 + x_\eta^2 + x_\zeta^2}{a^2} + \frac{y_\xi^2 + y_\eta^2 + y_\zeta^2}{b^2} + \frac{z_\xi^2 + z_\eta^2 + z_\zeta^2}{c^2} \right] \cdot \frac{abc}{J^{3/2}} \cdot d\zeta \cdot d\eta \cdot d\xi \tag{4.14}$$

where a, b, c are edge lengths of a reference hexahedron along ξ, η, ζ directions respectively, and J is the determinant of the transformation Jacobian:

$$J = \begin{vmatrix} G_{00} & G_{01} & G_{02} \\ G_{10} & G_{11} & G_{12} \\ G_{20} & G_{21} & G_{22} \end{vmatrix}, \; G_{ij} = x_{\xi_i} \cdot x_{\xi_j} + y_{\xi_i} \cdot y_{\xi_j} + z_{\xi_i} \cdot z_{\xi_j}, \; \xi_{0,1,2} = \xi, \eta, \zeta \tag{4.15}$$

In the present approach, the expression for deformation energy density (4.14) is employed as the functional for the optimization problem.

4.3.3 Solution methodology

To solve the problem of local optimization, a procedure similar to that of local untangling is employed. For each internal mesh node the functional is defined as a function of node coordinates. The functional accounts for contributions from each cell in the respective ball $Ball(N)$.

These contributions are computed as follows. Each cell in $Ball(N)$ is decomposed using either the 8-tet or 24-tet decomposition pattern, as described in Section 4.2.2. The functional of a cell is computed as a sum of functionals of either eight or twenty four tetrahedra, depending on the decomposition pattern employed.

Computation of the functional (4.14) of a tetrahedron requires components of the Jacobian matrix of mapping $\partial(x,y,z)/\partial(\xi,\eta,\zeta)$ to be discretized on this tetrahedron. In the present approach, the derivatives are discretized differently depending on the decomposition pattern employed (see Figure 4.11). For the 8-tet pattern, the following expressions are used:

$$\begin{array}{l} \frac{\partial(x,y,z)}{\partial\xi} = (x,y,z)_1 - (x,y,z)_0 \\ \frac{\partial(x,y,z)}{\partial\eta} = (x,y,z)_2 - (x,y,z)_0 \\ \frac{\partial(x,y,z)}{\partial\zeta} = (x,y,z)_3 - (x,y,z)_0 \end{array} \tag{4.16}$$

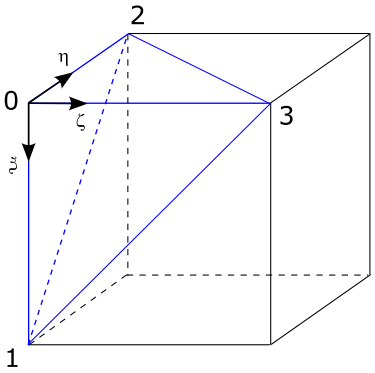
For the 24-tet pattern, the derivatives are discretized on the respective edge-based tetrahedra as follows:

$$\begin{array}{l} \frac{\partial(x,y,z)}{\partial\xi} = (x,y,z)_1 - (x,y,z)_0 \\ \frac{\partial(x,y,z)}{\partial\eta} = (x,y,z)_2 - (x,y,z)_1 \\ \frac{\partial(x,y,z)}{\partial\zeta} = (x,y,z)_3 - (x,y,z)_2 \end{array} \tag{4.17}$$

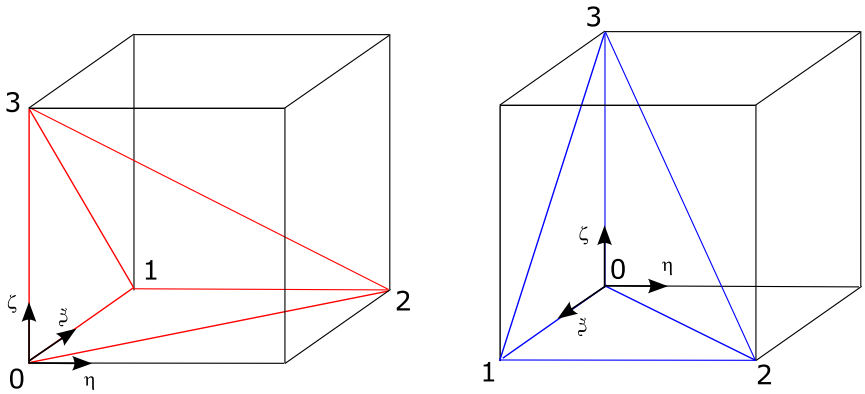
Note that the two tetrahedra depicted in Figure 4.11b correspond to the right-hand and the left-hand coordinate systems, respectively. Discretization of functional 4.14 using formulae (4.16) yields the following expression for the functional associated with a tetrahedron k of the 8-tet decomposition:

$$I_8^{(T,k)} = \left[\frac{\Delta x_{10}^2+\Delta x_{20}^2+\Delta x_{30}^2}{a_k^2} + \frac{\Delta y_{10}^2+\Delta y_{20}^2+\Delta y_{30}^2}{b_k^2} + \frac{\Delta z_{10}^2+\Delta z_{20}^2+\Delta z_{30}^2}{c_k^2}\right]_k \cdot \frac{a_k b_k c_k}{[J_8^{(T,k)}]^{3/2}} \tag{4.18}$$

whereas the respective components of $J_8^{(T,k)}$ are given as:



(a) 8-tet decomposition.



(b) 24-tet decomposition.

Figure 4.11: Computational stencils for functional discretization.

$$
\begin{aligned}
G_{8,00}^{(T,k)} &= (\Delta x_{10}^2 + \Delta y_{10}^2 + \Delta z_{10}^2)_k \\
G_{8,11}^{(T,k)} &= (\Delta x_{20}^2 + \Delta y_{20}^2 + \Delta z_{20}^2)_k \\
G_{8,22}^{(T,k)} &= (\Delta x_{30}^2 + \Delta y_{30}^2 + \Delta z_{30}^2)_k \\
G_{8,01}^{(T,k)} = G_{8,10}^{(T,k)} &= (\Delta x_{10}\Delta x_{20} + \Delta y_{10}\Delta y_{20} + \Delta z_{10}\Delta z_{20})_k \\
G_{8,02}^{(T,k)} = G_{8,20}^{(T,k)} &= (\Delta x_{10}\Delta x_{30} + \Delta y_{10}\Delta y_{30} + \Delta z_{10}\Delta z_{30})_k \\
G_{8,12}^{(T,k)} = G_{8,21}^{(T,k)} &= (\Delta x_{20}\Delta x_{30} + \Delta y_{20}\Delta y_{30} + \Delta z_{20}\Delta z_{30})_k
\end{aligned}
\tag{4.19}
$$

Here, Δx_{ij}, Δy_{ij}, Δz_{ij} denote $x_i - x_j$, $y_i - y_j$, $z_i - z_j$, respectively. Similar expression for the functional associated with a tetrahedron k of the 24-tet decomposition can be written as follows:

$$I_{24}^{(T,k)} = \left[\frac{\Delta x_{10}^2+\Delta x_{21}^2+\Delta x_{32}^2}{a_k^2} + \frac{\Delta y_{10}^2+\Delta y_{21}^2+\Delta y_{32}^2}{b_k^2} + \frac{\Delta z_{10}^2+\Delta z_{21}^2+\Delta z_{32}^2}{c_k^2}\right]_k \cdot \frac{a_k b_k c_k}{[J_{24}^{(T,k)}]^{3/2}} \tag{4.20}$$

whereas the components of $J_{24}^{(T,k)}$ are given as:

$$\begin{aligned} G_{24,00}^{(k)} &= (\Delta x_{10}^2 + \Delta y_{10}^2 + \Delta z_{10}^2)_k \\ G_{24,11}^{(k)} &= (\Delta x_{21}^2 + \Delta y_{21}^2 + \Delta z_{21}^2)_k \\ G_{24,22}^{(k)} &= (\Delta x_{32}^2 + \Delta y_{32}^2 + \Delta z_{32}^2)_k \\ G_{24,01}^{(k)} = G_{24,10}^{(k)} &= (\Delta x_{10}\Delta x_{21} + \Delta y_{10}\Delta y_{21} + \Delta z_{10}\Delta z_{21})_k \\ G_{24,02}^{(k)} = G_{24,20}^{(k)} &= (\Delta x_{10}\Delta x_{32} + \Delta y_{10}\Delta y_{32} + \Delta z_{10}\Delta z_{32})_k \\ G_{24,12}^{(k)} = G_{24,21}^{(k)} &= (\Delta x_{21}\Delta x_{32} + \Delta y_{21}\Delta y_{32} + \Delta z_{21}\Delta z_{32})_k \end{aligned} \tag{4.21}$$

Cell-associated values of functionals are computed as the average of contributions from all the tetrahedra of a decomposition. For a cell m, one obtains:

$$\begin{aligned} I_8^{(C,m)} &= \tfrac{1}{8} \cdot \sum_{k=1}^{8} I_8^{(T,k)} \\ I_{24}^{(C,m)} &= \tfrac{1}{24} \cdot \sum_{k=1}^{24} I_{24}^{(T,k)} \end{aligned} \tag{4.22}$$

Finally, each nodal functional value is contributed by cells attached to the respective node as follows:

$$\begin{aligned} I_8^{(N)} &= \tfrac{1}{N_C} \cdot \sum_{m=1}^{N_C} I_8^{(C,m)} \\ I_{24}^{(N)} &= \tfrac{1}{N_C} \cdot \sum_{m=1}^{N_C} I_{24}^{(C,m)} \end{aligned} \tag{4.23}$$

In practice, nodal functional is contributed by those tetrahedra of decompositions, for which node N is a vertex. The reason is that only these tetrahedra are affected by motion of node N, while calculating redundant functionals appears to be too costly. Contributions from these tetrahedra are summed up to result in a cell-based value. Finally, the nodal functional value $Functional(N)$ is obtained as the sum of contributions from cells in $Ball(N)$.

The behaviour of the functional across $Ball(N)$ is shown in Figure 4.12. For a given node N and its ball $Ball(N)$, an arbitrary straight line passing through the respective kernel is chosen. The functional is computed for all positions of the node on this line. It is easy to see that this functional demonstrates extremely smooth behaviour while between positions where the straight line intersects linear constraints. The functional value approaches infinity at the intersection points. This important property of the functional is employed in

the present approach to guarantee that no untangled node can become tangled again due to optimization operations.

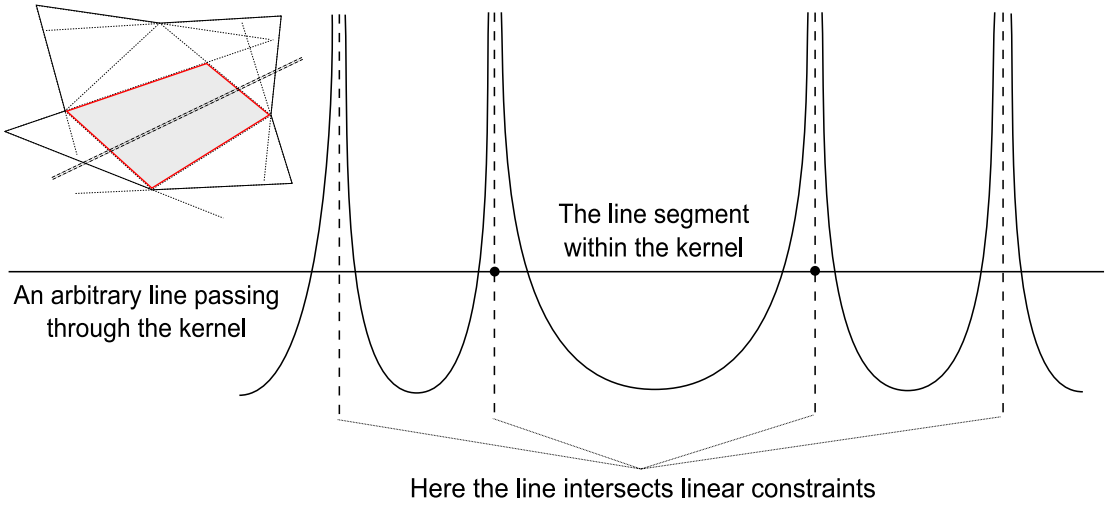


Figure 4.12: Behaviour of the nodal functional across a ball of cells.

An arbitrary ball of cells around a mesh node is optimized as follows. Direction of the highest gradient is identified by approximately computing partial derivatives of the functional along Cartesian directions:

$$\begin{aligned} \frac{\partial I^{(N)}}{\partial x} &\simeq \frac{I^{(N)}(x+\Delta x,y,z)-I^{(N)}(x,y,z)}{\Delta x} \\ \frac{\partial I^{(N)}}{\partial y} &\simeq \frac{I^{(N)}(x,y+\Delta y,z)-I^{(N)}(x,y,z)}{\Delta y} \\ \frac{\partial I^{(N)}}{\partial z} &\simeq \frac{I^{(N)}(x,y,z+\Delta z)-I^{(N)}(x,y,z)}{\Delta z} \end{aligned} \tag{4.24}$$

Here, x, y, z describe the current position of node N, and $\Delta x, \Delta y, \Delta z$ represent arbitrary small displacements. In practice, value of 0.01 percent of the average target size of surrounding edges is used for $\Delta x, \Delta y, \Delta z$. Finally, the search direction vector $\mathbf{D}$ is computed as follows:

$$\mathbf{D} = \left(\frac{\partial I^{(N)}}{\partial x}, \frac{\partial I^{(N)}}{\partial y}, \frac{\partial I^{(N)}}{\partial z}\right) \tag{4.25}$$

The local minimum of the functional is located on the line passing through the current node position and is oriented along the search direction. To find the position where this minimum is reached, a 1-D minimization problem is solved using a weighted steepest descent approach. The node is relocated to the new position. The extremely smooth nature of the chosen mesh quality measure guarantees fast convergence of this process. The full procedure is described in Algorithm 2.

Algorithm 2 Local optimization of node N.

Begin the optimization *(node N, decomposition mode)*
Identify $Ball(N)$ - the set of surrounding cells, all of which are convex with respect to N
Decompose each cell in $Ball(N)$ into tetrahedra using one decomposition pattern (the 8-tet or 24-tet)
Apply optimization:
  Compute the nodal functional value for the current position of N
  Find the direction of functional gradient
  Solve the 1-D minimization problem
  Find the new optimized position P_{OPT} for node N
Move node N to P_{OPT}
End.

4.4 Combined approach

The two approaches described in previous sections, unstructured hexahedral mesh untangling and optimization, generally do not show their best performance when applied as stand-alone tools. The untangling method is capable of eliminating the majority of concave cells. However, untangling some nodes requires the nodal quality measures of surrounding nodes to first be optimized. The proposed optimization procedure is obviously limited by the fact that it can only optimize non-tangled nodes, i.e. any mesh must be concave in order to be subject to full optimization. Therefore, a methodology combining these techniques into one global quality improvement procedure is proposed herein (see Algorithm 3).

This procedure is designed as an external loop that includes three steps. The first one combines Laplacian smoothing iterations with the untangling technique. This step eliminates all "simple" concave cells in shortest time possible. The Laplacian smoothing is aimed at slightly displacing (i.e., perturbing) concave mesh nodes in order to make the untangling more efficient. The untangling is always applied using both the 8-tet and 24-tet decomposition patterns. Finally, the second step combines the untangling procedure with optimization in the 8-tet mode; the third one combines the untangling procedure with optimization in the 24-tet mode. The two latter steps are aimed at resolving complex clusters of invalid and poor-quality cells. This is done by gradual quality improvement of nodes surrounding the problematic region. In Algorithm 3 N_{SU}, N_{O8U}, N_{O24U}, N_{O8}, N_{O24} are equal to ten; N_{FINAL8}, $N_{FINAL24}$ are equal to five.

The combined approach generally shows itself to be much more efficient than stand-alone untangling and optimization methods even when applied successively. Iterational procedure is the key component here, as it allows revisiting each node several times. Optimization and untangling procedures are applied in several sweeps, thus benefiting at each next sweep from previously improved elements. The procedure is terminated when no concave cells remain in the mesh and the average mesh cell quality is sufficiently high. Sometimes it is impossible

to eliminate all concave cells. In this case, a variation in the number of remaining concave cells is used as the termination criterion. If another sweep does not untangle any cell and the number of concave cells remains the same (non-zero), the procedure is terminated.

Algorithm 3 Combined approach.

Begin the combined smoothing/untangling/optimization procedure *(mesh M)*
Loop external iterations $i_E = 1, 2$
  If no concave cells remain, ***Exit*** external iterations
  Loop smoothing/untangling iterations $i_{SU} = 1, ..., N_{SU}$
    Disturb vertices of concave cells using Laplacian smoothing
    Untangle mesh M in the 8-tet mode until no "8-tet" concave cells remain or no cells are untangled at the last sweep
    Untangle mesh M in the 24-tet mode until no "24-tet" concave cells remain or no cells are untangled at the last sweep
    If no concave cells remain in both modes, ***Exit*** smoothing/untangling iterations
  End smoothing/untangling iterations
  If no concave cells remain, ***Exit*** external iterations
  Loop 8-tet-optimization/untangling iterations $i_{SU} = 1, ..., N_{O8U}$
    Optimize mesh M in 8-tet mode for N_{O8} iterations
    Untangle mesh M in 8-tet mode until no "8-tet" concave cells remain or no cells are untangled at the last sweep
    Untangle mesh M in 24-tet mode until no "24-tet" concave cells remain or no cells are untangled at the last sweep
    If no concave cells remain in both modes, ***Exit*** 8-tet-optimization/untangling iterations
  End 8-tet-optimization/untangling iterations
  If no concave cells remain, ***Exit*** external iterations
  Loop 24-tet-optimization/untangling iterations $i_{SU} = 1, ..., N_{O24U}$
    Optimize mesh M in 24-tet mode for N_{O24} iterations
    Untangle mesh M in 8-tet mode until no "8-tet" concave cells remain or no cells are untangled at the last sweep
    Untangle mesh M in 24-tet mode until no "24-tet" concave cells remain or no cells are untangled at the last sweep
    If no concave cells remain in both modes, ***Exit*** 24-tet-optimization/untangling iterations
  End 24-tet-optimization/untangling iterations
  If no concave cells remain, ***Exit*** external iterations
End external iterations
Optimize mesh M in 8-tet mode for N_{FINAL8} iterations
Optimize mesh M in 24-tet mode for $N_{FINAL24}$ iterations
End.

4.5 Results and discussion

The combined untangling/optimization approach was tested for various test cases in order to demonstrate its applicability to geometries of industrial interest. As an example, an unstructured hexahedral mesh was generated on the exterior domain of a geometric configuration of a missile shown in Figure 4.13. Quality of this mesh was improved by the combined approach presented in this chapter.

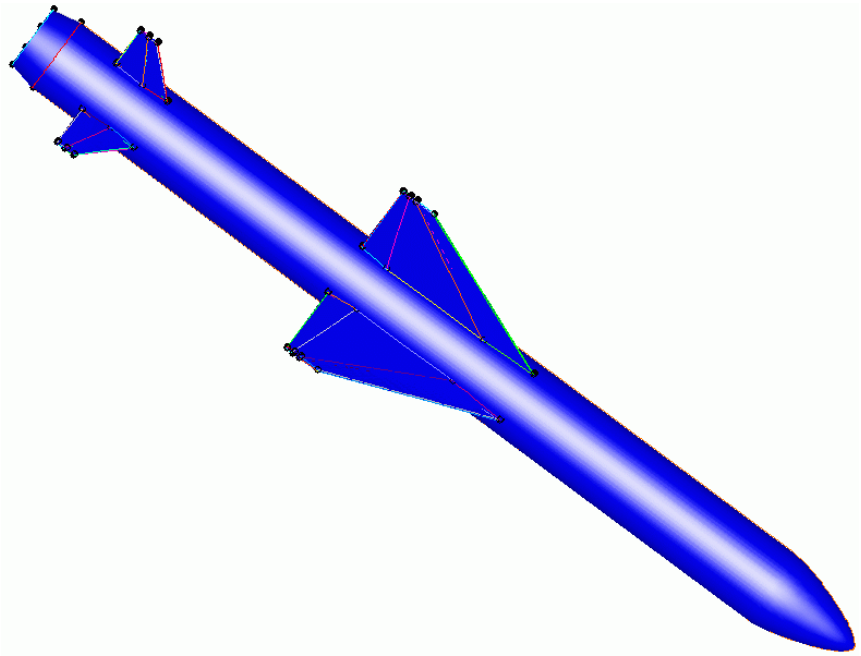


Figure 4.13: Geometry of a missile (Courtesy of Aerospatiale).

Before the quality improvement step is performed, the mesh contains a significant number of concave cells, some of which are highly distorted or have negative volumes. Typically, these cells are concentrated near the surface of a model, especially in vicinity of topological edges. Figure 4.14 depicts a surface mesh on a missile wing, and Figure 4.15 illustrates concave cells located in the vicinity of the wing. Quality of concave cells ranges from rather acceptable up to extremely distorted or even inverted. Figure 4.16 shows a close-up view of cluster of concave cells located near one of the rear wings of the missile. Quality distribution of poorly-shaped cells (see Figure 4.17) shows that some of them are very distorted and a quality improvement procedure is necessary.

The combined approach described in previous sections is applied to the missile mesh as the final step of mesh generation process. In this example, the approach has transformed one hundred percent of concave cells into convex ones. Figures 4.18 and 4.19 compare quality distributions of the meshes before and after optimization. It is of interest that only three cells (as compared to two hundred and five prior to the quality improvement step)

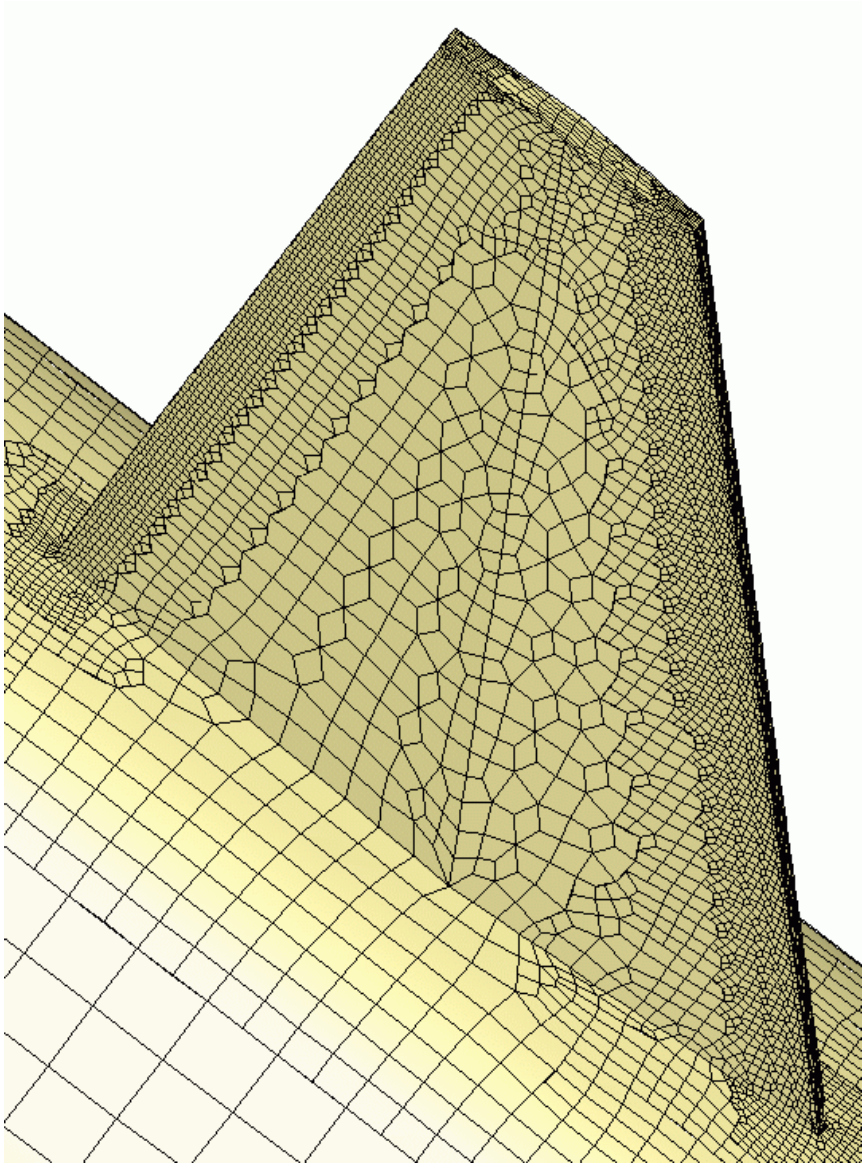


Figure 4.14: Surface mesh on a missile wing.

with dihedral angles lower than twenty degrees remain in the mesh. The optimization step consumes approximately the same CPU-time as the other mesh generation steps with the exception of insertion of viscous layers.

Table 4.1 presents the statistics of a series of meshes improved by the presented method. The Table provides a good picture of average efficiency of the untangling method. It represents a set of industrial geometries of various complexity. These geometries were used for testing and validating the approach implementation. Test cases are sorted by the total number of mesh cells, which reflects the degree of complexity of the geometries. The Table shows that nearly all concave cells were corrected in cases with small numbers of

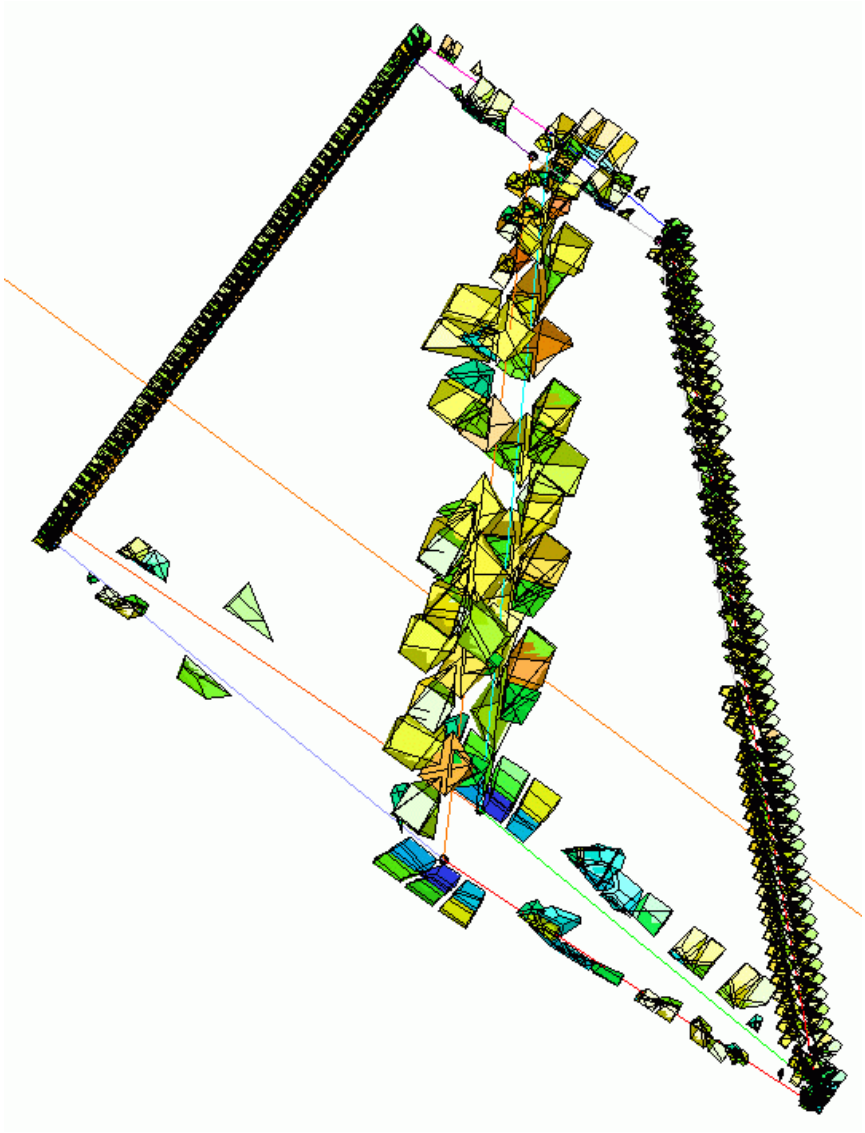


Figure 4.15: Concave cells near a missile wing.

cells. Although larger meshes are not as easy to untangle completely, percentage of the remaining concave cells is very low and ranges between 0.001 and 0.002 percent.

In general, the combined smoothing/untangling/optimization method presented in the thesis has been proved to be a highly efficient tool for quality improvement of unstructured hexahedral meshes. Among multiple real industrial 3-D geometries meshed using the presented method, only a few had concave cells remaining after the quality improvement step. The method has shown itself to be very robust and it does not break down provided that the input geometry is valid.

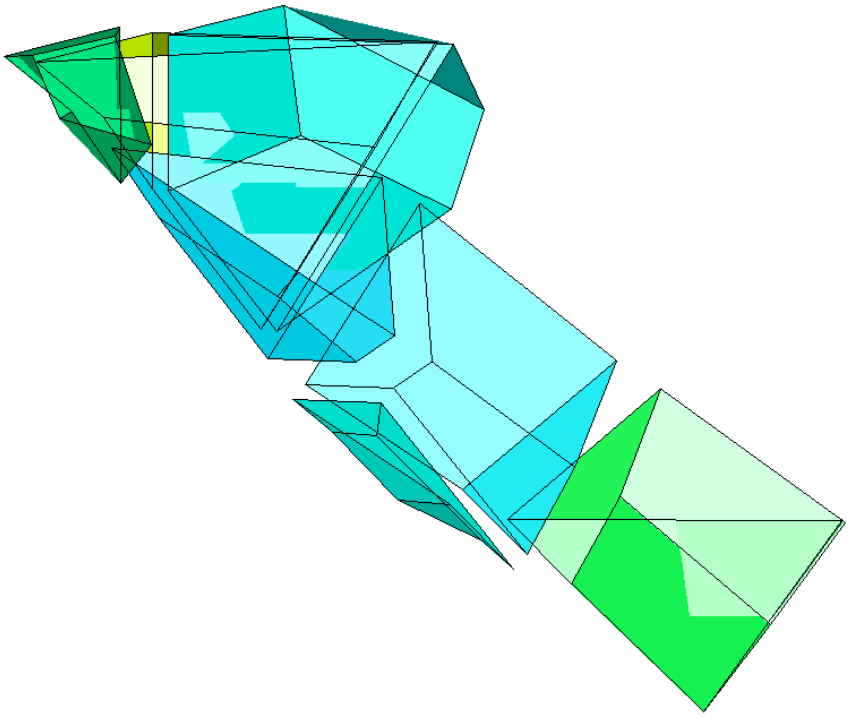


Figure 4.16: Example of a cluster of concave cells.

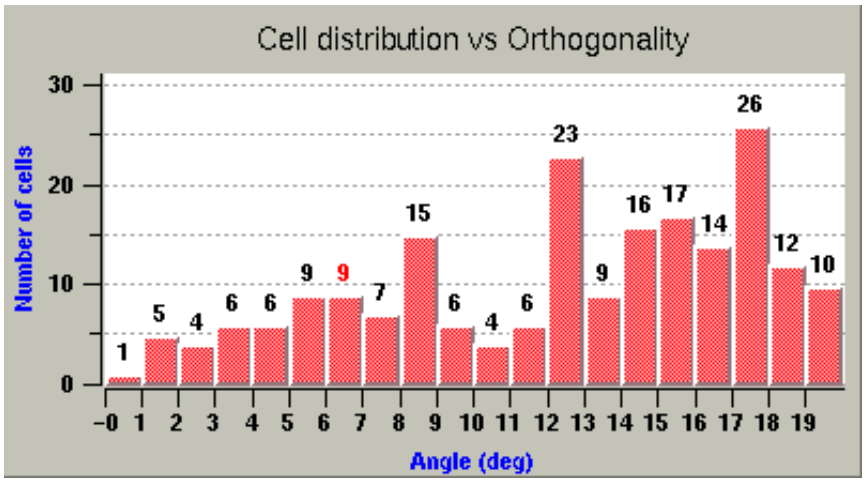


Figure 4.17: Orthogonality distribution in low angle range.

4.6 Conclusions and recommendations

The Chapter presented a new approach for quality improvement of hexahedral meshes based on a low-cost simplicial untangling algorithm and a structural-analogy-based optimization methodology. The results prove high efficiency of the combined approach at relatively

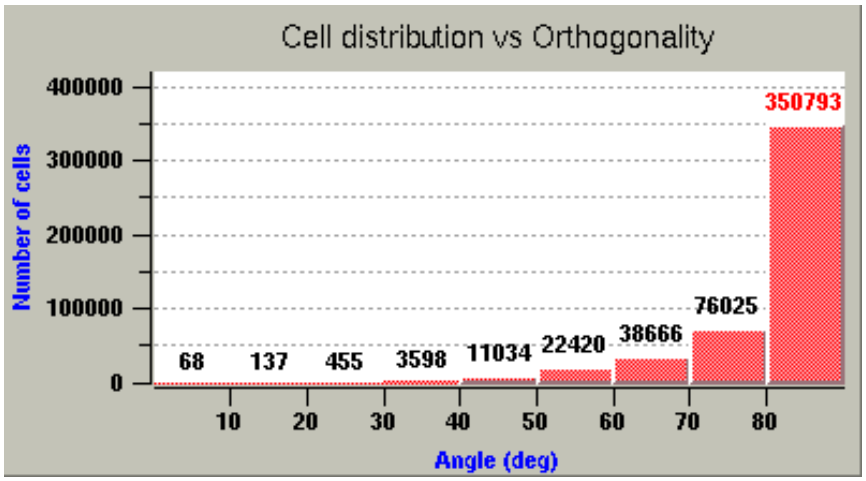


Figure 4.18: Orthogonality distribution before quality improvement.

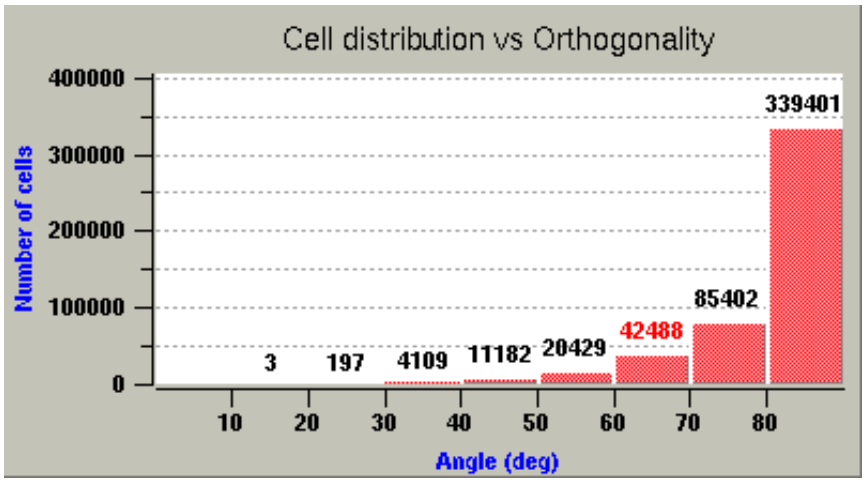


Figure 4.19: Orthogonality distribution after quality improvement.

low computational costs. Only the optimization part of the combined approach takes significant amounts of computational time. On the contrary, both the untangling and smoothing are very fast and computationally inexpensive. The combination is not only capable of untangling concave cells, but also improving their quality without optimizing them directly, due to the averaging procedure described in above sections. This allows to save a significant amount of computational time as compared to optimization-based untangling procedures which do not benefit from linear nature of the untangling problem.

We conclude that the combined approach is capable of improving mesh quality much more efficiently than stand-alone optimization tools or optimization-based untangling algorithms. The main advantage of the combined approach is that similar mesh quality improvement can be achieved in shorter computational time as compared to optimization-based approaches. This is due to the fact that the fast combination of untangling and Laplacian

Test case	The number of mesh cells	Concave cells before quality improvement		Concave cells after quality improvement	
1	54864	153	0.279%	0	0.0%
2	72764	86	0.118%	0	0.0%
3	75070	120	0.160%	0	0.0010%
4	98056	299	0.305%	1	0.0017%
5	115662	246	0.213%	2	0.0%
6	125477	161	0.128%	0	0.0%
7	198362	338	0.170%	0	0.0%
8	235370	497	0.211%	0	0.0%
9	265913	884	0.332%	3	0.0011%
10	301298	901	0.299%	0	0.0%
11	440678	1344	0.305%	1	0.0002%
12	497672	3292	0.661%	0	0.0%
Missile	559169	5664	1.013%	0	0.0%
14	773268	2633	0.341%	0	0.0%
15	1130575	2205	0.195%	5	0.0004%

Table 4.1: Comparisons of population of concave cells before and after application of the combined approach.

smoothing presented here typically corrects at least eighty to ninety percent of concave cells in a mesh before any optimization is applied. Therefore, the method usually untangles the majority of concave cells in a mesh very quickly, thus providing valid meshes in minimal time.

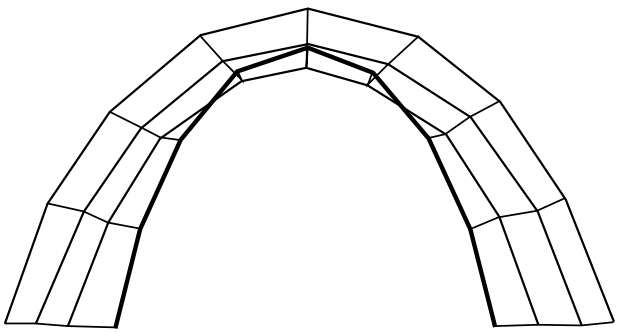


Figure 4.20: Overlaid mesh produced by Laplacian smoothing.

Despite its high efficiency, the combined approach still leaves sufficient room for improvement. The Laplacian smoothing applied at the smoothing-untangling step sometimes ex-

hibits undesirable behaviour in vicinity of concave geometrical items and may lead to self-overlaid meshes in such regions (Figure 4.20). To address this issue, the Laplacian smoothing procedure can be upgraded to its "smart" version [41] which forbids node motion should it result in either appearance of new concave cells or degradation of quality of surrounding cells. In other words, if a node displacement provided by the Laplacian smoothing decreases mesh quality, the node retains its location. Such a modification may decrease the overall time spent on mesh untangling.

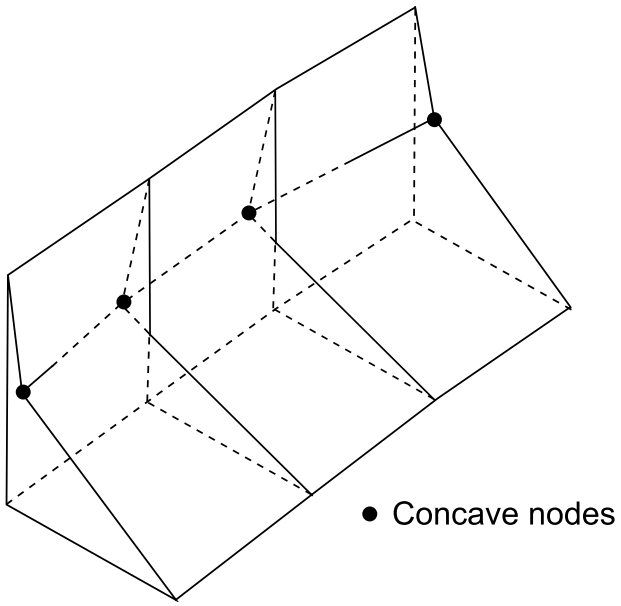


Figure 4.21: A row of adjacent concave cells.

Performance of the untangling procedure itself can be improved as well. Namely, when two cell vertices which share a common edge are tangled, they may not always be corrected successfully. Example of a group of concave cells posing such problem is depicted in Figure 4.21. This happens as respective balls and kernels interfere, and the final result depends on the particular mesh configuration. A possible improvement in this direction is development of some kind of an edge-based untangling procedure which analyzes cells surrounding both edge vertices. Finally, the optimization procedure can be instrumented with other quality metrics in order to provide better control of particular properties being optimized. For instance, measures can be introduced based on minimal dihedral angle in a cell or on flatness of cell faces. These improvements may further increase efficiency of the combined procedure described in the Chapter.

Chapter 5

High-quality Viscous Mesh Generation

5.1 Introduction

Multiple physical problems of industrial interest that involve spatial distribution of certain quantities exhibit strong gradients in particular directions, while gradients in other directions remain moderate in same regions. In order to perform successful numerical simulations and obtain high accuracy results, respective computational meshes must be sized according to length scales necessary for capturing these gradients properly. Therefore, anisotropically graded meshes with sufficiently small cell sizes in appropriate directions can significantly reduce computational costs. In particular, requirements for accurate viscous computations have grown significantly over the last two decades. Two main reasons are the continuous increase of computational power of new computers and the intensive development of new technologies in this area. The combination of these two factors creates a demand to search for new and efficient methods for viscous computations, namely the mesh generation methods and viscous flow solvers.

Up-to-date methods for high Reynolds number computations as well as integrated physical models often require high quality computational meshes to be used. This primarily concerns control of the position of the first mesh point close to wall and smooth variation of cell sizes along wall normals. The latter includes two aspects. On one hand side, the stretching ratio in the viscous layer h_{i+1}/h_i (the ratio of sizes of two adjacent layers of viscous cells measured in normal-to-wall direction) must range around a certain value (1.2-1.3), especially in the near-wall region. Maintaining the first cell size is equally important. Therefore, an automatic viscous mesh generation tool capable of generating meshes that satisfy the above requirements is necessary.

A new method for viscous layer insertion into an inviscid mesh generated by means of unstructured hexahedral non-conformal approach is presented in this chapter [76]. First,

the problem is formulated in detail and the respective solution procedure is described. Following that, the results of application of the new method to test cases of industrial interest are presented; properties of the method such as efficiency and robustness are discussed; final conclusions are drawn and recommendations for future developments are proposed.

5.2 Insertion of viscous layers

In framework of the overlay methodology, viscous layers are generated via tangential refinement of cells adjacent to solid walls (further referred to as buffer cells) as shown in Figure 5.1. The refinement process enables to generate a sufficient number of highly stretched cells along solid walls of computational domain. Vertices of these cells are redistributed along wall normals according to specified first cell size and stretching ratio.

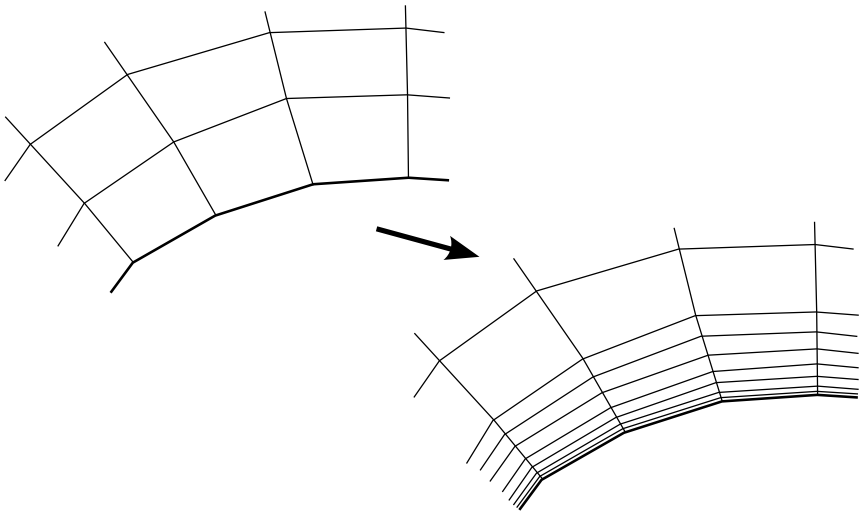


Figure 5.1: Viscous layer insertion.

The layer insertion procedure is illustrated in Figure 5.2. For the sake of clarity, it is convenient to assume that a reasonably large number of layers is specified. Refinement is performed in a number of steps. This number depends on the number of layers to be inserted. The green marks in Figure 5.1 point out cells which are subject to refinement in each step. First, each buffer cell is divided into two new ones. These are also refined resulting in four layers. At the next step, the three lower cells are subdivided and the uppermost cell is not. At this point, seven layers are generated. Similarly, the five lower cells are refined at the next step and the two uppermost cells are not. The process is continued until the specified number of layers is generated. The upper cells are not refined due to specifics of the overlay approach.

After a sufficient number of viscous cells is inserted, cell nodes are redistributed between

buffer cell nodes in normal direction according to the specified first cell size and stretching ratio (Figure 5.3):

$$h_0 = \delta_0, \; h_{i+1} = a \cdot h_i \tag{5.1}$$

Here, δ_0 and a denote the first cell size and stretching ratio respectively. Should it be necessary, the stretching ratio may be adjusted for a particular layer to fit the respective buffer cell.

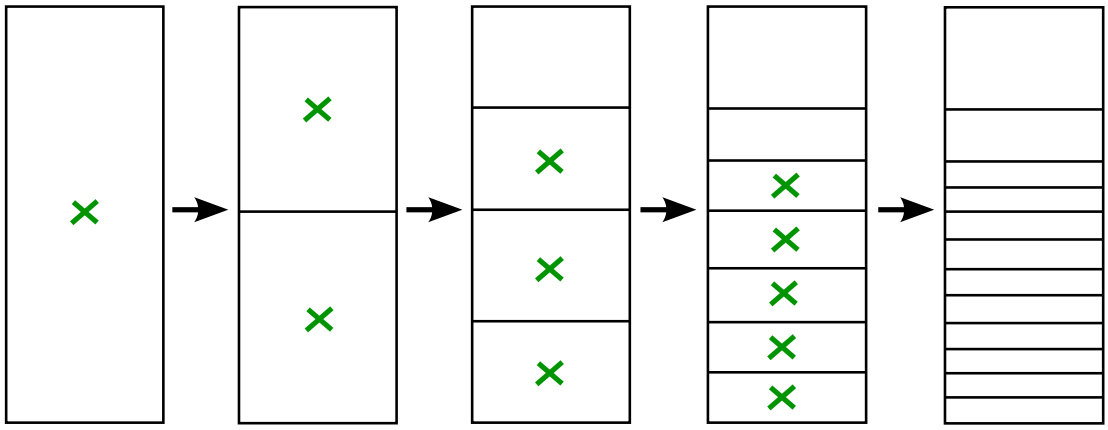


Figure 5.2: Tangential refinement.

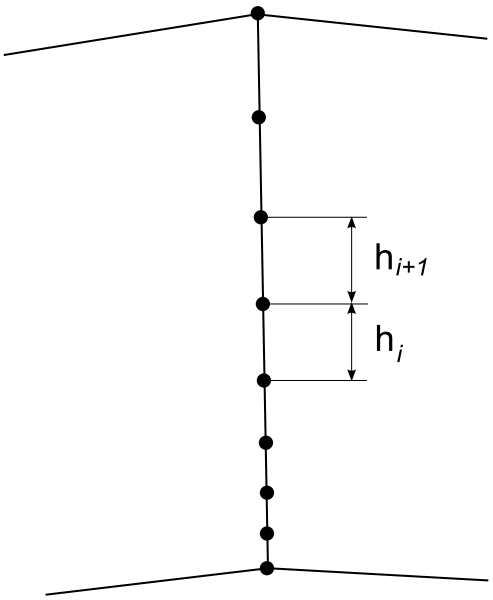


Figure 5.3: Node redistribution.

5.3 Problem statement

One obvious limitation of the method presented above is the size of buffer cells prior to refinement. These initial sizes result entirely from the Euler mesh generation process. Regardless of how many layers are specified and what their parameters are, the total thickness of a layer is limited by the size of the underlying buffer cell (cell a in Figure 5.4). Therefore, layer parameters must be adjusted for the layer to be able to fit in the available space. Moreover, cell b is approximately of the same size as a. Thus, the more layers are inserted and the lower the stretching ratio is, the greater is the discontinuity in sizes of cells b and c. Successful viscous computations require the transition of normal cell sizes between viscous and external inviscid cells to be sufficiently smooth (see Figure 5.5). This is almost unattainable without resizing buffer cells and their neighbors.

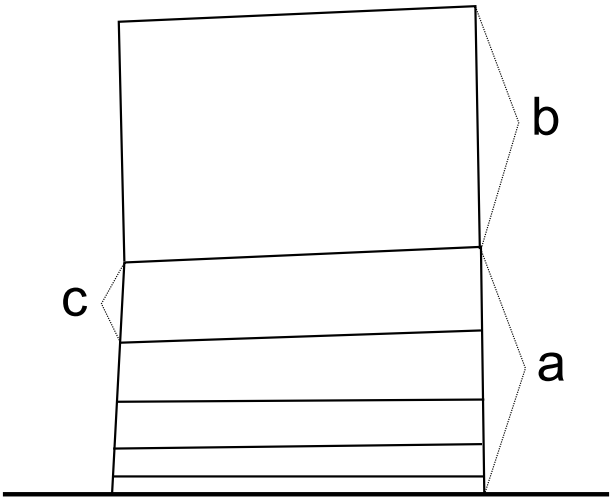


Figure 5.4: Viscous layer scheme.

These requirements create a necessity to instrument the layer insertion procedure with a pre-processing step that resizes buffer cells in normal direction. This inflation procedure should increase normal sizes of buffer cells such that the resulting viscous layer satisfies the smoothness conditions.

The problem of insertion of a viscous layer with predefined cell size distribution into an Euler mesh can be formulated as follows:

Problem. *There exists an unstructured hexahedral non-conformal mesh with a fixed boundary. Let $\{C_B\}$ be the set of mesh cells adjacent to solid boundaries along which layers of anisotropic cells are to be generated; $\{V_I\}$ - the set of vertices of cells in $\{C_B\}$ not belonging to boundaries; $\{E_N\}$ - the set of edges of cells in $\{C_B\}$ normal to boundaries (Figure 5.6). A new set of positions for nodes of $\{V_I\}$ must be found, such that sizes of edges in $\{E_N\}$ are sufficiently large to provide smooth cell size transition between the viscous and inviscid*

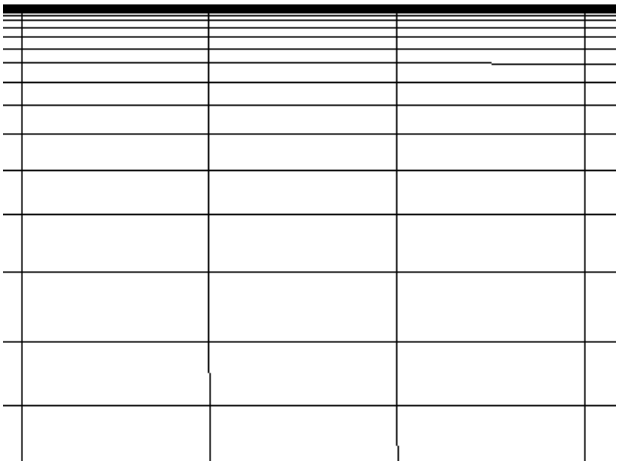


Figure 5.5: Smooth viscous mesh.

sub-meshes. Cells in $\{C_B\}$ *must be tangentially refined and nodes of new viscous cells must be redistributed in normal-to-wall direction.*

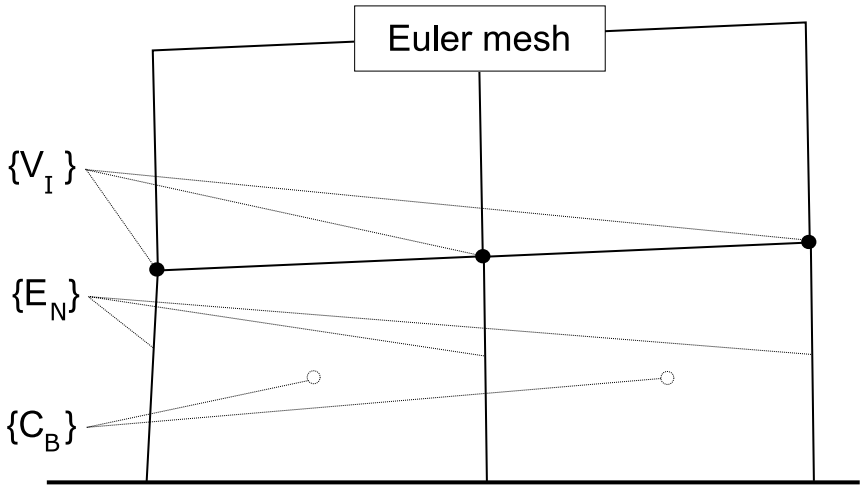


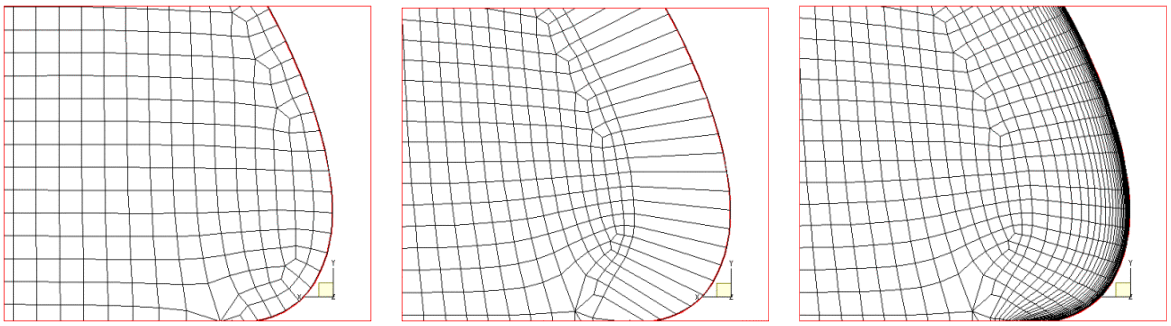
Figure 5.6: Problem formulation.

These positions are meant for reserving enough space to fit the desired viscous layer and for providing good conditions for smooth cell size transition after refinement and node redistribution. Mesh quality must be maintained at a level as high as possible. Cells in $\{C_B\}$ are further referred to as *buffer cells*.

In general, it is difficult to satisfy all three parameters (the first cell size, the stretching ratio and the number of layers) for an arbitrary case. It is impossible to say in advance whether the specified parameters are optimal or not. However, it is possible to correct the parameters during layer insertion. The correction improves quality of the resulting layer,

while preserving all of its key features at the same time. For instance, the number of layers can be modified if it is too large or the stretching ratio can be adjusted in the external part of the layer, while the parameters in the near-wall region remain fixed.

As usual, the desired total thickness of the specified layer is larger (sometimes significantly) than the average normal size of buffer cells in the Euler mesh. This difference in size introduces a local anisotropic direction (normal to boundary), in which these cells must be extended or *inflated* (Figure 5.7). Such mesh modification may represent quite a large deformation. Displacements of internal mesh vertices can be of an order of several isotropic cell sizes in the region. During testing, ratio of the initial and inflated buffer cell sizes increases by several fold. The inflation process is driven by Laplacian smoothing, which usually leads to appearance of concave and even invalid cells (i.e., cells with negative volumes). Presence of such cells is unacceptable and a posteriori quality improvement step is required.



(a) Initial mesh. (b) Inflation. (c) Insertion.

Figure 5.7: Inflation-insertion scheme.

Therefore, a coupled procedure capable of gradually inflating layer zones (i.e., increasing edge sizes in $\{E_N\}$) as well as untangling new concave or negative cells and locally improving mesh quality in problematic regions is to be developed. This procedure must be followed by the layer insertion process described above.

5.4 Solution methodology

Methodology for solving the earlier formulated problem of layer inflation is described below. After the desired viscous layer parameters are specified, the total anticipated layer thickness can be computed. For the constant stretching ratio a ($a > 1$), the first viscous cell size δ_0 and the number of layers N, we obtain the total layer thickness T as:

$$T = \delta_0 \cdot \frac{a^N - 1}{a - 1} \tag{5.2}$$

This thickness is used to define new target sizes to be assigned to edges in $\{E_N\}$. Following that, an iterative procedure developed for inflating cells in $\{C_B\}$ in normal direction with simultaneous quality control is applied. This coupled procedure combines Laplacian smoothing for buffer layer inflation with an efficient untangling/optimization module. The details of this approach are described in the next sections.

5.4.1 Possible implementations

Two potential schemes to solve the problem (Figure 5.8) are considered. The first one is to apply a modified version of Laplacian smoothing instrumented with an internal quality filter, which is usually referred to as "smart" Laplacian smoothing. In this approach, mesh vertices are moved to new positions only if the move does not lead to decrease in local mesh quality.

The second possibility is to apply the original form of Laplacian smoothing (without quality check) followed by a posteriori mesh quality improvement procedure. The mesh is smoothed for a certain number of Laplacian sweeps, after which its quality is locally improved.

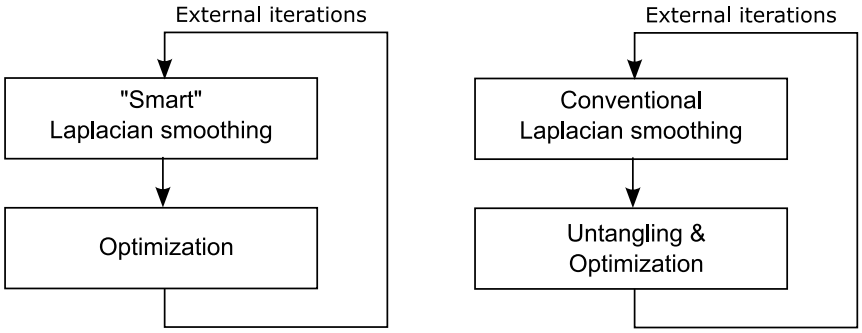


Figure 5.8: Alternative inflation algorithms.

The difference between two approaches which is shown in Figure 5.8 is described next. In the first approach, mesh quality is controlled during each inflating iteration. In the second one, conversely, mesh quality recovery follows each inflating iteration. A combination of smoothing and optimization leads to more efficient quality control during the inflation process. However, the efficiency of inflation itself declines as an additional condition of non-decreasing local quality is applied to each vertex subject to Laplacian smoothing (inflation). On the contrary, the second approach performs these operations within every external iteration sequentially. This leads to more efficient inflation, as weaker quality limitations are implied for intermediate meshes.

The ultimate goal is to approach the target size of layer zone as closely as possible. Apparently, the main limitation of this goal is the requirement that the final mesh must be valid and its quality must be sufficiently high for further use by a flow solver. As long as powerful quality improvement tools are available, the second approach appears to be more efficient.

5.4.2 Optimal approach

The inflation procedure is an iterative process. During each iteration, target sizes of edges in $\{E_N\}$ are multiplied by a factor ranging between 1.4 and 2.5. These values were identified empirically. Increase of target sizes by a factor larger than 2.5 negatively affects stability of the inflation process. At the same time, target sizes of adjacent Euler mesh cells (cell b in Figure 5.4) are slightly decreased.

After new target sizes of mesh edges are fixed, Laplacian smoothing is applied. A weighted form of smoothing is employed, with weights equal to reciprocals of respective target edge sizes. The smoothing inflates buffer cells but simultaneously creates concave or inverted cells that appear during this process. This negative effect of smoothing on mesh quality depends on the complexity of geometries being meshed. To solve this problem, smoothing is followed by the combined untangling/optimization procedure (see previous chapter) which eliminates concave cells from the mesh.

Iterations are repeated until at least one of termination conditions is satisfied. If a layer reaches its specified size, the inflation process is completed. Besides, there are many situations when further inflation appears to be impossible due to quality reasons. Namely, if the untangling/optimization procedure does not succeed in removal of all concave cells at latest iteration, the process is terminated and a valid mesh saved at the previous iteration is used.

Algorithm 4 provides a detailed description of the inflation methodology as implemented in the mesh generator. The inflation procedure is followed by the layer insertion process as described in Section 5.2.

5.5 Results and discussion

Performance of the viscous layer module developed according to the described approach is demonstrated in this section using one 2-D and three 3-D examples.

The first example contains the 2-D internal geometry of a cooled turbine blade. Figure 5.9 depicts a common view of the domain and the final viscous mesh. The initial Euler mesh contains 10,252 cells, while the final viscous mesh contains more than 40,000 cells. The following parameters of the viscous layer were specified: the first cell size is 10^{-5}, the stretching ratio is 1.3, and the number of layers is 20.

Figure 5.10 shows a comparison between a fragment of the initial Euler mesh and the corresponding fragment of the viscous mesh. Details of the resulting viscous mesh are also illustrated in Figures 5.11 and 5.12. This geometry is internal and the mesh boundary is almost convex (except for inner orifices). This creates nearly perfect conditions for the inflation module, as it is based on smoothing algorithms which are generally more efficient in convex domains. Both Figures mentioned above show good cell size transition between

Algorithm 4 Viscous layer insertion.

Begin the inflation *(mesh M, boundary B)*

Identify:

$\{C_B\}$ - the set of cells adjacent to solid walls (buffer cells);
$\{V_B\}$ - the set of vertices of cells in $\{C_B\}$ belonging to the boundary B (boundary vertices);
$\{V_I\}$ - the set of vertices of cells in $\{C_B\}$ not belonging to the boundary B (internal buffer vertices);
$\{E_N\}$ - the set of edges of cells in $\{C_B\}$ that connect vertices of $\{V_B\}$ and $\{V_I\}$ (normal edges);

Loop external iterations $i_{IE} = 1, ..., N_{IE}$

If the total thickness of specified layer is not larger than the average actual size of edges in $\{E_N\}$, ***Exit*** external iterations

Inflate target sizes of edges in $\{E_N\}$ by the inflation factor (1.4-2.5)

Apply Laplacian smoothing to the surface mesh using new target sizes

Apply the 2-D/Surface version of the combined optimization approach (Algorithm 3) to the surface mesh

If concave faces remain in the surface mesh
Reload the mesh resulting from the previous external iteration
Exit external iterations

Apply Laplacian smoothing to the volume mesh using new target sizes
Apply the combined optimization approach (Algorithm 3) to the volume mesh
If concave cells remain in the mesh
Reload the mesh resulting from the previous external iteration
Exit external iterations

End external iterations
Refine cells in $\{C_B\}$ until the required number of viscous layers is generated
Redistribute nodes in the layer along respective edges in $\{E_N\}$ according to specified layer parameters
End.

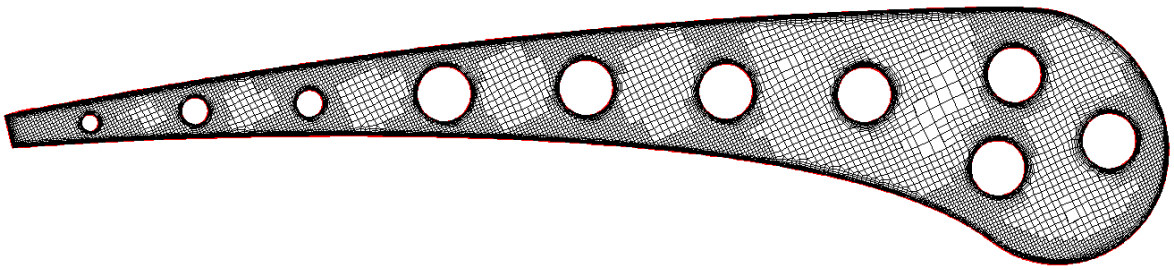
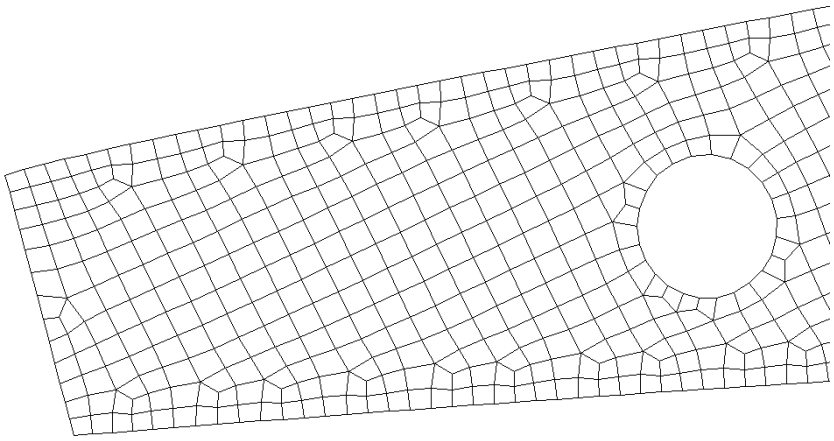


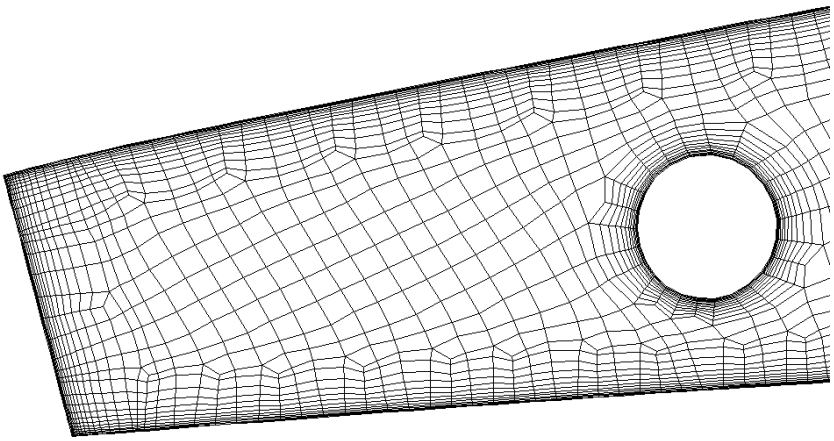
Figure 5.9: Mesh on a cooled blade.

viscous and inviscid cells. Comparison of mesh quality diagrams of both the initial and final meshes shows no decrease of mesh quality at all.

This example reveals one important property of the method. The efficiency of the inflation



(a) Initial Euler mesh.



(b) Final viscous mesh.

Figure 5.10: Comparison of the initial Euler mesh and the final viscous mesh on the cooled blade.

module is not uniform along each surface but depends on cell sizes of Euler mesh in the same region. If cells of different refinement levels (with sizes differing by at least the factor of two) are snapped to the same surface, then the thickness of viscous layer generated on such surface is also not uniform. This effect is due to local action of the smoother, and the resulting absolute displacements depend on sizes of surrounding cells. However, further fine-tuning and algorithm modification may decrease the impact of this problem.

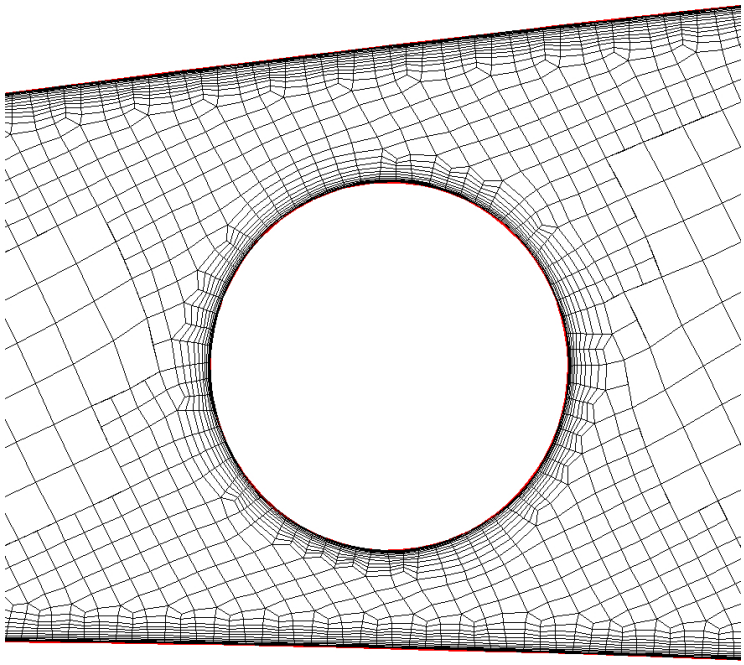


Figure 5.11: Viscous mesh on the cooled blade; Fragment 1.

The next example is the external 3-D geometry around the Ahmed body. The initial Euler mesh consists of 168,057 cells and the viscous mesh contains more than 1,100,000 cells after insertion of 25 layers with the initial cell size of 10^{-3} and the stretching ratio of 1.32. The corresponding geometrical model is shown in Figure 5.13.

A mesh is generated between the model and the external boundary. The latter is a simple box consisting of a bottom ground plane, a vertical mirror plane, and four artificial external boundaries.

Figure 5.14 demonstrates a mesh view in a cross-section parallel to the mirror plane. This case demonstrates performance of the method on a relatively simple external 3-D geometry. In general, external cases are more difficult for the inflation module to handle, as such geometries contain a lot of locally concave (with respect to the mesh) regions. Therefore, implementation of a Laplacian smoother requires additional modifications in order for it to be efficient on such geometries. Comparison of diagrams of mesh quality before and after layer insertion shows no decrease in mesh quality.

Figures 5.15 and 5.16 depict two fragments of the viscous mesh in vicinity of the gap between the body and the ground plane.

The next example is the internal 3-D geometry of a volute. The initial Euler mesh contains 229,000 cells and the viscous mesh contains more than 1.7 million cells after insertion of 17 layers with the initial cell size of 10^{-2} and the stretching ratio of 1.3. The corresponding geometrical model is shown in Figure 5.17. Figures 5.18 and 5.19 depict two fragments of

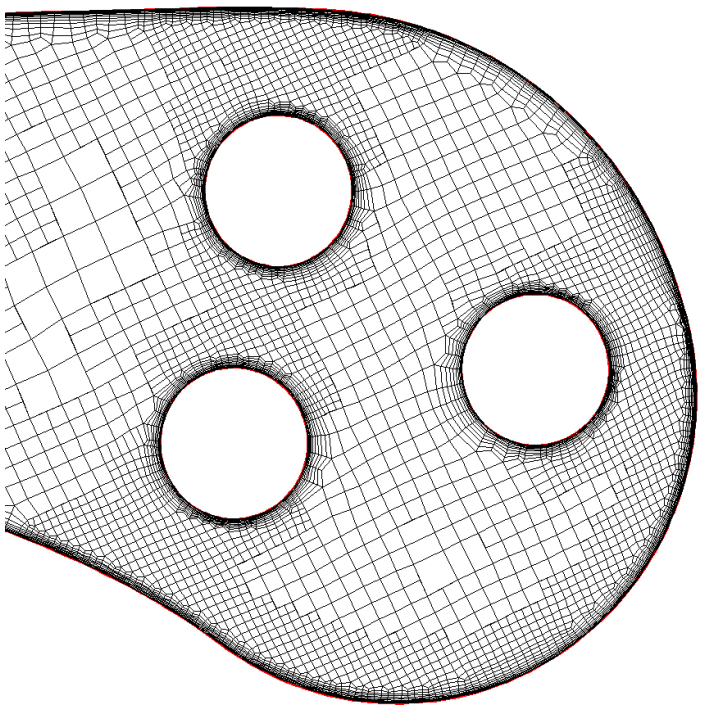


Figure 5.12: Viscous mesh on the cooled blade; Fragment 2.

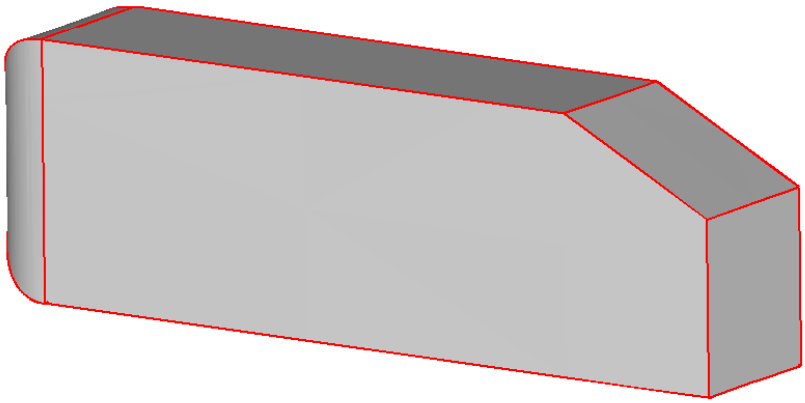


Figure 5.13: Ahmed body geometry.

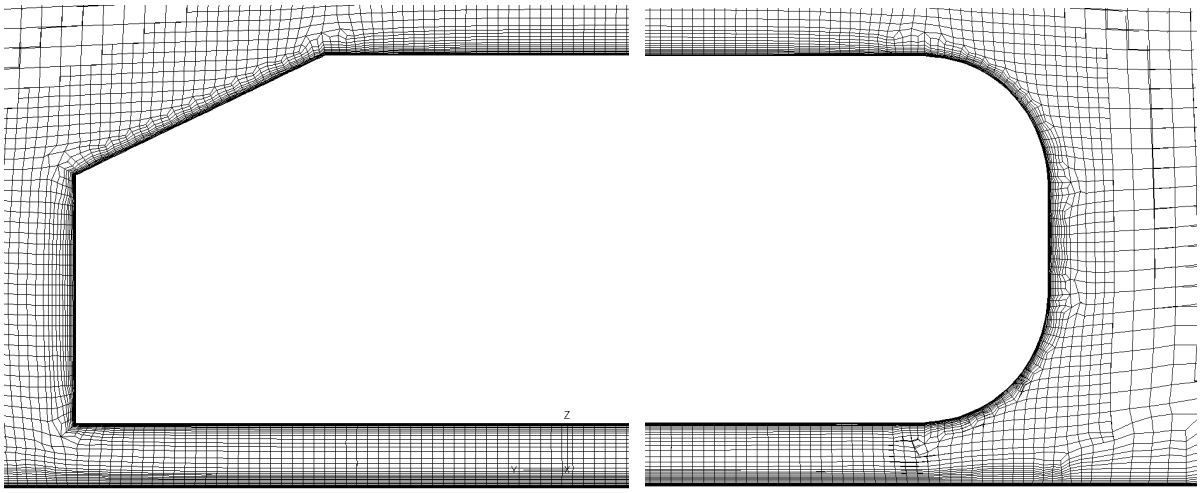


Figure 5.14: Mesh cross-section along the Ahmed body center-plane.

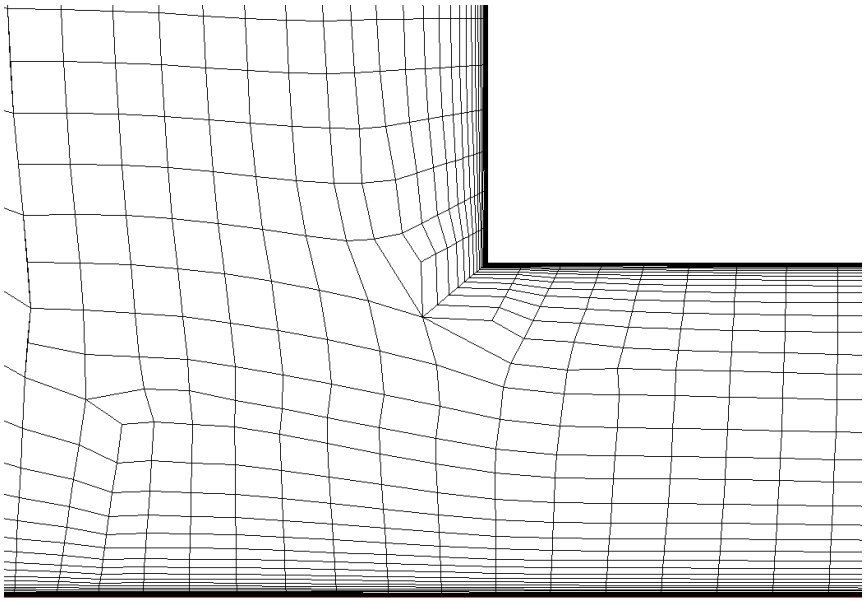


Figure 5.15: Viscous mesh on the Ahmed body; Fragment 1.

the viscous mesh. This internal geometry poses minimum problems for the viscous layer generation procedure.

The last example is a viscous mesh generated for the external wing-body configuration DLR-F4 [78]. The corresponding geometrical model is shown in Figure 5.20. Unlike the Ahmed body case, this configuration represents a more complex problem for the viscous layer insertion module due to presence of highly concave regions in the geometry, i.e. the rear tip of the fuselage, the wing tip and the wing trailing edge.

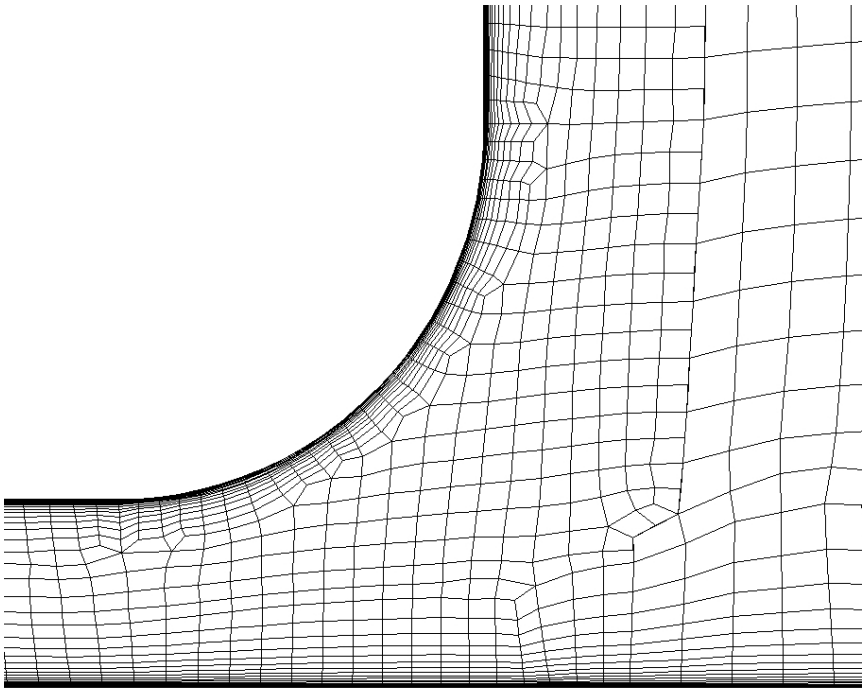


Figure 5.16: Viscous mesh on the Ahmed body; Fragment 2.

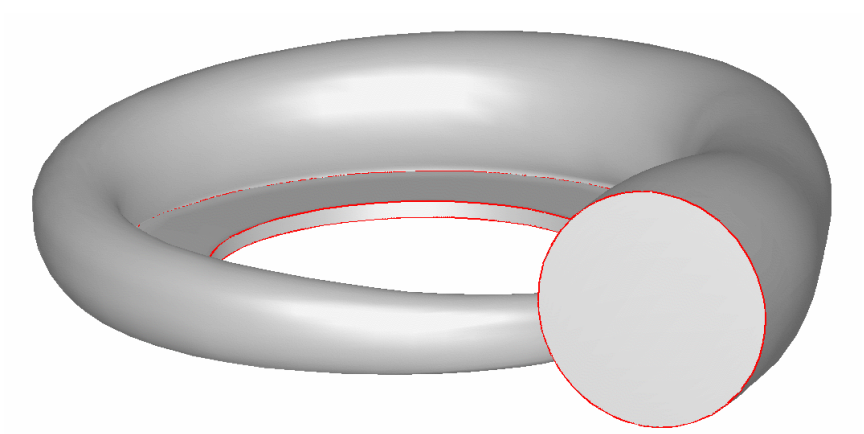


Figure 5.17: Geometry of a generic volute.

The initial Euler mesh contains 662,744 cells while the final viscous mesh contains more than 3.5 million cells. The parameters of inserted viscous layers are: the first cell size is 10^{-3}, the stretching ratio - 1.3, and the number of layers 25.

Figure 5.21 shows a common view of a mesh cut across the fuselage and a major part of the wing. Smooth cell size transition between the viscous and inviscid meshes can be seen in the Figure clearly. Non-uniform viscous layer thickness related to non-uniform cell sizes

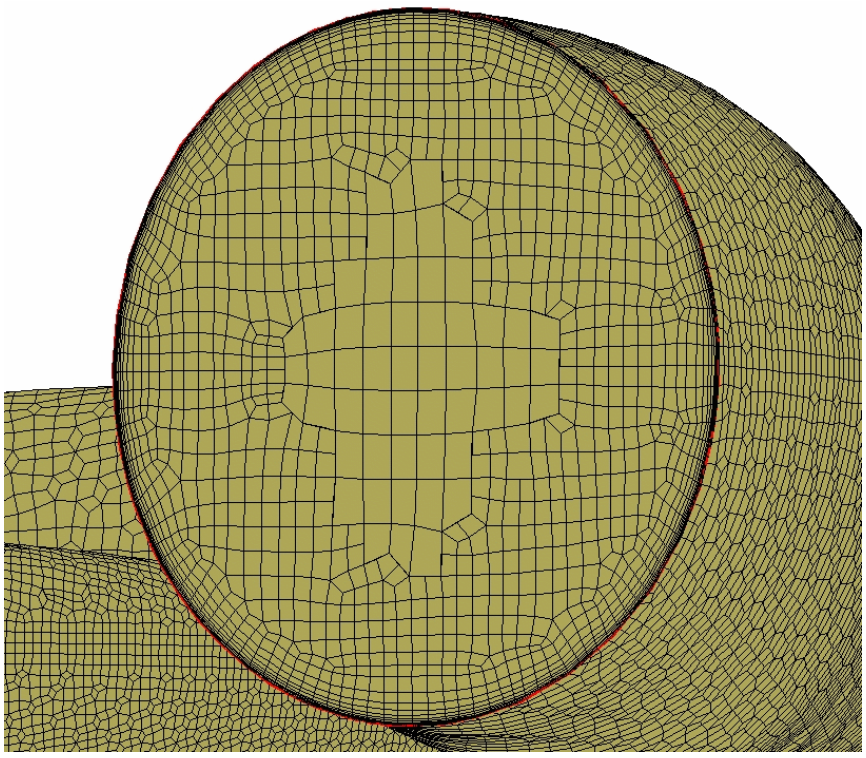


Figure 5.18: Footprint of a viscous mesh on the outlet surface of the volute.

along the fuselage surface can also be observed.

Figure 5.22 illustrates good mesh quality in vicinity of the junction between the wing and fuselage. Intersecting viscous layers generated at adjacent surfaces result in an isotropic refinement of the region.

Figure 5.23 depicts close views of a mesh around two main highly concave items of the model: the rear fuselage tip and the wing trailing edge. Both pictures show some decrease in efficiency of the inflation procedure. This is a typical effect of Laplacian-like smoothers near strongly concave boundaries. However, local quality of the mesh tends to increase, as mesh lines become more rounded and smooth. Finally, Figure 5.24 shows two mesh cuts across the wing. The left picture shows a mesh cut near the wing root, and the right one shows a similar mesh cut near the wing tip. The Figure demonstrates good uniformity of layer thickness all over the wing surface. It shows quite poor quality of some cells, which in fact is an effect of the mesh cutting tool that does not cut across cells but rather projects them onto the cutting plane. Figure 5.25a presents a diagram of cell orthogonality distribution in the mesh. The minimal angle of eight degrees appears to be due to the extremely sharp trailing edge of the wing. Figure 5.25b depicts respective distribution of aspect ratios of mesh cells.

Finally, an example of flow computation using viscous meshes generated by means of the

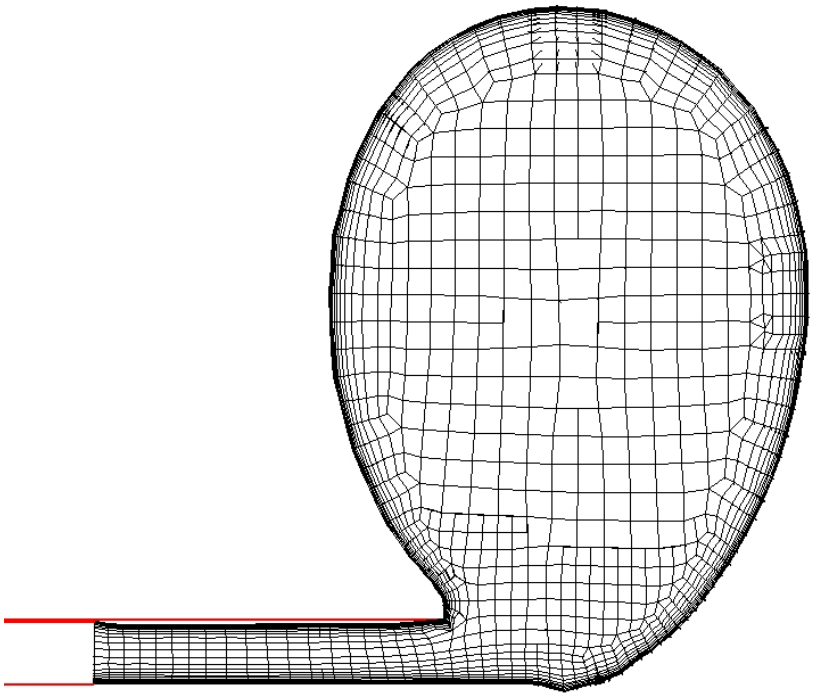


Figure 5.19: Cross-section of a viscous mesh on the volute.

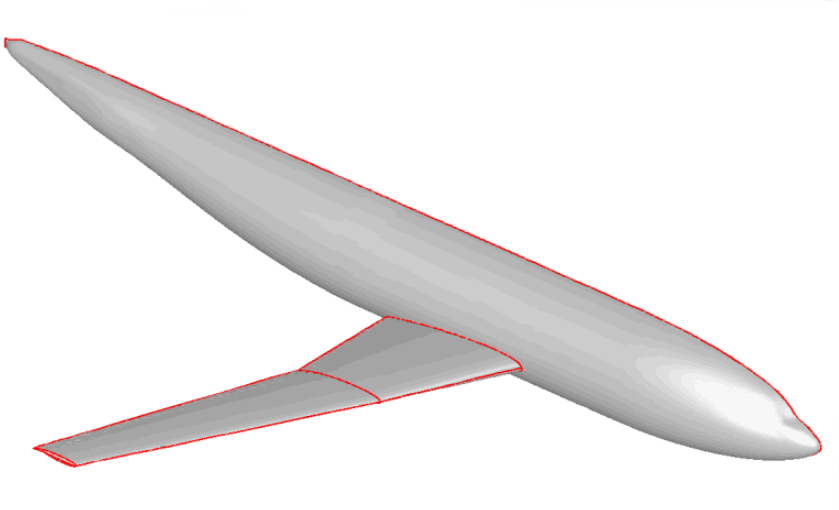


Figure 5.20: DLR-F4 wing-body geometry.

developed methodology is presented. Simulation of the flow around a NACA 64A010 airfoil was performed. Regime of the simulation corresponds to the AGARD CT6 test case [31].

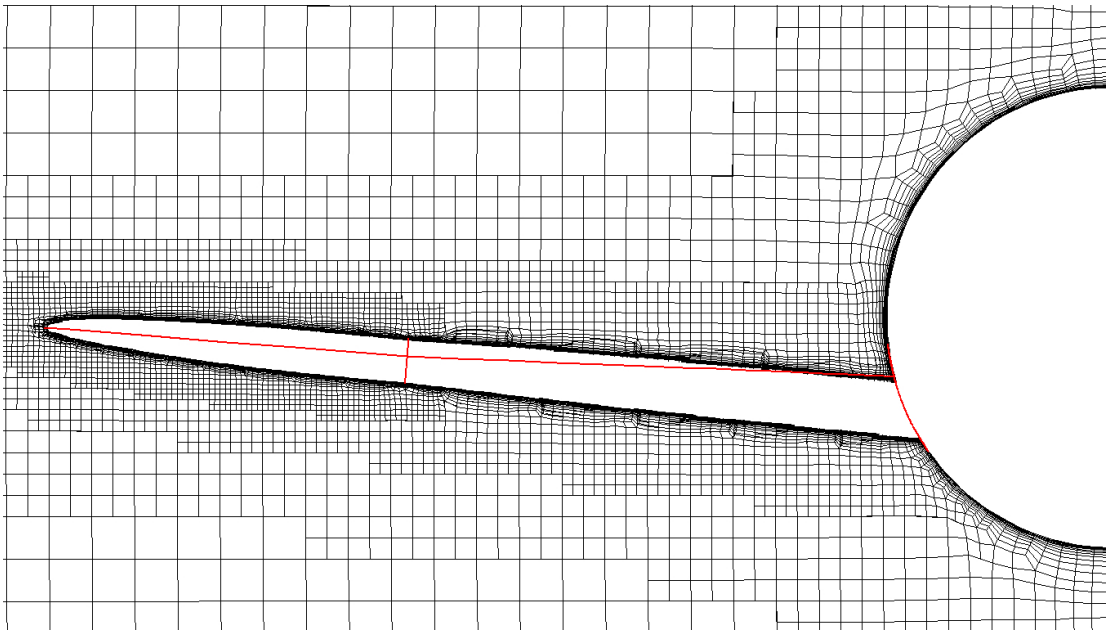


Figure 5.21: A wing-fuselage cross-section of the DLR-F4 configuration.

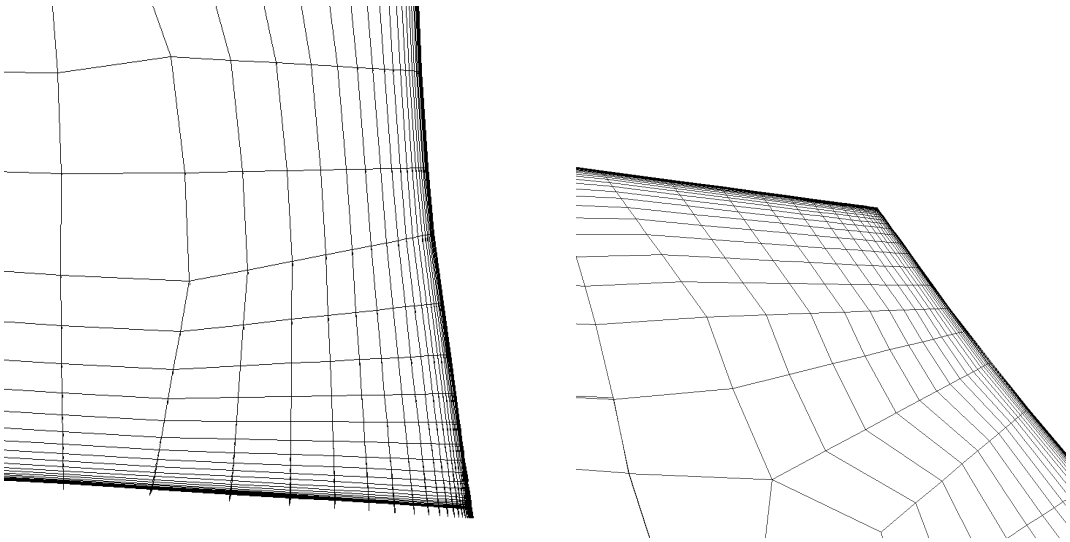
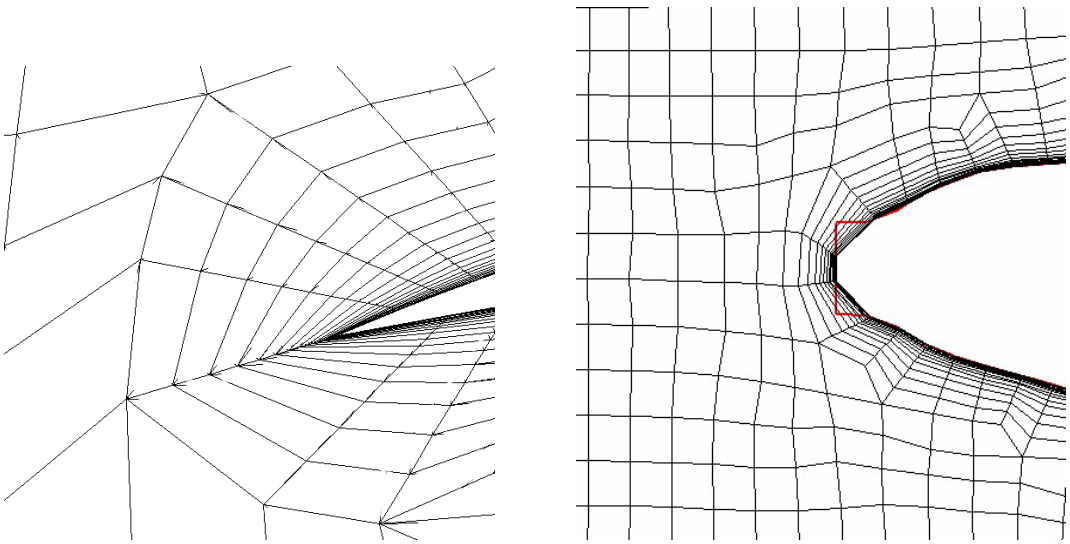


Figure 5.22: Viscous mesh at wing-body junction on the DLR-F4 configuration.

The following flow parameters were used:

$$Re = 12.56 \cdot 10^6,\ M_\infty = 0.796,\ p_\infty = 133912.0\,Pa,\ T_\infty = 293\,K,\ \alpha = -0.21^o \qquad (5.3)$$

A viscous mesh containing 20,194 cells was generated. Twenty five viscous layers with



(a) The wing trailing edge.

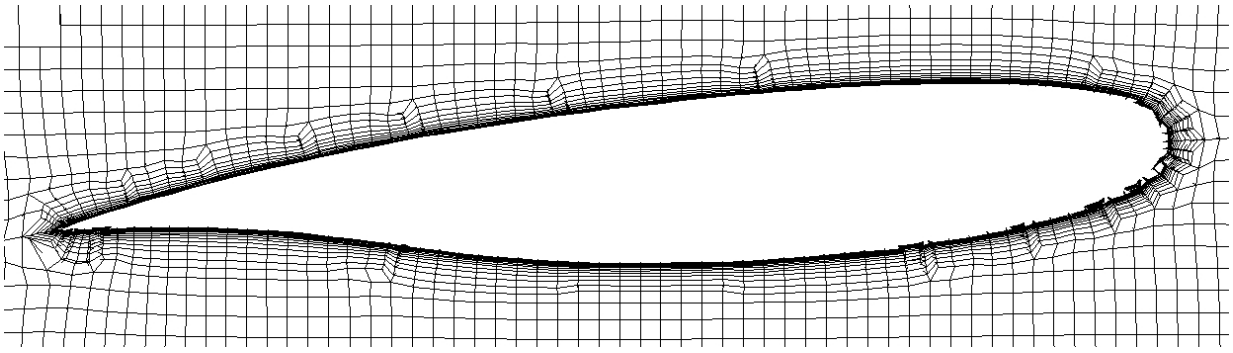
(b) The fuselage rear tip.

Figure 5.23: Viscous mesh near highly-concave geometry on the DLR-F4 configuration.

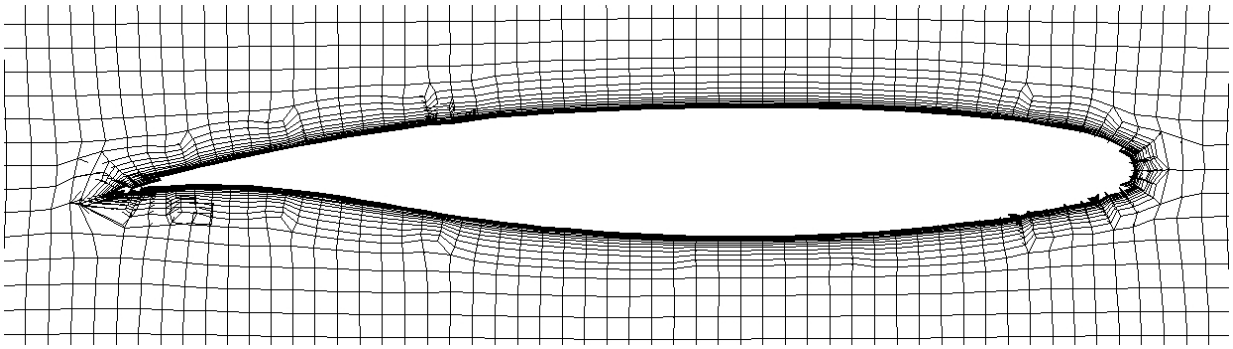
the following parameters were inserted: the first cell size of $5 \cdot 10^{-5}$ and the stretching ratio of 1.2. The inflation procedure preceding the insertion successfully increased the normal sizes of buffer cells by the factor of 3.5 on average. The mesh around this geometry is depicted in Figure 5.26. Mach number distribution over the flow field is presented in Figure 5.27a. It is compared to the distribution obtained on a similar mesh generated by the older approach which performs no inflation (Figure 5.27b). Comparison of orthogonality diagrams corresponding to both meshes is presented in Figure 5.28. Pressure distributions obtained on both meshes were compared with experimental data, as shown in Figure 5.29. Both distributions are identical. This can be explained by the fact that the boundary layer is thinner than buffer cell sizes and the both meshes successfully captured the boundary layer. However, a significant benefit of the new approach can be observed by comparing convergence histories for both computations. The comparison is presented in Figure 5.30. The difference in convergence rate is about four fold. This result can be considered as great success of the new viscous mesh generation approach developed in the thesis.

5.6 Conclusions and recommendations

An efficient methodology for generation of high-quality unstructured hexahedral meshes for high-Reynolds number computations has been developed in the Chapter. This methodology includes two basic stages. The first one is a procedure for inflation of the near-wall layer of



(a) Wing root.



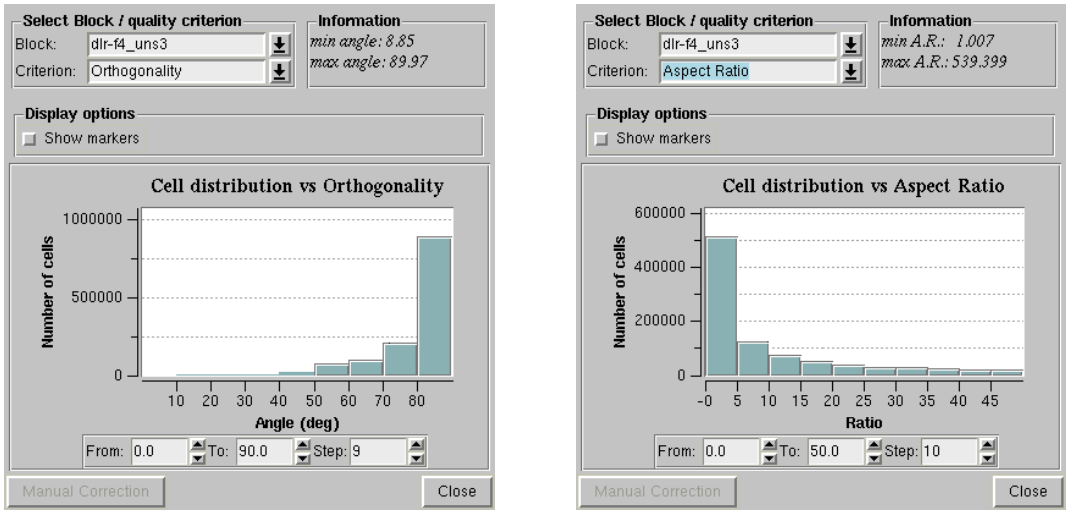
(b) Wing tip.

Figure 5.24: Viscous mesh at two wing cross-sections of the DLR-F4 configuration.

cells of the initial Euler mesh. It is followed by tangential refinement of these cells along walls. The inflation step ensures that smooth size distribution of new anisotropic cells is achieved. The refinement step populates as many tangentially stretched cells as necessary for high-quality viscous analysis.

The methodology has shown good results for a set of real test cases of industrial interest. The inflation of buffer cells allows to satisfy user-specified parameters of viscous layer and to provide a smooth cell sizing in normal-to-wall direction simultaneously. It is important that the inflation procedure shows good performance near concave boundaries provided that the degree of concavity is not too high, even though it is based on Laplacian smoothing. In fact, robust behaviour is ensured even for highly concave geometries: efficiency may be decreased, however, the method does not fail. This enables to successfully deal with many external configurations.

Despite good performance of the inflation methodology, there are certain aspects in which it can be improved. The Laplacian smoothing employed in the methodology experiences



(a) Orthogonality.

(b) Aspect ratio.

Figure 5.25: Distributions of orthogonality and aspect ratio of mesh cells on the DLR-F4 configuration.

loss of efficiency in highly concave regions as, for instance, shown in Figure 5.23. In such regions, the Laplacian smoothing tends to compress the mesh towards the concave geometry, which may potentially lead to invalid overlaid meshes as illustrated in Figure 4.20. This effect is opposite to the effect of layer inflation, as it pulls mesh points towards boundary. A possible solution can be found by applying some kind of local space mapping in order to compensate high surface curvature (Figure 5.31). As Laplacian smoothing is applied locally, motion of any nodes is driven by positions of its neighbors only. Therefore, it is sufficient to apply the mapping to sets of cells attached to one node. Smoothing can be performed in the mapped space, after which the inverse mapping should be applied. Alternatively, application of other smoothing approaches can be attempted.

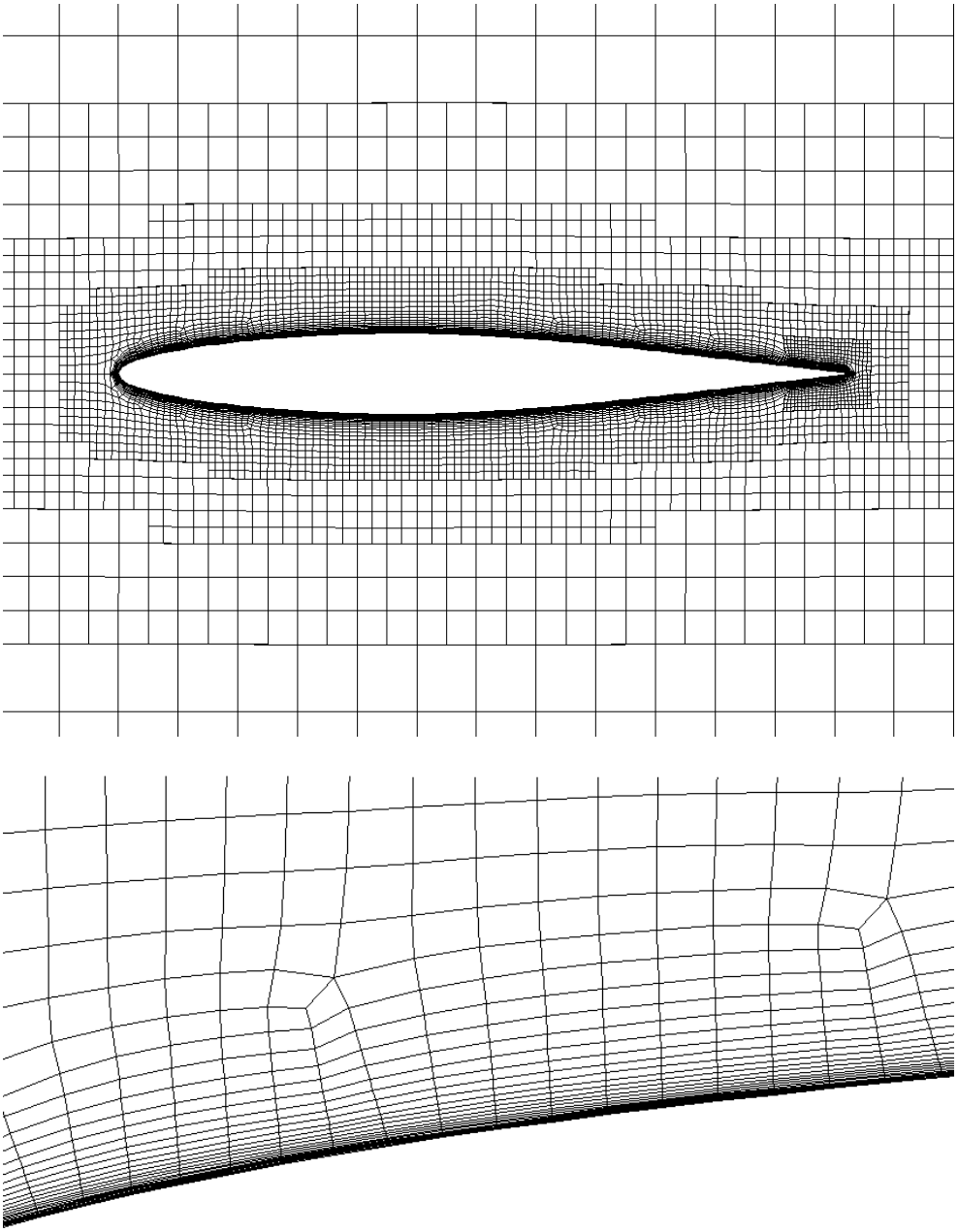
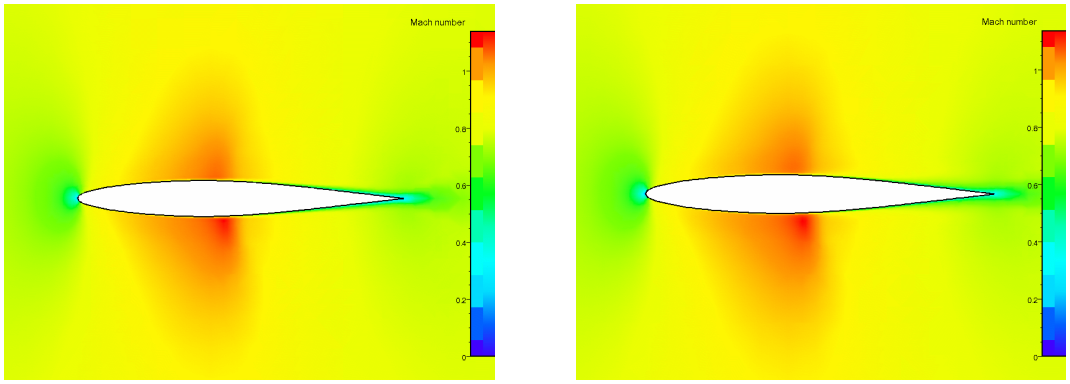
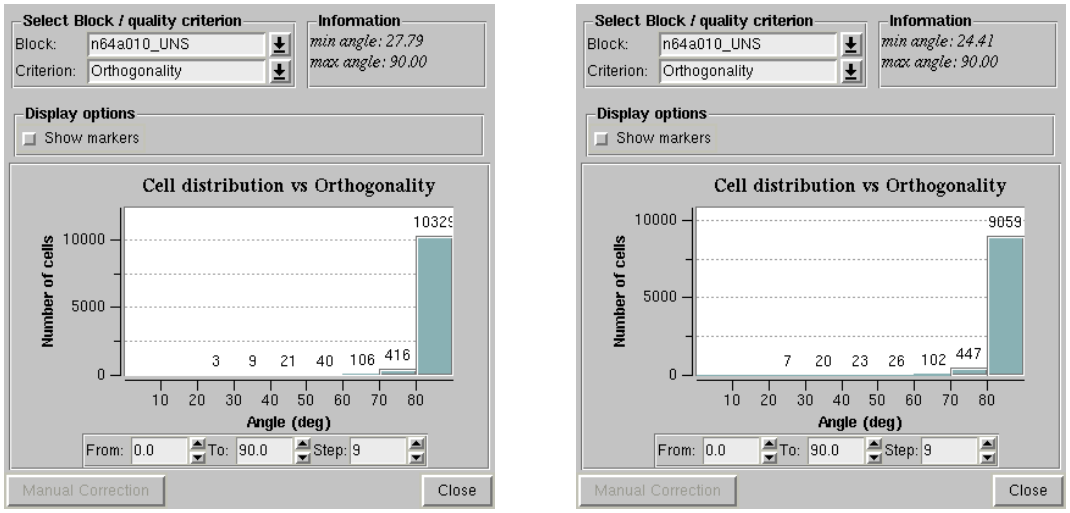


Figure 5.26: Viscous mesh around NACA64A010 airfoil.



(a) New approach. (b) Older approach.

Figure 5.27: Mach number distributions over meshes generated by the new and the older approaches.



(a) New approach. (b) Older approach.

Figure 5.28: Diagrams of orthogonality of meshes generated by the new and the older approaches.

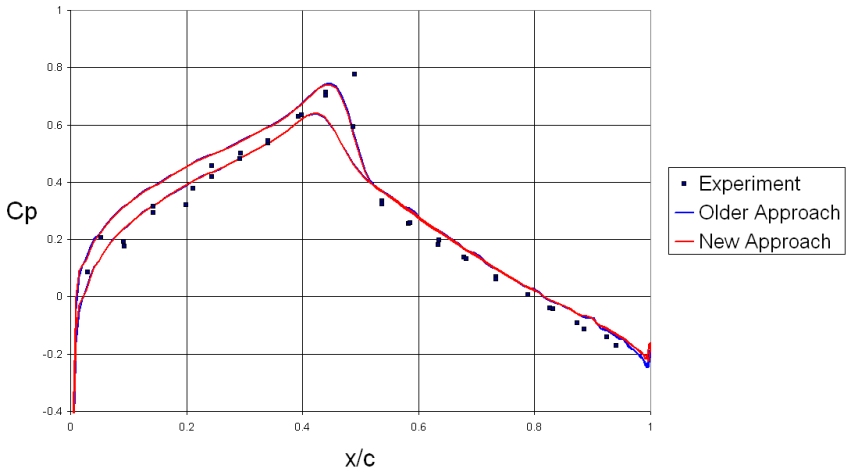


Figure 5.29: Comparison of pressure distributions obtained on both meshes and experimental data.

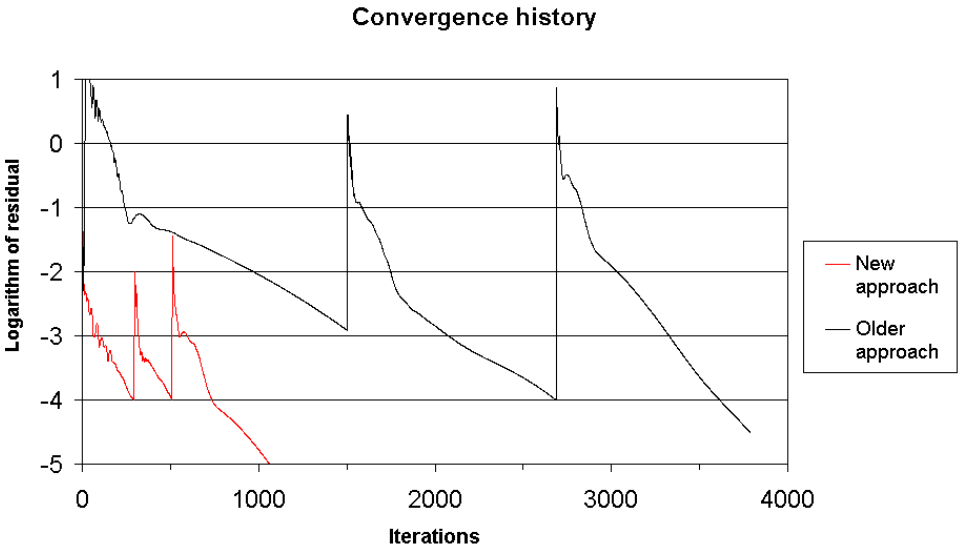


Figure 5.30: Comparison of convergence histories of the computations performed on both meshes.

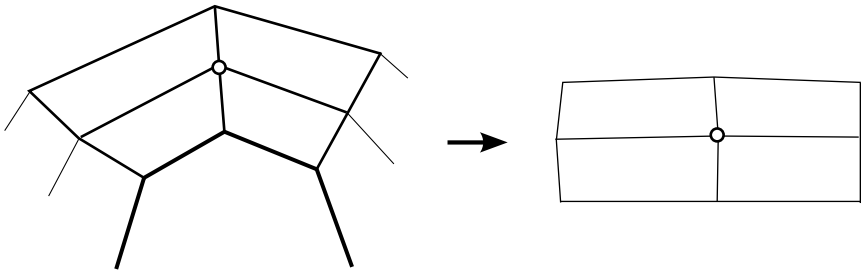


Figure 5.31: Example of a mapping that can be used for surface curvature compensation.

Chapter 6

Mesh Deformation

6.1 Introduction

The task of mesh deformation plays an important part in analysis of physical problems which involve moving bodies and domains with dynamic boundaries. The lack of robust and efficient mesh deformation tools still remains a challenging obstacle for reliable application of numerical analysis methodologies such as CFD or Computational Structural Mechanics (CSM) to multidisciplinary, optimization and non-steady problems. Furthermore, rapid evolution of mesh generation tools in the last decades only increased the demand for development of reliable mesh deformation methods and tools.

One class of challenges which require application of mesh deformation includes various coupled problems where regions with different physics are separated by flexible interface. Typically, these interfaces separate domains with different media properties. For example, in Fluid-Structure Interaction (FSI) problems, CFD analysis is performed on one domain, while structural analysis is applied on the other one. An interface represents a dynamic boundary between domains through which mechanical forces are transferred. A solution to such problem usually consists in finding the new shape of an interface which provides mechanical equilibrium between separated domains. The problem is solved iteratively. Interface shape variation is accounted for in flow analysis in one domain, while pressure distribution over the interface is accounted for in structural analysis in the other domain. In many cases, such coupled procedure shows relatively slow convergence due to explicit nature of the interaction scheme. Hence, mesh deformation tools required by this class of problems must primarily be robust and computationally inexpensive.

Another area of interest in which mesh deformation is involved includes various optimization problems. Usually, the initial shape of geometry of interest must be optimized in such a way that the solution resulting from numerical analysis of this geometry satisfies certain criteria. For instance, in aircraft shape optimization the target surface pressure distribution is specified. A new shape which induces flow structure exhibiting pressure distribution

sufficiently close to target data must be found. In various optimization problems, boundary deformation is usually relatively small and vanishes with convergence. However, the number of optimization loops can be significant. Furthermore, the accuracy of flow analysis performed at each loop should not be affected by quality degradation of the respective mesh. Therefore, this class of problems requires fast mesh deformation tools capable of preserving high mesh quality.

Obviously, the variety of potential applications of mesh deformation is far beyond the two classes of problems described above. For instance, non-steady computations often require deforming meshes. Unlike in coupled or optimization problems, deformation of almost any kind may be required by the specifics of a problem.

This Chapter presents the mesh deformation methodology developed within the generator described in Chapter 3. It is capable of deforming unstructured, fully hexahedral, non-conformal meshes. The methodology employs a generalized interpolation approach for propagating boundary deformation into a volume mesh. Accounting for both translational and rotational components of arbitrary deformation ensures high quality of deformed meshes. This approach is developed based on ideas from the works by Martineau and Georgala [85] and Samareh [104].

6.2 Problem statement

The problem of deformation of an unstructured hexahedral mesh can be formulated as follows:

Problem. *There exists an unstructured hexahedral non-conformal mesh with a fixed boundary. Let $\{V_I\}$ be the set of internal non-hanging nodes of the mesh, $\{V_B\}$ - the set of non-hanging boundary nodes. Initial deformation is specified at nodes of $\{V_B\}$. It is expressed as translation vectors assigned to each node. A new set of positions for nodes of $\{V_I\}$ must be found, such that no cells in the mesh are concave and the overall mesh quality is maintained at a level as high as possible, while topology of the mesh is preserved.*

In other words, nodes of a volume mesh should follow pre-specified boundary motion in such a way that mesh quality is preserved. Mesh topology is assumed to be constant as topology modification is nearly impossible in non-conformal hexahedral framework.

6.3 Methodology

General approach to mesh deformation includes the following several stages. To begin with, initial deformation specified at every point of the surface mesh is converted into a certain data set chosen for the particular method. Following that, the data is interpolated onto the volume mesh nodes by means of chosen interpolation procedure. Finally, interpolated

data is converted back into the respective set of Cartesian displacement vectors. The latter are used for final relocation of mesh points.

In the method developed herein, the initial deformation is divided into two components: translational and rotational. Translation is represented in terms of Cartesian displacement vectors, while rotation is described in terms of quaternions (Section 2.4). For interpolation purposes, a procedure similar to that described in [85] is employed. In this procedure, the actual interpolation is performed on the dual of a volume mesh. Optionally, if quality of the deformed mesh is not satisfactory, the combined mesh quality improvement procedure described in Chapter 4 can be applied. Algorithmic details of the presented approach are described in the following sections.

6.3.1 Representation of deformation data

In the developed method, an alternative representation of deformation as opposed to the traditional description in terms of local displacements is used. The latter may not be successful in situations when rigid properties of deforming geometry must be preserved or in the worst case modified as little as possible. Ideally, shapes of mesh cells must primarily be preserved as much as possible during deformation, while their position and orientation can be modified.

Initial deformation is provided in terms of displacement vectors assigned to each boundary node. These displacements are converted into combinations of translation and rotation, which are assigned to all faces of a surface mesh. Translation is still described in terms of vectors, while rotation is expressed in terms of quaternions. A brief introduction to basic properties of quaternions and respective operations is given below.

Quaternions [104] can be described as generalized complex numbers composed of one real and three imaginary terms:

$$Q = q_0 + q_1 \cdot i + q_2 \cdot j + q_3 \cdot k \tag{6.1}$$

where i, j, k can be considered as analogues of the imaginary unit i. The following relations exist:

$$\begin{gathered} ii = jj = kk = -1 \\ ij = -ji = k \\ jk = -kj = i \\ ki = -ik = j \end{gathered}$$

The three imaginary components represent a vector along the axis of rotation, while the real component represents the angle of rotation. The following properties of quaternions will be useful in the subsequent sections:

- Conjugate of a quaternion, $Q^{\star} = q_0 - q_1 \cdot i - q_2 \cdot j - q_3 \cdot k$
- Magnitude of a quaternion, $\|Q\| = \sqrt{Q \cdot Q^{\star}} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}$
- Unit quaternion, $\|Q\| = 1$
- Associative property, $(Q_1 \cdot Q_2) \cdot Q_3 = Q_1 \cdot (Q_2 \cdot Q_3)$
- Non-commutative property, $Q_1 \cdot Q_2 \neq Q_2 \cdot Q_1$
- Inverse of a quaternion, $Q^{-1} = Q^{\star} / \|Q\|^2$

A quaternion can be considered as combination of a scalar with a Cartesian vector:

$$Q = (r, \mathbf{a}),\ r = q_0,\ \mathbf{a} = (q_1, q_2, q_3) \tag{6.2}$$

A quaternion Q describing rotation of an arbitrary point around the axis along the unit vector $\hat{\mathbf{a}}$ by angle α can be written as:

$$Q = (r, \mathbf{a}),\ r = \cos\frac{\alpha}{2},\ \mathbf{a} = \hat{\mathbf{a}} \cdot \sin\frac{\alpha}{2} \tag{6.3}$$

After such rotation, in order to find the new position of point $\mathbf{p}$ using the quaternion algebra, the point is converted into the respective quaternion $P = (0, \mathbf{p})$. The quaternion P_r corresponding to rotated position $\mathbf{p}_r$ is computed as:

$$P_r = Q \cdot P \cdot Q^{-1} \tag{6.4}$$

As Q represents a unit quaternion (Relation 6.3), a conjugate can be used instead of an inverse:

$$P_r = Q \cdot P \cdot Q^{\star} \tag{6.5}$$

A series of successive rotations can be reduced to a single quaternion using the following relation:

$$Q_1^{-1} \cdot Q_2^{-1} = (Q_2 \cdot Q_1)^{-1} \; or \; Q_1^{\star} \cdot Q_2^{\star} = (Q_2 \cdot Q_1)^{\star} \tag{6.6}$$

Therefore:

$$Q_2 \cdot (Q_1 \cdot P \cdot Q_1^{\star}) \cdot Q_2^{\star} = (Q_2 \cdot Q_1) \cdot P \cdot (Q_1^{\star} \cdot Q_2^{\star}) = (Q_2 \cdot Q_1) \cdot P \cdot (Q_2 \cdot Q_1)^{\star} \tag{6.7}$$

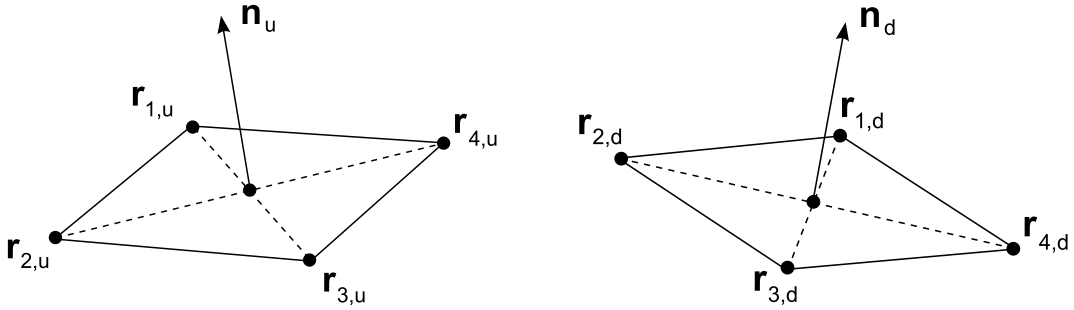
For more details on quaternions, works by Altmann [3], Shoemake [111] and Philips *et al* [98] should be addressed.

Motion of an arbitrary surface mesh face f is defined by displacements specified in four face nodes. The motion can be represented as a combination $(\mathbf{t}_f, Q_f)$ of translation of the face center and rotation of the face by a certain angle around the axis passing through the face center. The translational contribution is denoted by the vector $\mathbf{t}_f$ and is computed as the average of displacements of the face vertices:

$$\mathbf{t}_f = \frac{1}{N_{v,f}} \cdot \sum \mathbf{t}_{v,f} \tag{6.8}$$

Here, $N_{v,f}$ is the number of vertices of face f and $\mathbf{t}_{v,f}$ are the displacements specified in vertices of this face. The rotational contribution for face f is represented by the quaternion Q_f and is constructed using the following procedure. Figure 6.1 depicts the undeformed and deformed positions of face f. As only relative displacements of face vertices affect the face rotation, the first step is to match the centers of both the deformed and undeformed positions (Figure 6.2a). The quaternion describing a complete rotation which face f undergoes during the deformation is constructed as a combination of two rotations:

$$Q_f = Q_1 \cdot Q_2 \tag{6.9}$$



(a) Undeformed face position.

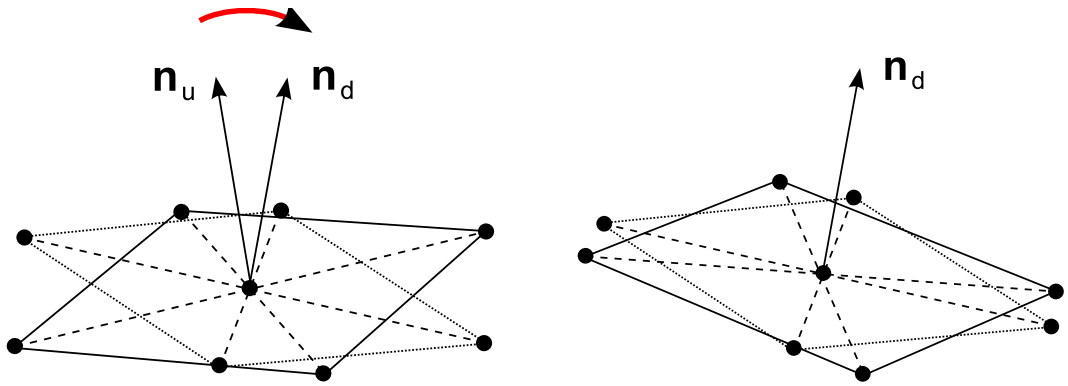
(b) Deformed face position.

Figure 6.1: Undeformed and deformed positions of face f.

Quaternion $Q_1 = (\cos\frac{\alpha_1}{2}, \hat{\mathbf{a}}_1 \cdot \sin\frac{\alpha_1}{2})$ rotates face f from its undeformed position to an intermediate position, in which the face normal is aligned with that of the deformed position (Figure 6.2b). The axis and angle of this rotation are found as:

$$\hat{\mathbf{a}}_1 = \frac{\mathbf{n}_u \times \mathbf{n}_d}{\|\mathbf{n}_u \times \mathbf{n}_d\|},\ \alpha = \arccos\frac{\mathbf{n}_u \cdot \mathbf{n}_d}{\|\mathbf{n}_u\| \cdot \|\mathbf{n}_d\|}, \tag{6.10}$$

where $\mathbf{n}_u, \mathbf{n}_d$ are the face normals corresponding to the undeformed and deformed positions respectively.



(a) Translation to common origin.

(b) Alignement of face normals

Figure 6.2: Alignment of normals of undeformed and deformed positions of face f.

Quaternion Q_2, in turn, rotates the face from the intermediate position with aligned normals to the final deformed position. The axis of this rotation is $\mathbf{n}_d$. In order to find the respective angle, both the intermediate and undeformed positions of face f are projected onto a plane normal to $\mathbf{n}_d$. The angle of rotation is computed as the average of angles by which the respective projections of the face vertices are rotated around $\mathbf{n}_d$ (Figure 6.3).

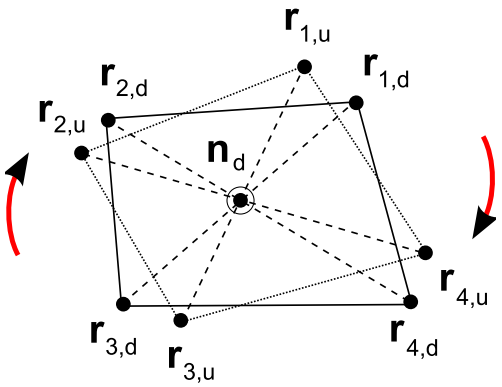


Figure 6.3: Alignment of face vertices of intermediate and deformed positions of face f.

After Q_f is constructed, the translational component is updated as follows:

$$\mathbf{t}_f = \mathbf{t}_f + \mathbf{r}_{f,u} - Q_f \cdot (0, \mathbf{r}_{f,u}) \cdot Q_f^{\star} \tag{6.11}$$

This correction compensates for translation that is induced by rotation of an undeformed face by a respective quaternion.

6.3.2 Dual mesh generation

Once initial deformation is converted into translations and rotations of surface mesh faces, an interpolation procedure is necessary to propagate deformation into the volume mesh. The interpolation procedure described in Section 6.3.5 is based on weighted Laplacian smoothing approach. In order to increase efficiency of this procedure, interpolation is performed on the dual of the original mesh.

The dual of a 3-D mesh is generated as follows. To begin with, vertices of the dual mesh are generated at cell centers of the original mesh. Following that, dual mesh edges are generated. Two vertices are connected with a new edge provided that respective cells of the original mesh share a face (Figure 6.4). A new dual mesh edge is generated for every original mesh face.

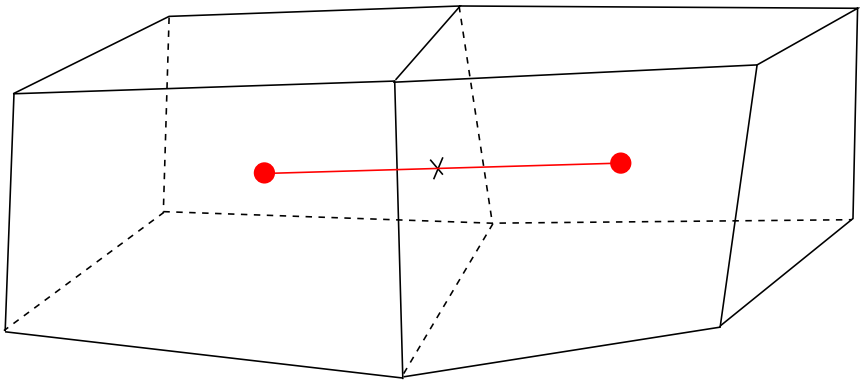


Figure 6.4: Creation of dual mesh edge.

In the next step, faces of the dual mesh are created. A closed set of dual mesh edges forms a new face provided that respective faces of the original mesh share an edge (Figure 6.5). A new dual mesh face is generated for every original mesh edge.

Finally, a set of dual mesh faces constitutes a cell if respective edges of the original mesh are connected to one vertex (Figure 6.6). A new dual mesh cell is generated for every original mesh vertex. A 2-D illustration is used for clarity. In the 3-D, each edge emanating from a vertex of the original mesh corresponds to a face of the dual mesh cell associated with this vertex.

Example of a 2-D dual mesh is depicted in Figure 6.7. The key advantage of using a dual mesh is the fact that it contains no hanging nodes. This positively affects the Laplacian smoothing procedure as hanging nodes are known to decrease the efficiency of the method.

6.3.3 Boundary conditions

Boundary translations and rotations computed as described in Section 6.3.1 are associated with surface mesh faces. A dual mesh vertex is generated for each cell of the original volume

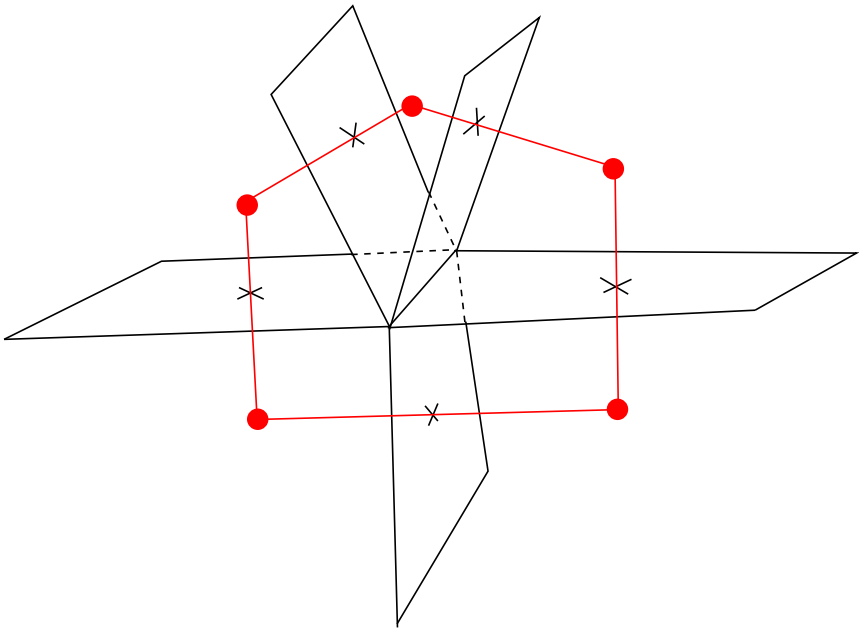


Figure 6.5: Creation of dual mesh face.

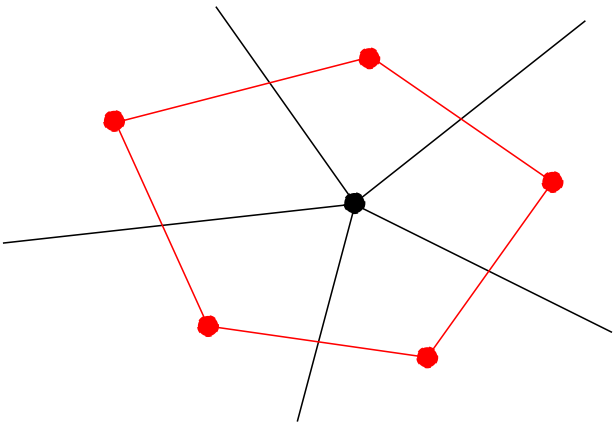


Figure 6.6: Creation of dual mesh cell.

mesh. Boundary vertices of the dual mesh correspond to original mesh cells adjacent to surface faces. Therefore, each boundary vertex of the dual mesh can be associated with a respective face of the original mesh. The quaternion and the translation vector computed for a surface face of the original mesh are assigned to the respective boundary vertex of the dual mesh.

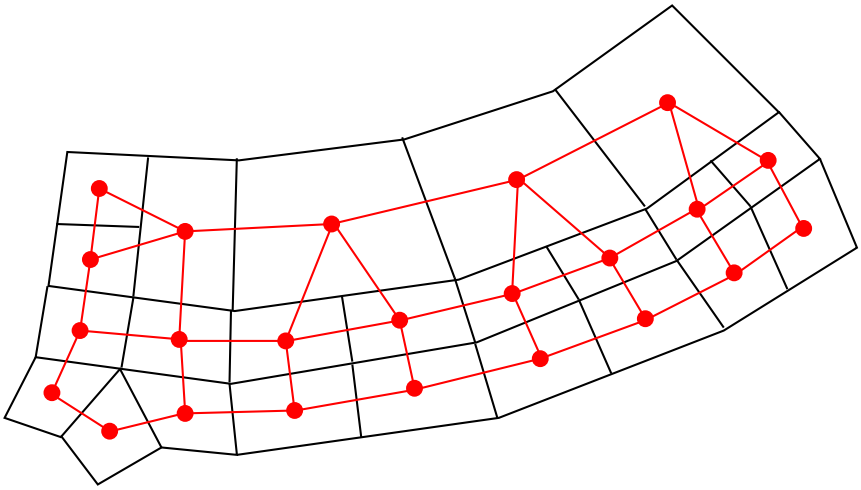


Figure 6.7: Example of a dual mesh.

6.3.4 Initial guess

In order to initiate the interpolation process, an initial solution must be generated in the volume mesh. Generation of the initial guess is of critical importance for efficiency of the further iterative smoothing process. The closer the initially generated solution is to the exact one, the fewer iterations are required for the smoothing to result in a solution of certain accuracy. In other words, the initial guess does not influence the final solution but strongly affects the convergence speed of the iterative procedure. Usually, simple algebraic interpolation methods are employed.

In the present approach, a procedure called *Rigid Body Initialization* (RBI) proposed by Leatham and Chappell [77] is used for generation of the initial solution. For each mesh vertex where a solution must be found (i.e., each internal vertex of the dual mesh in the method described herein), two closest boundary surfaces (i.e., topological faces of the domain) are identified as shown in Figure 6.8.

In this Figure, surfaces a and d are the closest to vertex P. For each of these surfaces, the vertex closest to P is found. In this example, these are vertices A and D, respectively. These vertices as well as the two respective distances are used to generate an initial solution for vertex P. The latter is computed by weighted averaging deformations in vertices A and D, with the weights being equal to inverses of the two computed distances. The translational component is obtained as follows:

$$\mathbf{t}_P = w_A \cdot \mathbf{t}_A + w_D \cdot \mathbf{t}_D,\ w_A = \frac{s_d}{s_a + s_d},\ w_D = \frac{s_a}{s_a + s_d} \tag{6.12}$$

For the rotational component, the so-called spherical interpolation is employed (Figure 6.9). For quaternions, spherical interpolation is performed in 4-D as follows:

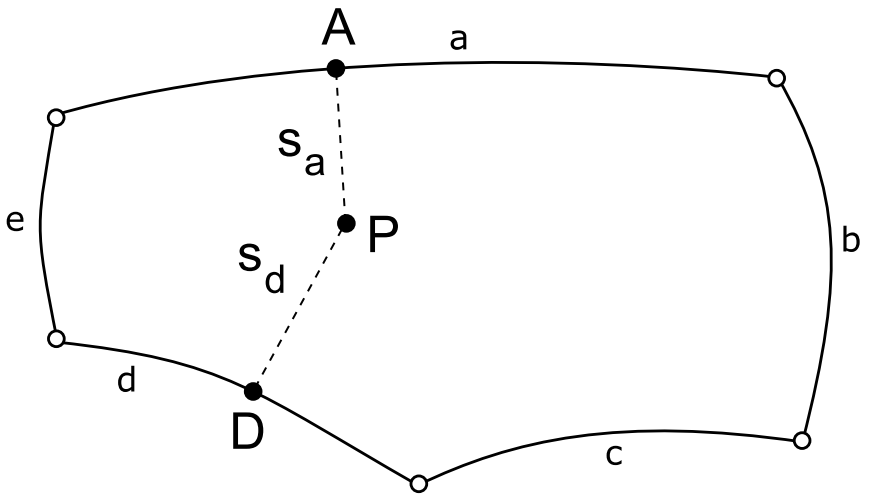


Figure 6.8: Rigid Body Initialization.

$$Q_P = si(Q_A, Q_D, w_A, w_D) = w_A^{(R)} \cdot Q_A + w_D^{(R)} \cdot Q_D \tag{6.13}$$

where $w_A^{(R)}$ and $w_D^{(R)}$ are computed as:

$$w_A^{(R)} = \frac{\sin(\omega \cdot w_A)}{\sin \omega},\ w_D^{(R)} = \frac{\sin(\omega \cdot w_D)}{\sin \omega} \tag{6.14}$$

Here, ω denotes the angle between the two quaternions in 4-D and is computed as $\omega = \arccos(Q_A \cdot Q_D)$. It is assumed that Q_A and Q_D represent unit quaternions.

6.3.5 Interpolation

Weighted Laplacian approach is applied to smooth out the deformation field on the dual mesh. For each dual mesh node N, the new value is computed as the weighted average of values of its neighboring nodes (i.e., those connected to node N by edges). The weights are computed in exactly the same manner as described in Section 6.3.4. The new translational component is computed for node N as the weighted average of respective values in neighboring nodes:

$$\mathbf{t}_N^{(NEW)} = \frac{\sum w_i^{(T)} \cdot \mathbf{t}_i}{\sum w_i^{(T)}},\ w_i^{(T)} = \frac{1}{l_{iN}} \tag{6.15}$$

Here $i = 1, ..., m$ refers to nodes neighboring node N. The new rotational component $Q_N^{(NEW)}$ is computed using formulae (6.13-6.14) for spherical interpolation. As spherical interpolation can be applied only to a pair of quaternions, contributions from all neighboring vertices are added successively:

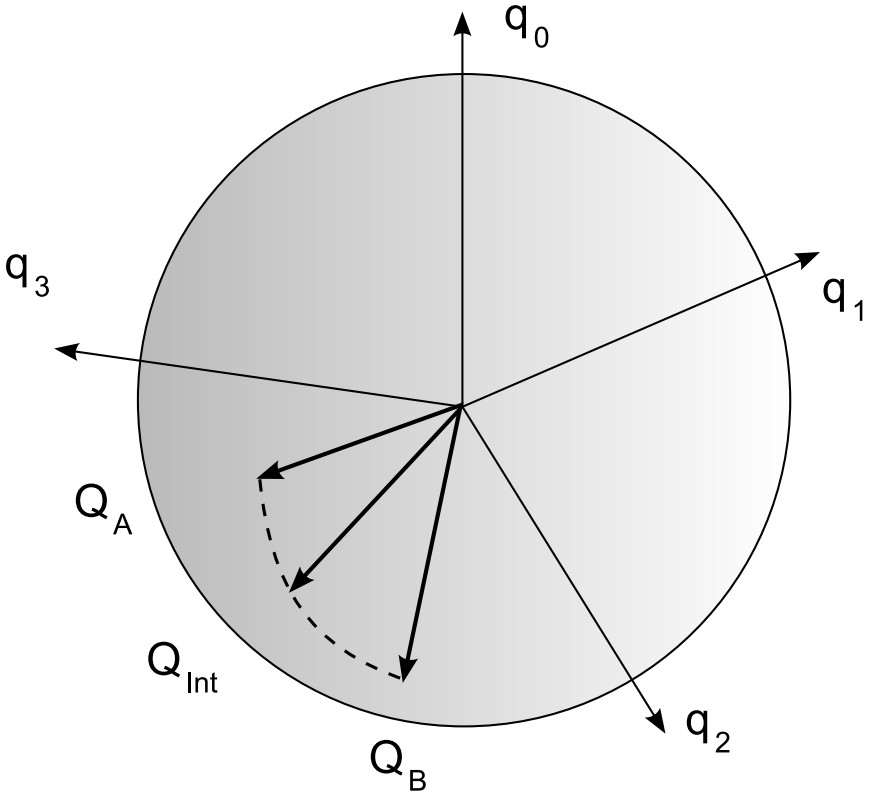


Figure 6.9: Spherical interpolation.

$$Q_N^{(NEW)} = si(Q_1, ..., Q_m, w_1, ..., w_m) = si(Q_1, si(Q_2, si(...), w_2, w_{3...m}), w_1, w_{2...m}) \quad (6.16)$$

Finally, the new quantities are relaxed by the factor r in order to ensure stability of the smoothing procedure:

$$\mathbf{t}_N^{(UPDATE)} = w_A(\frac{s_d}{s_a + s_d}) \cdot \mathbf{t}_N^{(OLD)} + w_D(\frac{s_a}{s_a + s_d}) \cdot \mathbf{t}_N^{(NEW)} \quad (6.17)$$

$$Q_N^{(UPDATE)} = si(Q_N^{(OLD)}, Q_N^{(NEW)}, r, 1 - r) \quad (6.18)$$

The quantities $\mathbf{t}_N^{(UPDATE)}$and $Q_N^{(UPDATE)}$ are used to update positions of mesh nodes in current iteration. The process is considered converged when the ratio of maximal nodal displacement obtained in the latest and in the first iterations drops below certain threshold. The value of 10^{-3} is used in practice.

6.3.6 Displacement of mesh vertices

After the smoothing process converges, interpolated data is transferred from the dual mesh onto the original one. Each dual mesh vertex corresponds to a cell of the original mesh. Thus, initially the data is associated with cells of the original mesh. In order to interpolate the data from cells to vertices, linear interpolation is used for translational components once again. Spherical interpolation is used for rotational terms.

Finally, both the translation vector and the quaternion $(\mathbf{t}_i, Q_i)$ are available for an arbitrary node i of the original mesh. For undeformed nodal position $\mathbf{r}_u$, the deformed position $\mathbf{r}_d$ is defined in two steps. First, the vertex is rotated around the origin using the respective quaternion, resulting in rotated position $\mathbf{r}_r$:

$$\mathbf{r}_r = Q_i \cdot (0, \mathbf{r}_u) \cdot Q_i^\star \tag{6.19}$$

After which the translational component is introduced:

$$\mathbf{r}_d = \mathbf{r}_r + \mathbf{t}_i \tag{6.20}$$

These new deformed positions are used for relocating vertices of the original undeformed mesh. Should any concave or poor-quality cells appear in the mesh after the deformation, the combined mesh quality improvement procedure described in Chapter 4 is applied.

6.4 Results and discussion

The results of application of the mesh deformation procedure to two 2-D meshes are presented.

The first example is the external domain around an isolated airfoil RAE2822. A general view of the generated mesh is shown in Figure 6.10. Six refinements were performed on the initial twenty by seventeen cell grid. The final isotropic mesh contains 2,051 cells. A close-up view of the mesh is depicted in Figure 6.11a; Figure 6.11b shows the respective orthogonality diagram. The minimum angle in the mesh is equal to fifty three degrees.

The deformation specified on the airfoil boundary represents a solid rotation of the latter by fourty five degrees in counterclockwise direction around its leading edge located at the origin. While mesh nodes in the vicinity of the leading edge are rotated and only slightly translated, those near the trailing edge undergo significant translation in addition to rotation. General views of meshes after deformation with and without quaternions are shown in Figures 6.12 and 6.13, respectively.

In the first mesh, rotational deformation is distributed sufficiently far from the airfoil, while the near-field mesh is rotated almost as a solid object. Figure 6.14a depicts a close-up view

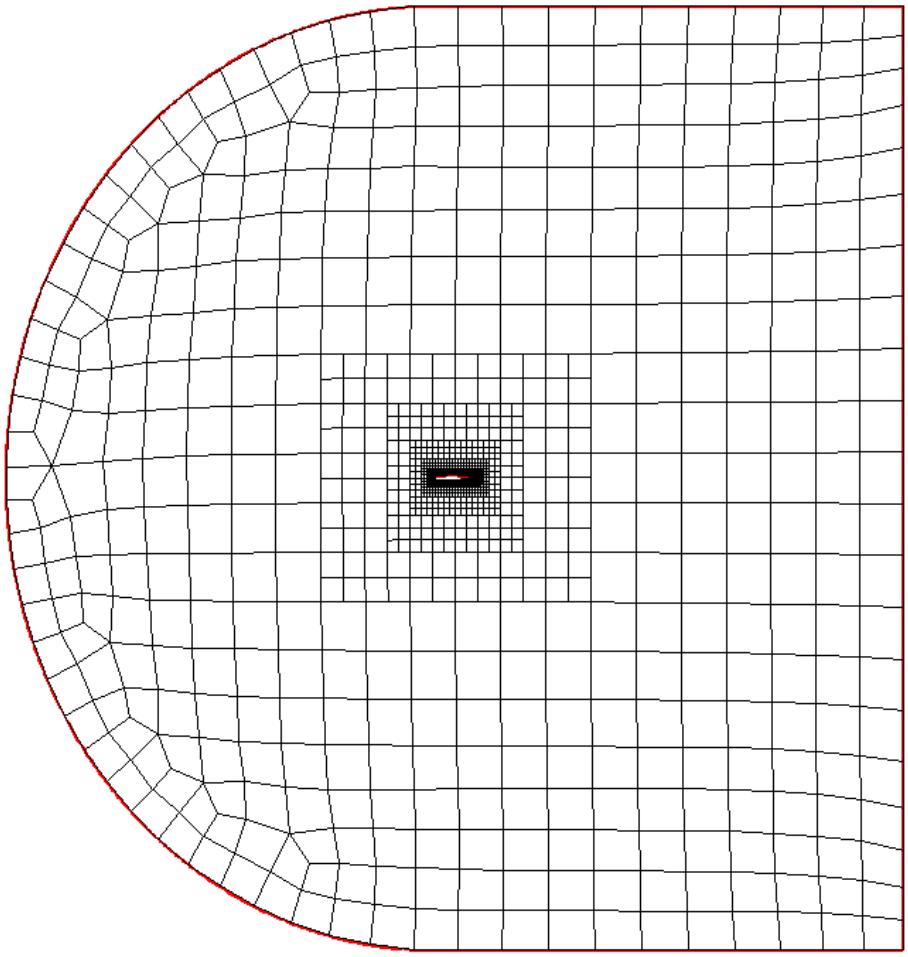
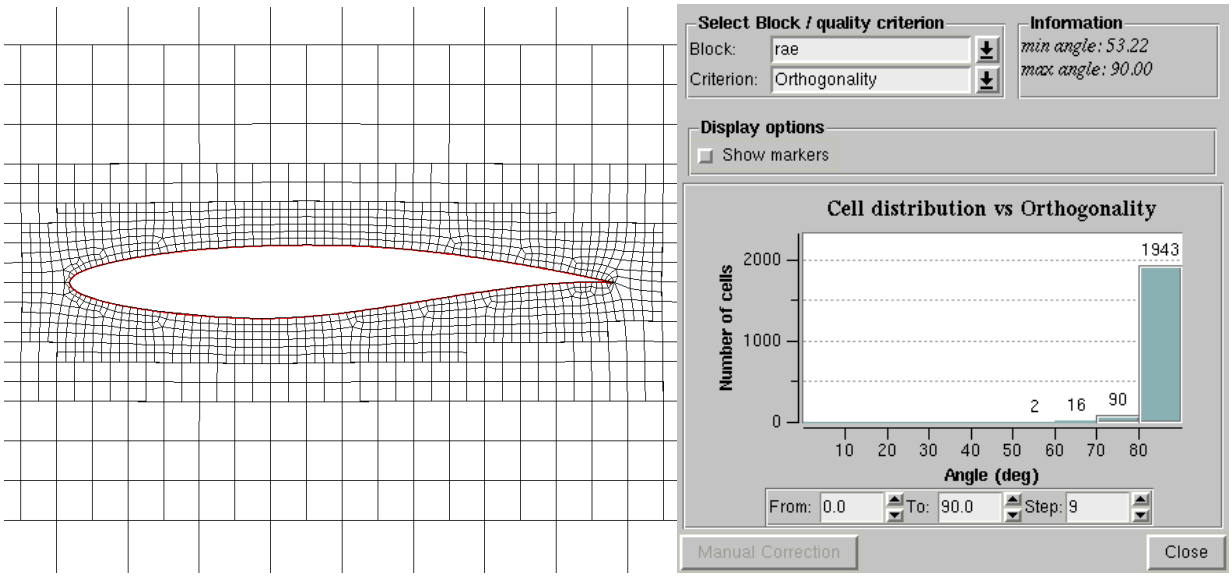


Figure 6.10: General view of initial mesh around RAE airfoil.

of the near-field mesh. On the contrary, the mesh obtained without using quaternions appears to be quite distorted near the airfoil surface, especially around the leading and trailing edges (Figure 6.14b).

Respective orthogonality diagrams confirm this observation (Figure 6.15). In the diagram corresponding to quaternion-free deformation, a peak in the number of cells is present in the range between fourty and fifty degrees. The reason for this is obvious. As rotational components of the deformation were ignored, the mesh cells acquired skew angles, which are approximately equal to the angle of boundary rotation.

Additionally, a ninety degree airfoil rotation was implied as the initial boundary deformation. Application of the method to this test case without using quaternions fails to provide a valid deformed mesh. Conversely, employing quaternions results in a relatively high quality mesh. Figure 6.16 presents a general view of deformed mesh. Rotation is scattered



(a) Close-up view.

(b) Orthogonality diagram.

Figure 6.11: Close-up view of the initial mesh around the RAE airfoil and the orthogonality diagram.

sufficiently far from the airfoil, while the near-field mesh is rotated nearly as a solid body. Generally, the last property is very important as the majority of mesh quality problems typically appear in regions near the geometry of interest. A close-up of the deformed mesh is shown in Figure 6.17a.

Finally, Figure 6.17b shows the respective diagram of orthogonality distribution. It shows good performance of the deformation method: the minimum skewness angle is equal to thirty two degrees, which is sufficiently high; more than seventy five percent of mesh cells remain in the range between eighty and ninety degrees, which corresponds to nearly ideal quadrilaterals.

The second example is another external domain around a multi-element airfoil. The geometry consists of three elements: the main element, a flap, and a slat. Figure 6.18 shows a general view of the mesh generated around the airfoil. The final isotropic mesh consists of about ten thousand cells. A close-up view of the mesh around the slat is depicted in Figure 6.19. This example is intended to demonstrate efficiency of the method on non-single-body geometries.

The specified deformation represents rotation of the slat in clockwise direction around an axis near the leading edge of the main element. Close-up views of the deformed meshes obtained with and without quaternions are shown in Figure 6.20. Obviously, quaternions result in a much better mesh in the vicinity of the slat. Note, that full advantage of the RBI approach for generation of the initial solution (Section 6.3.4) was taken in this example.

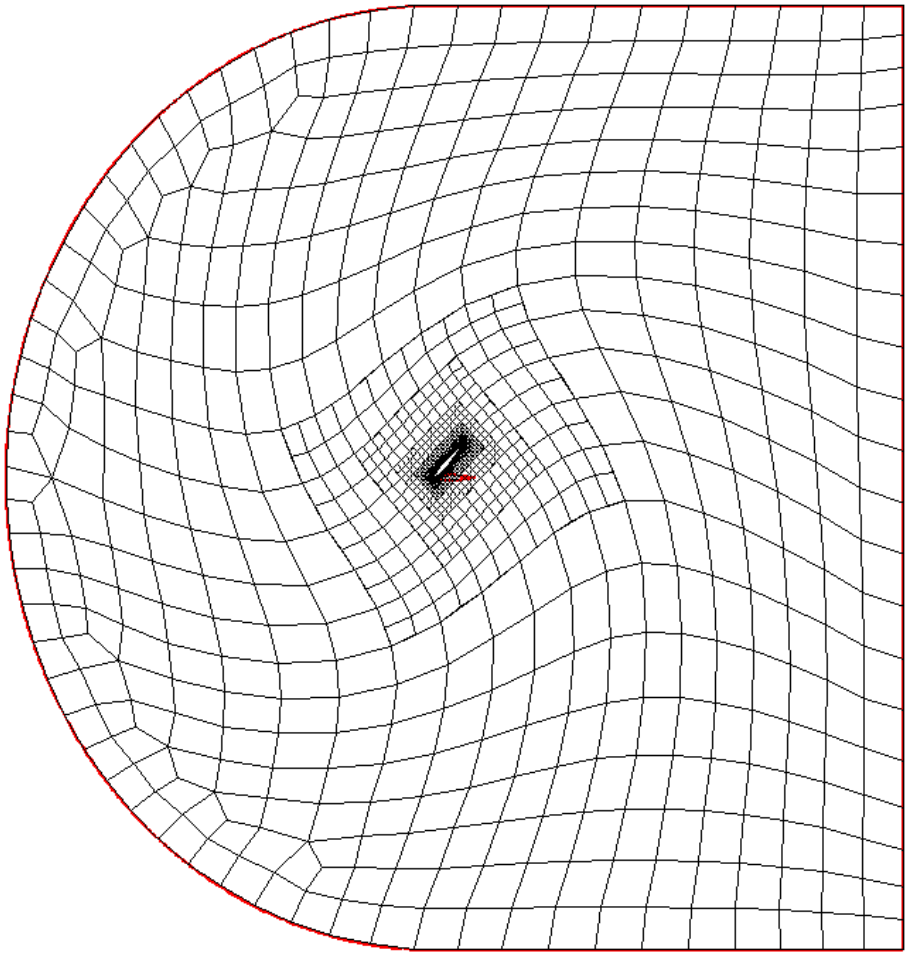


Figure 6.12: General view of the mesh after the RAE airfoil rotation by fourty five degrees using quaternions.

Figure 6.21 presents the respective diagrams of orthogonality distribution over mesh cells. Clearly, the mesh deformed by means of quaternions exhibits better quality. The minimum angle in the mesh is equal to thirty one degrees, which is ten degrees greater than the minimum angle in the mesh deformed using simple displacements. The overall distribution of orthogonality also confirms advantage of the approach equipped with quaternions.

The three-dimensional counterpart of the deformation method presented in the Chapter is currently undergoing final stages of implementation.

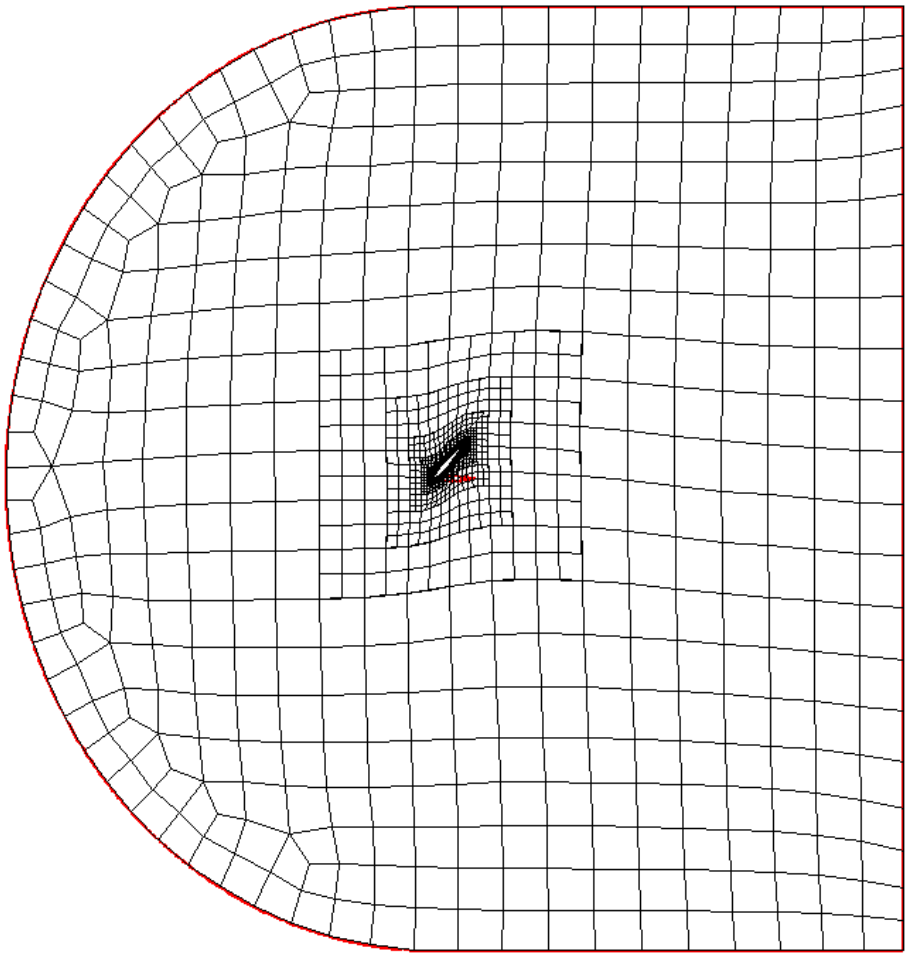
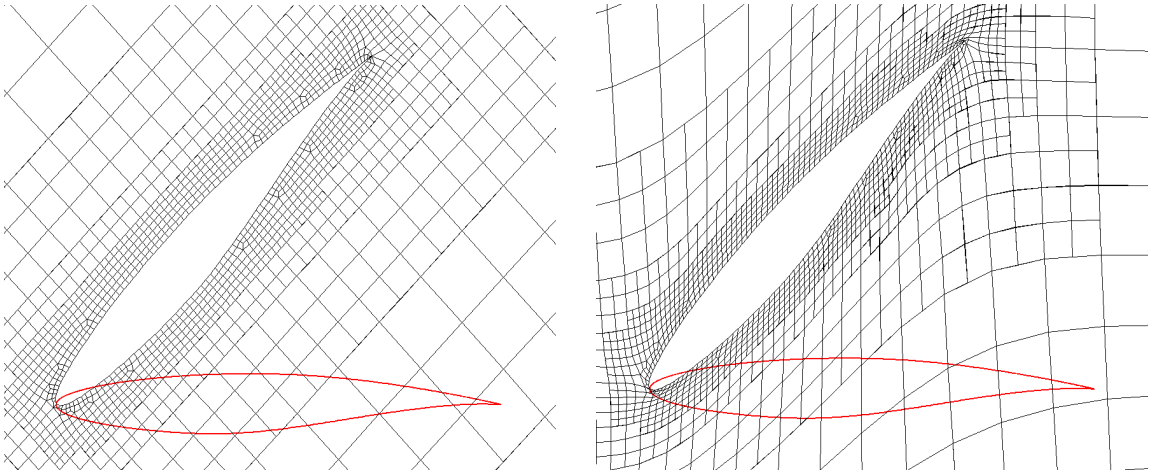


Figure 6.13: General view of the mesh after the RAE airfoil rotation without quaternions.

6.5 Conclusions and recommendations

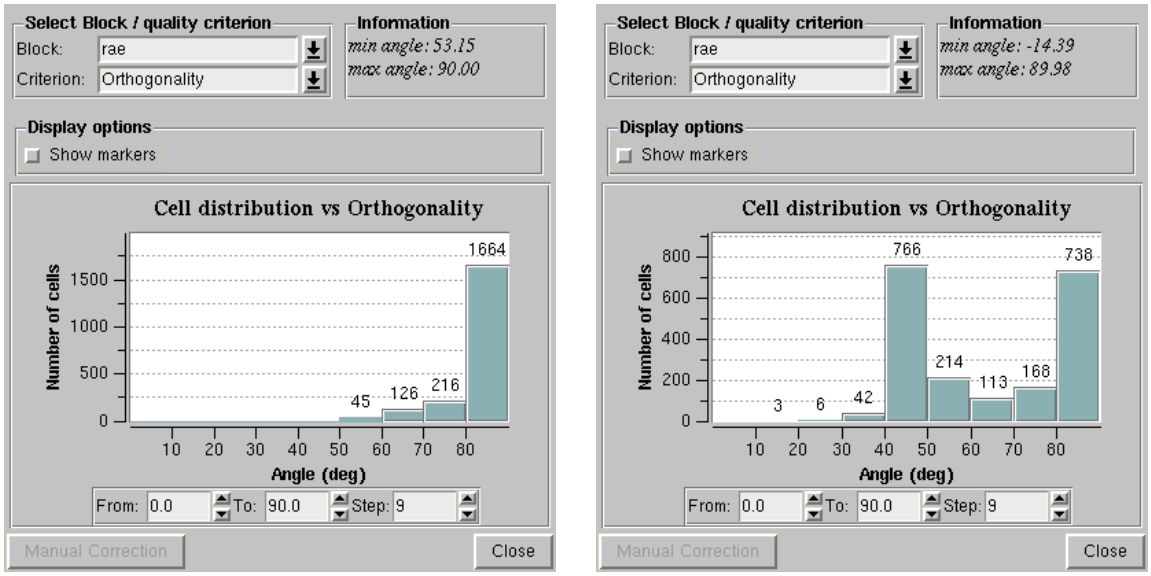
A powerful procedure for deformation of unstructured hexahedral meshes has been developed based on various ideas present in literature. One of key elements of the method is the application of quaternions, a mathematical tool for qualitative representation of rotations. Using quaternions in the mesh deformation algorithm allows obtaining high quality deformed meshes even when rotation by large angles is involved. In particular, a mesh around an airfoil was successfully deformed following rotation by ninety degrees. The final mesh not only appears to be valid but also exhibits sufficiently high quality of its cells.

Another important feature of the method is the use of a dual mesh for interpolation of deformation into a volume mesh. The non-conformal refinement employed in the mesh



(a) Quaternions are applied. (b) No quaternions are applied.

Figure 6.14: Close-up views of the meshes around the rotated RAE airfoil.



(a) Quaternions are applied. (b) No quaternions are applied.

Figure 6.15: Orthogonality diagrams of the meshes around the rotated RAE airfoil.

generation approach results in a certain number of hanging nodes (Figure 4.1 in Section 4.2.1), presence of which decreases efficiency of Laplacian smoothing employed for defor-

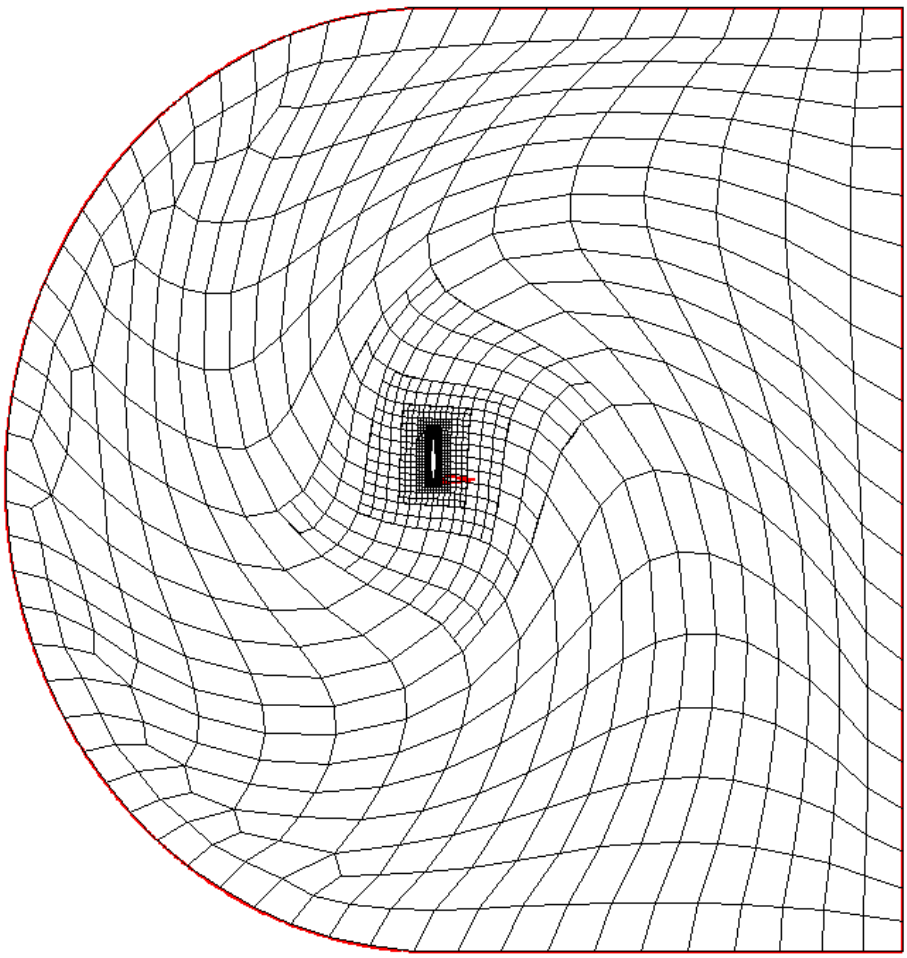
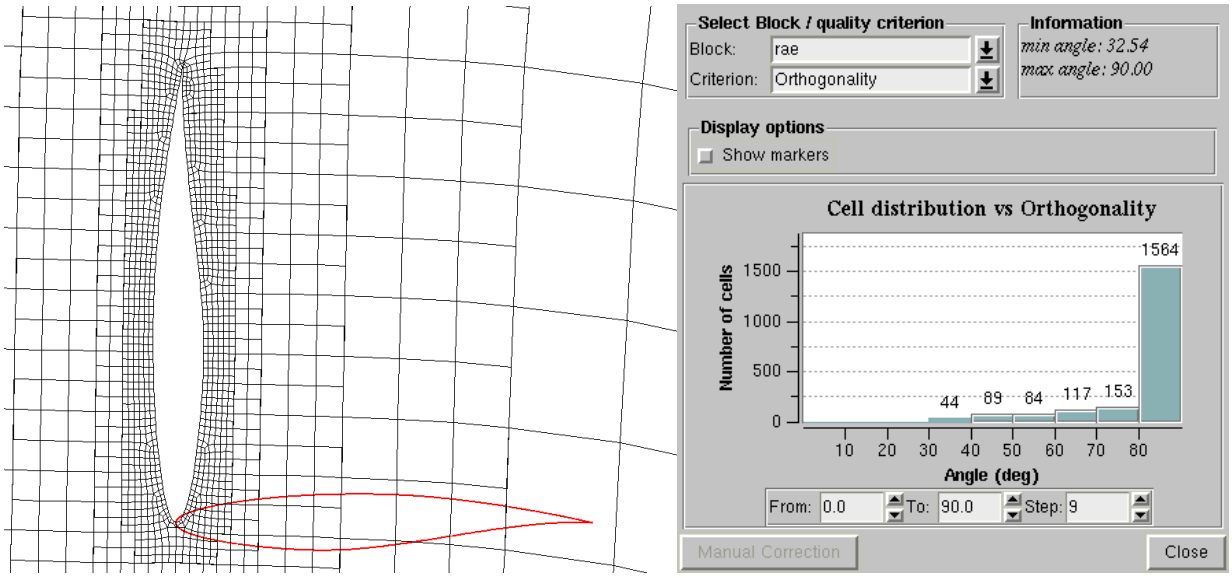


Figure 6.16: General view of the mesh after airfoil rotation by ninety degrees using quaternions.

mation interpolation. Application of a dual mesh enables to avoid this complication. It is constructed in such a way that no hanging nodes participate in the interpolation process. The effect of this improvement can be observed in faster convergence of smoothing iterations, which positively contributes to the overall speed of the deformation procedure.

Finally, using the RBI approach for constructing the initial solutions also results in acceleration of the interpolation procedure for cases in which deformations induced by different parts of input geometry interfere with each other. The RBI method allows for construction of better initial solutions for such cases.

The method has been proved to show exceptional performance in certain cases. Algorithmic structure appears to be highly suitable for the framework of unstructured hexahedral non-conformal meshing. The only aspect in which the method can be improved is the com-



(a) Deformed mesh around the airfoil.

(b) Orthogonality diagram.

Figure 6.17: Close-up view of the mesh around the rotated RAE airfoil and orthogonality diagram.

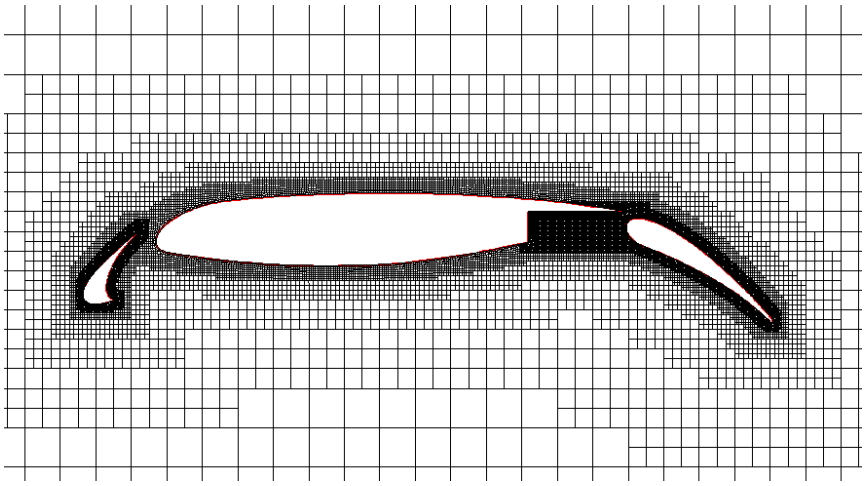


Figure 6.18: General view of the initial mesh around the multi-element airfoil.

putational cost. Currently, generation of initial solutions using the RBI method employs distances to closest boundaries calculated for each internal node. These are computed using direct approach. However, this operation is quite costly and may potentially result in unacceptable time consumption for large meshes. Therefore, it is proposed to employ more powerful tools for generation of distance maps. One possibility is to use some kind of

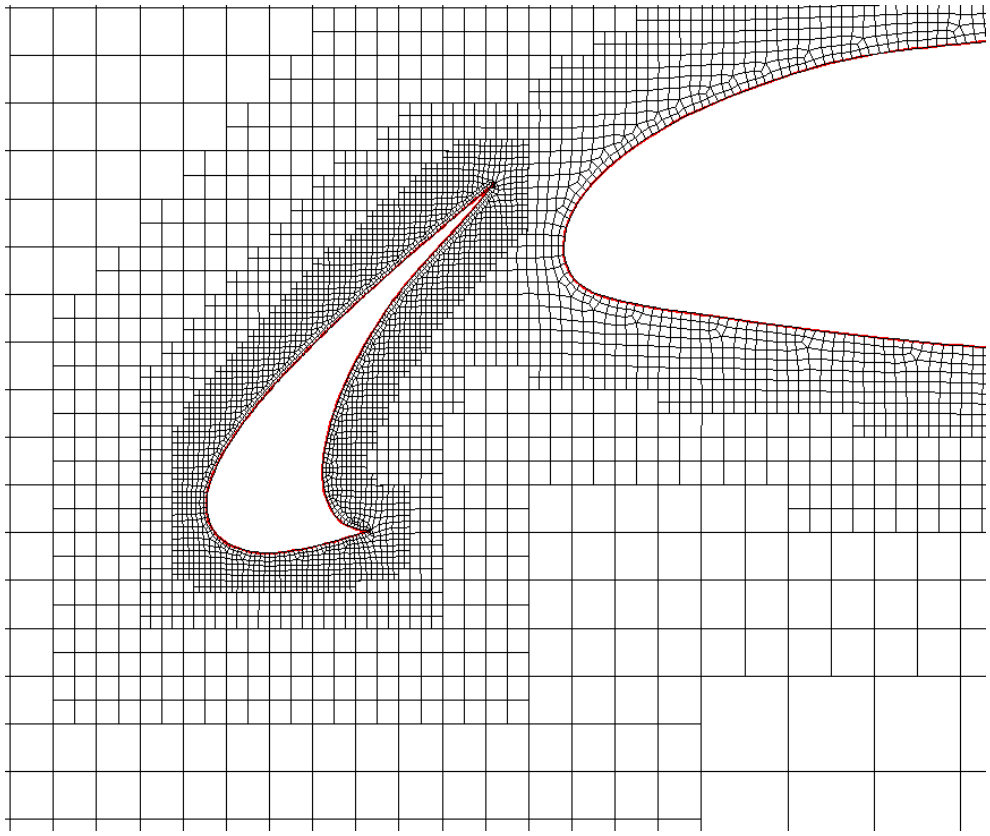
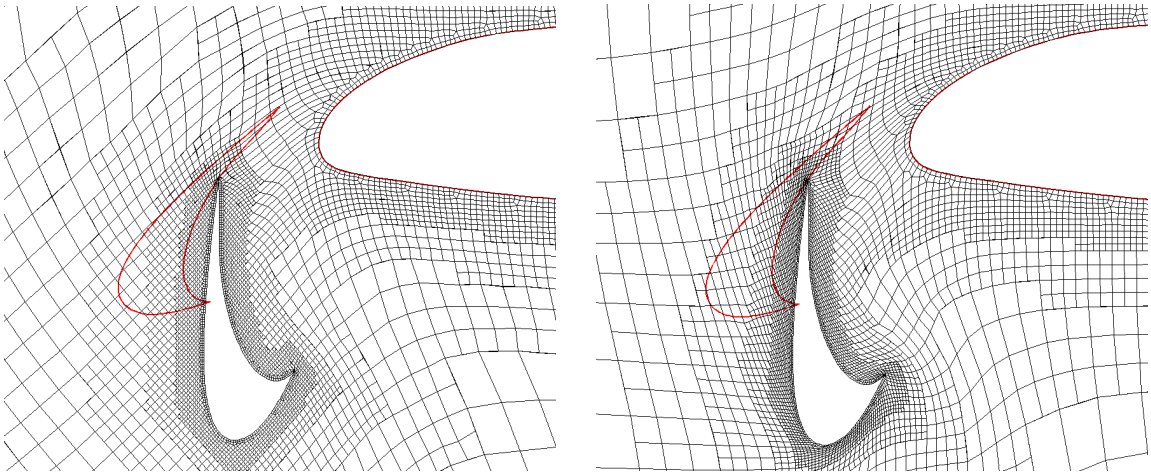


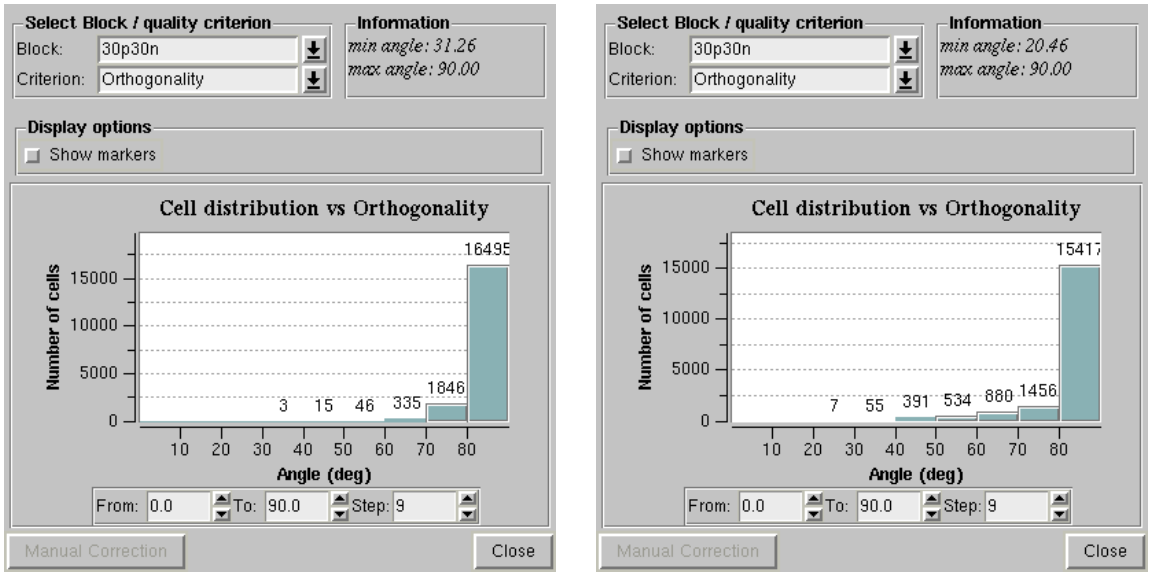
Figure 6.19: Close-up view of the initial mesh around the front element of the multi-element airfoil.

"predictor-corrector" approach. The distance between an internal mesh node and a boundary can be computed initially as a sum of lengths of edges which constitute the shortest path between a surface and a node. At this stage, the respective closest surface point is known for each internal mesh node and the distances can be recalculated precisely quickly.



(a) Quaternions are applied. (b) No quaternions are applied.

Figure 6.20: Close-up views of meshes around the rotated front element of the multi-element airfoil.



(a) Quaternions are applied. (b) No quaternions are applied.

Figure 6.21: The orthogonality diagrams of deformed meshes around the multi-element airfoil.

Chapter 7

Application: prediction of car tire aquaplaning

7.1 Introduction

Aquaplaning is a physical phenomenon that occurs as a result of driving on wet or flooded roads at high speed. It can pose a serious threat to public safety. Aquaplaning is characterized by a sudden reduction in ground traction which leads to the immediate loss of control and stability of the vehicle. This happens due to partial or full detachment of a tire from the road. Evidently, aquaplaning is responsible for a variety of potentially dangerous or even life-threatening incidents occurring in bad weather conditions. Taking this issue into consideration, tire manufacturers are obliged to do their best to gain full understanding of this phenomenon in order to be capable of designing tires demonstrating better stability on wet roads.

The challenge of predicting the effects of aquaplaning on a tire on water-covered pavement has been addressed by several authors since 1960's. In first works devoted to investigation of aquaplaning [21,57] researchers were able to carry out only experimental and mathematical analysis of the phenomenon as no sufficient computational methodology existed at that time. Results obtained in these early investigations could not allow predicting the onset conditions of aquaplaning.

Due to the rapid development of computer technology and computational models, simulation of aquaplaning underwent a significant move in the 1990's . Modeling of rather complex flows as well as simulations of multidisciplinary phenomena such as fluid-structure interaction became feasible at that time. These breakthrough achievements enabled to readdress the issue of aquaplaning as a complex numerical problem consisting of structural analysis of tire material as well as interaction of the tire surface in contact with pavement and the fluid flow in the vicinity of this contact. Overviewed here are the most interesting works discussing various approaches of numerical simulation of aquaplaning.

Aksenov *et al [2]* made an attempt to predict aquaplaning effect using the FlowVision software from Capvidia (www.capvidia.be) in 1996. Deformation of the tire shape was not considered in the described analysis. Therefore, the computational domain remained fixed in time. The domain was meshed via FlowVision CFD code, which generated a Cartesian octree-based non-conformal mesh. Due to intersection of Cartesian cells with boundaries, in particular, with the tire surface, such a technique produced irregular polyhedral cells near the domain boundaries. Authors did not provide much technical information on setup and conditions of the simulation. Results obtained from this investigation included a plot of lifting force acting on a tire versus angle of inclination of latitudinal grooves for both rolling and blocked tires.

Okano and Koishi have used the overlapping mesh approach in order to address the problem of prediction of aquaplaning using MSC.Dytran code [94]. The code combines the Finite-Element and Finite-Volume methods to solve coupled problems of interaction between solid deformation and fluid flow. In this analysis, the tire body is modeled by Lagrangian formulation and the fluid flow is modeled using Eulerian formulation. The previous attempt to use a similar procedure was made by Nakajima *et al* [92], but the aquaplaning onset velocity was not predicted in that work. The challenge was successfully tackled by Okano and Koishi, whose effort on simulation of aquaplaning provided a series of valuable results for the phenomenon. The aquaplaning onset velocity was determined for a slick tire, a longitudinally grooved tire, and a tire with given tread pattern. Obtained values were compared with experimental results and this comparison showed good agreement between experiments and numerical simulations.

In conclusion another approach developed by Grogger and Weiss [52,53] is considered. The authors have done two consecutive investigations of the aquaplaning phenomenon.

First, a computational model of a slick tire on pavement covered with a water film was developed [52]. The model was applied to the problem of simulation of aquaplaning, and computational results were validated with experimental data. The two-phase flow consisting of water and air was modeled by the mass fraction approach, in which an additional variable, the water mass fraction, was introduced. The tire body was meshed with a final elements hexahedral mesh. This mesh was generated for geometry of a tire deformed under static car load. No rotation was considered, therefore, no centrifugal force was present. It corresponds to flow simulation on a braking tire which is locked. The flow domain was represented as a flat box with the tire footprint on top of it. The footprint extends to the bottom of the box to form a contact patch. A multi-block structured mesh of almost twenty thousand cells was generated for this computational domain.

In this simulation the tire was assumed to be non-deforming. It means that any tire deformation that may potentially occur due to interaction between the tire and water film is neglected. This conclusion is in good agreement with the validation results. For low speeds between 30 km/h and 60 km/h, the absolute pressure distribution along pavement matches well with experiments. On the contrary, for higher speeds around 90 km/h the discrepancy is rather large. Ignoring tire deformation during simulation results in an overestimated

value of the predicted onset velocity.

In order to be able to handle dynamic deformation of tire surface and longitudinal grooves along tire surface [53], the approach was extended by the same authors in 1997. This approach is very similar to the one described above. A multi-block structured grid was generated for a grooved domain similar to the one shown in Figure 7.4a. Unlike in the previous investigation, this analysis accounted for dynamic tire deformation. Size of the contact patch between tire and pavement might change during simulation.

Results for the slick tire simulation included contours of the tire surface near the contact patch for three different speeds, namely 30 km/h, 60 km/h, and 90 km/h, and the water film thickness of 8 mm. These contours were compared with corresponding contours obtained for a non-deformable tire to show effect induced by tire deformation. The contours for the grooved tire simulation were obtained for various thicknesses of the water film, namely, 4, 8, and 12 mm, and the speed of 90 km/h. They were compared with respective contours for a non-deformable tire.

A coupled approach that employs an integrated software tool for simulation of the aquaplaning phenomenon is described in the upcoming sections. The approach is based on a weak coupled fluid-structure interaction model. The integrated software modules include the flow solver FINE^{TM}/Hexa [96, 97] for simulating the complex flow around a tire, the structural analysis solver MSC.Marc/MSC.Mentat [89] for simulation of the tire structural behaviour, a coupling module for efficient transfer of essential variables between the solvers. The flow solver operates using unstructured non-conformal hexahedral meshes. For the reason that flow simulation in a domain with variable boundary such as car tire surface is performed, the deformation module described in Section 6 is employed for deforming meshes used during the flow simulation. This work was performed by the software engineering company NUMECA, International, the Department of Fluid Mechanics of Free University of Brussels (Vrije Universiteit Brussel), and the software corporation MSC. The work was supported through the Growth TROPHY project funded by the European Commission under the contract No. G3RD-CT-2001-00510.

Further presentation of the approach is organized as follows. To begin with, a general description of the approach is provided. It includes certain details on application of each of the coupled components. The mesh deformation procedure employed in simulation is described in greater detail. Finally, results of the performed simulations are compared to respective experimental data and the comparison is analysed.

7.2 CFD module

The mathematical model used for description of the two-phase flow around a tire belongs to the class of single-field representations. Both liquid and gaseous phases are considered to be a single medium, properties of which, such as density and viscosity, undergo variations of orders of magnitude over the interface between fluid and gas. This feature of the model

allows for a proper identification of the interface without tracking it directly. The so-called *Volume-Of-Fluid* (VOF) is employed for capturing dynamics of the free surface separating the liquid and gaseous phases. More details on this method can be found in [99].

The utilized flow solver FINE^{TM}/Hexa developed at NUMECA, Int. uses unstructured hexahedral body-fitted meshes. The second order finite volume spatial discretization with central scheme as well as added artificial dissipation are employed. The only exception is the VOF equation, which is discretized using the first order upwind scheme. Unsteady flows are handled by means of a dual-time stepping procedure.

7.3 CSM module

A general purpose non-linear finite element software tool called MSC.Marc is employed for simulation of structural dynamics of a tire. This solver is capable of performing linear and non-linear stress analysis in static or dynamic framework. In order for it to be employed in the aquaplaning simulation procedure, the solver is equipped with additional material models for description of composite materials or rubber. Furthermore, definitions of finite-element models of both treaded and non-treaded tires were introduced.

7.4 Coupling module

The coupling module is designed to perform exchange of information between the two solvers in order to account for tire surface deformation. The MpCCI library (Mesh-based parallel Code Coupling Interface) is used for communications between the solvers [1]. This software was specifically designed for automatic coupling in multi-disciplinary applications.

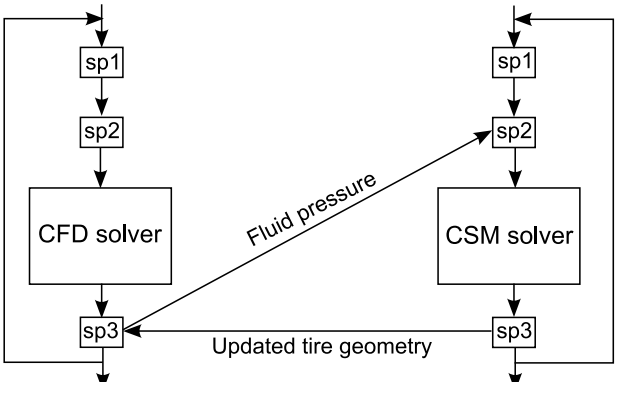


Figure 7.1: Coupled simulation scheme.

The coupling scheme for simulation of aquaplaning is depicted in Figure 7.1. Exchange of information between the solvers is performed by means of synchronization points (sp)

embedded in both solvers. When one of the two codes reaches such a point during execution, it is obliged to wait for the other solver to reach a respective point. After both codes reach a synchronization point, exchange of information is performed and execution of the codes continues.

7.5 Mesh deformation

Within the TROPHY project, two types of tires were used for simulation, namely a longitudinally-grooved non-treaded tire and a longitudinally-grooved treaded tire. The former represents a slick tire with a few longitudinal grooves. The latter is additionally equipped with tiny treads over the entire surface. Examples of the two tire types are shown in Figure 7.2. Mesh deformation approaches employed for both tire types are described below.

(a) Non-treaded grooved tire.

(b) Treaded tire.

Figure 7.2: Examples of a grooved non-treaded and a treaded Goodyear tires.

7.5.1 Non-treaded tire

A non-treaded tire represents a simpler problem in terms of mesh deformation as compared to a treaded one. The general scheme of the non-treaded mesh deformation framework is presented in Figure 7.3. The region of interest, i.e., the computational domain for flow simulation, represents a flat box parallel to the pavement surface. A footprint of the tire penetrates through the box. The area of contact between the tire and pavement defines the shrunken area of the domain (Figure 7.4a).

Shape and size of the area of contact between tire and pavement also known as the contact patch are subject to variation during simulation. Therefore, the initial mesh cannot be

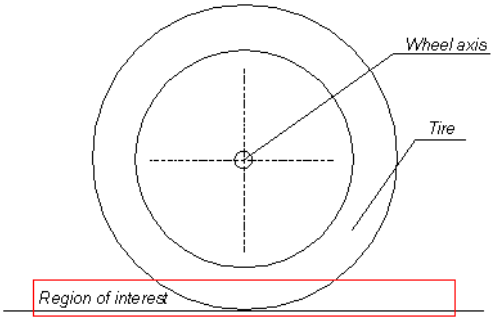
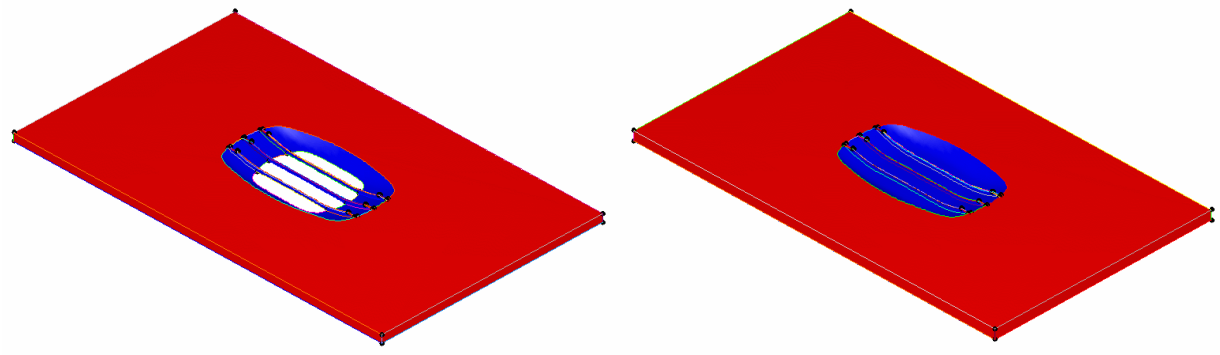


Figure 7.3: A framework for deformation of a non-treaded tire.

generated using the domain shown in Figure 7.4a, as it is limited to a fixed shape of contact patch. To overcome this obstacle, the domain depicted in Figure 7.4b is used for generation of the initial mesh.



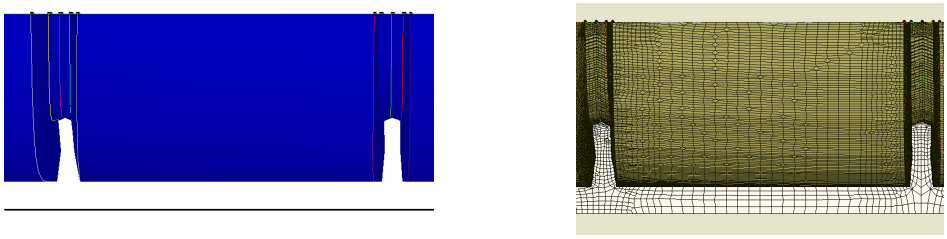
(a) Domain for flow simulation.

(b) Domain for mesh generation.

Figure 7.4: Domains used for flow simulation and mesh generation.

This domain contains a gap between the tire and pavement, which is populated with cells during the mesh generation process. A close-up view of the gap and a mesh cut across the gap are presented in Figure 7.5. After the gap is meshed, the bottom part of the overall mesh undergoes shrinking in upward direction until the gap is closed (Figure 7.6). This simulates an unloaded tire touching the road at a single point. However, the final domain is created using the loaded position of a tire. Loading a tire results in a downward displacement of the tire surface in the vicinity of contact patch. The lowermost points of the tire are in contact with the road. A group of facets of the tire surface which reaches the pavement during the loading process constitutes the resulting contact patch.

Such an approach for generation of the initial mesh allows for modification of the contact patch size directly during simulations. If the tire surface raises during simulation, some



(a) Domain view. (b) Mesh cut.

Figure 7.5: The gap between the tire and the pavement.

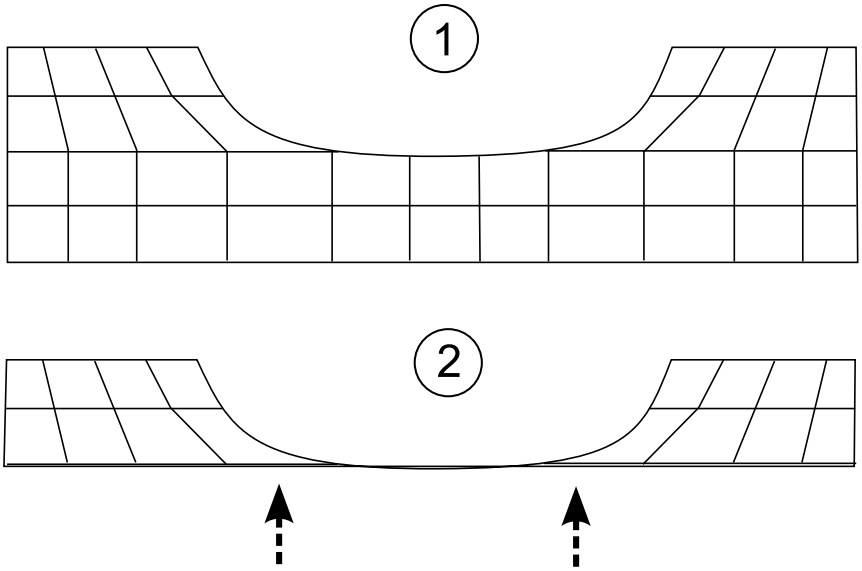
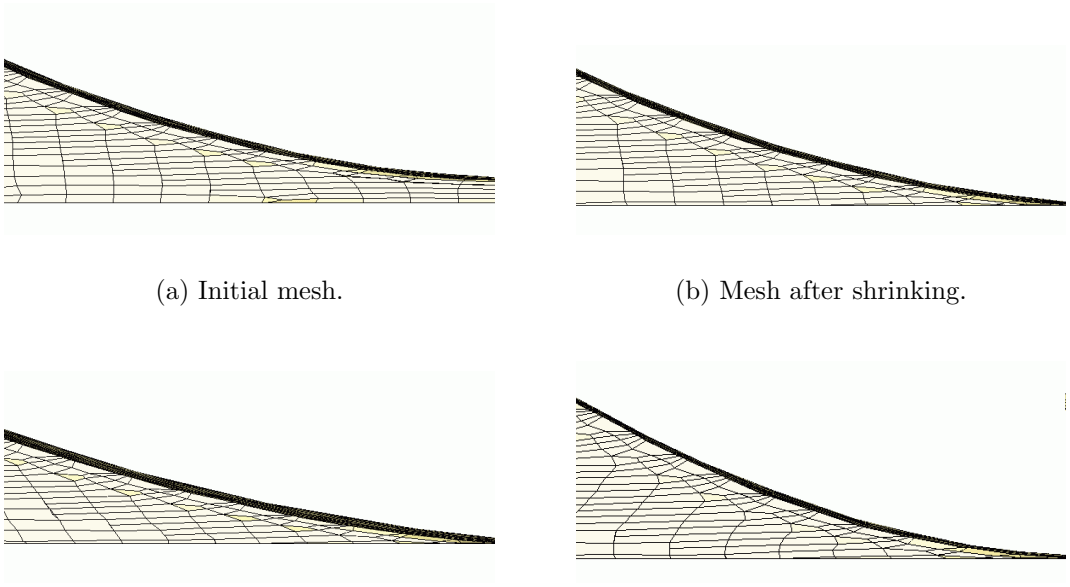


Figure 7.6: Shrinking of the initial mesh.

points of the contact patch are displaced upwards, thus reducing the size of the patch. And, conversely, the contact patch increases in size when the tire moves downwards.

Besides modifications of the contact patch, a deformation procedure is applied to the mesh as described in Chapter 6. The initial deformation is specified only on the surface of the contact patch. Other domain boundaries remain fixed. The mesh undergoes "unshrinking", i.e. the gap and cells belonging to it are recovered, and the deformation procedure is applied. The unshrinking is necessary in order to properly displace points in the gap, as they may potentially be involved in computation. After the mesh points are repositioned according to the boundary deformation, the gap is shrunk again in order to recover the domain existing prior to mesh deformation.

Figure 7.7 depicts examples of a mesh which was generated, shrunk and deformed utilizing the presented approach. Center-plane cuts across meshes for various stages are presented.



(a) Initial mesh.

(b) Mesh after shrinking.

(c) Deformed mesh.

(d) Another deformed mesh.

Figure 7.7: Example of a mesh generated and deformed using the described shrinking procedure.

Figure 7.7a shows the initial mesh with a open gap and Figure 7.7b depicts the same mesh with a closed gap. Finally, Figures 7.7c and 7.7d present the mesh deformed based on two different boundary deformation patterns.

7.5.2 Treaded tire

As compared to a non-treaded tire, treaded geometry additionally includes multiple treads located on the tire surface (Fig. 3). These treads usually have shapes of curved, very narrow, and relatively shallow grooves. Typically, they are not aligned with the principal directions of a tire and are oriented at an inclination angle with respect to longitudinal grooves.

One of the limitations of the non-treaded approach is that the flow motion around a non-treaded tire is modeled as if the tire were resting on the ground and the fluid flew around it. The major grooves of a non-treaded tire are aligned along the direction of motion and, therefore, the tire geometry represents a body of revolution slightly flattened at the contact patch. However, the latter remains in the same position with respect to the ground, which allows us applying the resting tire model for non-treaded tires.

The situation is more complicated for the treaded tire case. Unaligned treads do not allow to consider the tire geometry as a body of revolution any longer. To address this

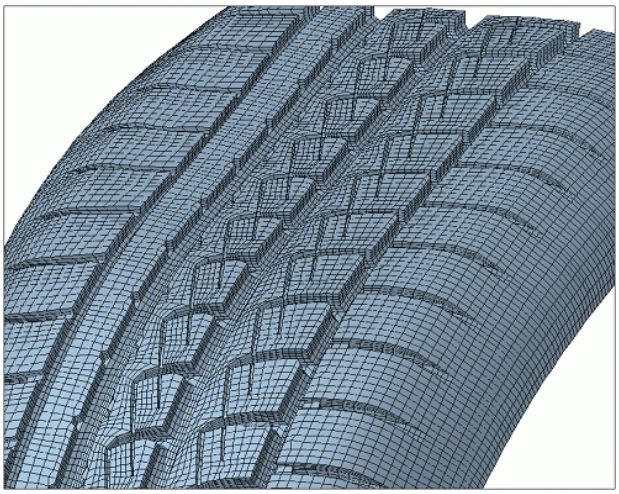


Figure 7.8: Fragment of the surface mesh on a treaded tire.

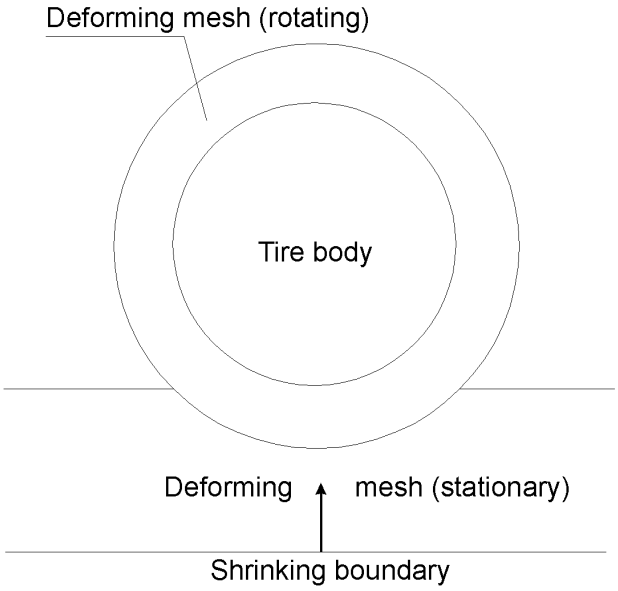


Figure 7.9: Framework for deformation of treaded tire.

problem, a composite approach is applied. It includes rotating and stationary domains and the connection between them. This setup is schematically illustrated in Figure 7.9. The stationary domain is at rest with respect to the ground and the rotating domain revolves along with the tire.

The coupled procedure for simulation of motion of a rolling treaded tire is organized as follows. As in the non-treaded approach, the initial meshes generated for both domains undergo shrinking between the contact patch and road surface (Figure 7.10). In each iteration of the coupled simulation procedure, the tire is rotated by a small angle corresponding to

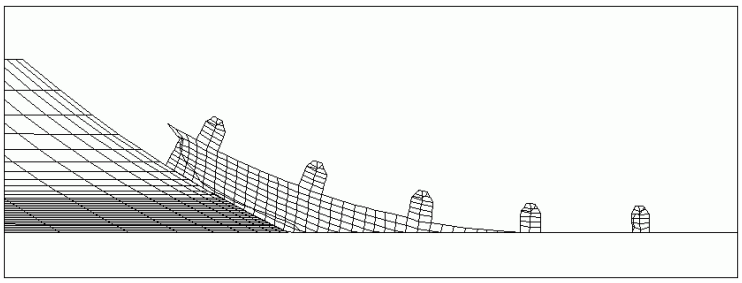
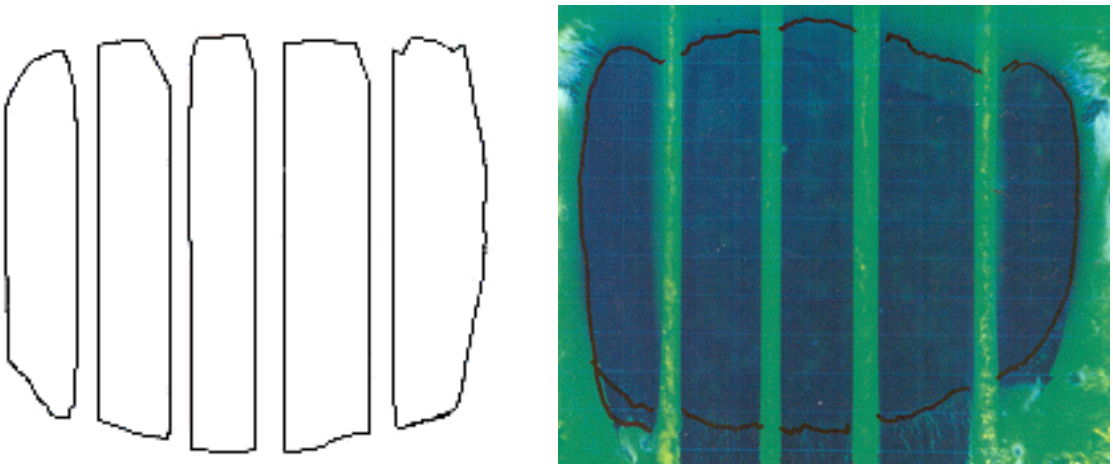


Figure 7.10: Rotating and stationary meshes after shrinking.

one time-step. At the same time, the contact patch shape undergoes certain deformation computed by the CSM solver in the preceding step. At this point, the meshes are subject to unshrinking. The moving mesh is rotated along with the tire, thus resulting in a modified connection between the two meshes. At the same time, this mesh is deformed following the contact patch modification. Finally, both meshes are shrunk, the next CFD iteration is performed, and the computed pressure field is communicated to the CSM solver.

7.6 Results

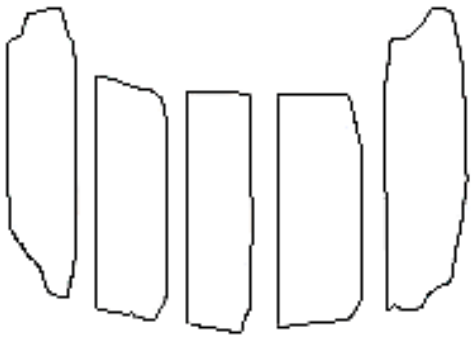


(a) Computed contact patch.

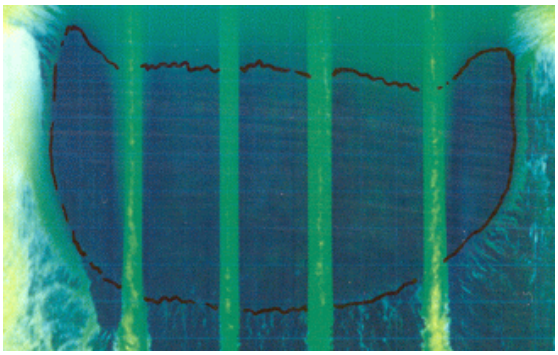
(b) Real contact patch.

Figure 7.11: Comparison of contact patch of non-treaded tire computed at 60 km/h with experimental data.

This section presents comparisons of obtained computational results with experimental data. Results for two speed regimes, namely 60 km/h and 80 km/h, are presented for non-treaded tire. One regime, namely 70 km/h, is presented for treaded tire. Contours

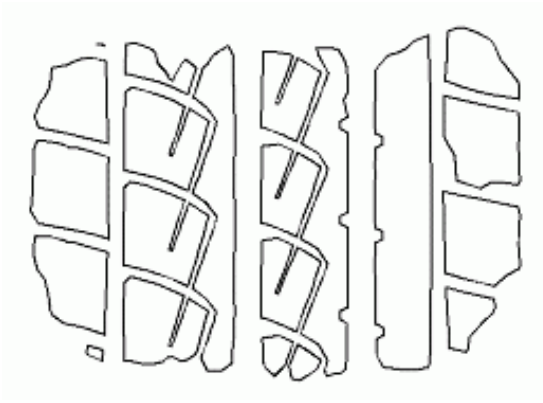


(a) Computed contact patch.

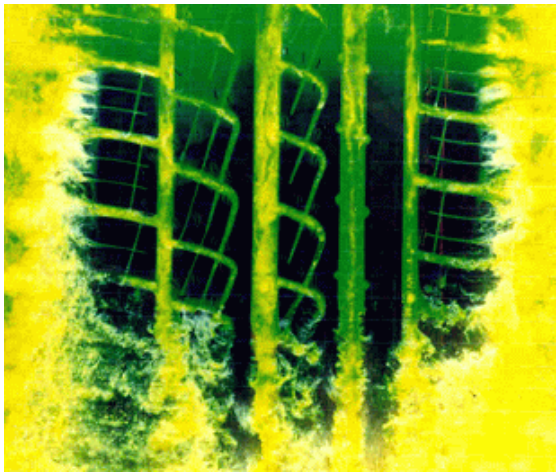


(b) Real contact patch.

Figure 7.12: Comparison of contact patch of non-treaded tire computed at 80 km/h with experimental data.



(a) Computed contact patch.



(b) Real contact patch.

Figure 7.13: Comparison of contact patch of treaded tire computed at 70 km/h with experimental data.

of computed contact patches were obtained as averages over the last ten iterations of the coupled procedure. The contours are measured in a horizontal plane located 0.01 mm above the road.

Figures 7.11 and 7.12 present results for a non-treaded tire at 60 km/h and 80 km/h, respectively. The computed shapes of contact patches are in relatively good agreement with experimental data. The same can be stated regarding the comparison between computed and experimental shapes of a contact patch obtained for treaded tire at 70 km/h and shown

in Figure 7.13. The darkest area on the experimental image corresponds to the shape of the contact patch.

The comparisons presented above successfully validate both the non-treaded and treaded numerical models.

Chapter 8

Conclusions and recommendations

Interest in fully automatic hexahedral mesh generation has grown significantly for the last decade. One of the major reasons for this growth is the natural suitability of hexahedral elements for numerical analysis. In Finite-Volume methods, capturing viscous effects near solid walls benefits from using hexahedra. Furthermore, hexahedral linear elements increase the accuracy of finite-element analysis. Finally, fewer hexahedral elements are required to mesh a domain as compared to tetrahedra for the same number of nodes. These factors make automatic hexahedral unstructured mesh generation one of the most attractive challenges among industrial demands in numerical design.

In recent years, attempts to solve the problem of automated unstructured hexahedral meshing have been focused on few principal approaches. Among others, the Overlay approach is considered as one of the most robust automatic tools capable of dealing with geometries of practically arbitrary complexity. The method appears to be a promising step towards full automation of hexahedral meshing process. That is the reason why the most critical aspects of this approach constitute the subject of this thesis.

A new approach for quality improvement of hexahedral meshes based on a low-cost simplicial untangling algorithm and a structural-analogy-based optimization methodology was presented in Chapter 4. Results prove the high efficiency of the combined approach at relatively low computational costs. Only the optimization part of the combined approach takes a significant amount of computational time, whereas both the untangling and smoothing are very fast and computationally inexpensive. The approach is not only capable of correcting concave cells, but also improving their quality without optimizing them directly due to the application of the averaging technique. The latter is the key element responsible for the overall speed-up of the combined approach. It is concluded that the combined approach is capable of improving mesh quality much more efficiently than stand-alone optimization or untangling algorithms. The main advantage of the combined approach is that similar mesh quality improvement can be achieved in shorter computational time with respect to optimization-based approaches.

The combined approach can still be improved in certain aspects. The Laplacian smoothing

exhibits undesirable behaviour in vicinity of concave geometrical entities, which creates the possibility for a respective mesh to become self-overlaid. Therefore, the Laplacian smoothing procedure can be upgraded to a more reliable version, for instance, the "smart" Laplacian smoothing [41] which changes nodal positions only if the change does not decrease mesh quality. Besides, other smoothing methods can be considered as substitutions of the Laplacian approach. Performance of the untangling procedure can be improved as well. Particularly, cases in which a false position of one node is responsible for the existence of more than one concave cells are of interest. Finally, the optimization procedure may benefit from employment of alternative mesh quality metrics.

An efficient methodology for generation of high-quality unstructured hexahedral meshes for viscous computations was developed in Chapter 5. The methodology is based on inflation of Euler mesh cells adjacent to solid boundaries and tangential refinement of these cells. The inflation step ensures that a smooth size distribution of new anisotropic cells is achieved. The methodology has shown reliable results for a set of real test cases of industrial interest. Inflation of buffer cells allows to satisfy the user-specified parameters of viscous layer and to provide smooth cell sizing in normal-to-wall direction simultaneously. The inflation procedure shows good performance near concave boundaries provided that the degree of concavity is not too high. Convergence rate of computations performed on meshes generated using the inflation procedure is higher, sometimes significantly, than that of computations on meshes generated without inflation.

Although the results of application of the methodology to industrial test problems are promising, certain improvements are still necessary to be introduced. Once again, the Laplacian smoothing employed in the methodology may produce self-overlaid meshes in highly concave regions. This effect is opposite to the effect of layer inflation. One possibility to solve the problem is to apply a certain space transformation which compensates high surface curvature. Smoothing should then be applied in the transformed space. Application of other smoothing approaches can be attempted as well.

A powerful procedure for deformation of unstructured hexahedral meshes was described in Chapter 6. The key feature of the procedure is application of quaternions, a mathematical tool for qualitative description of arbitrary rotations. Quaternions allow obtaining high quality deformed meshes even for boundary deformations involving large angle rotations. Another important element of the method is the use of a dual mesh for interpolation of deformation into a volume mesh. This feature results in faster convergence of the smoothing procedure employed in interpolation. Finally, the RBI approach is used for for constructing initial solutions. It also accelerates the interpolation procedure for cases in which deformations induced by different topological surfaces interfere with each other. Algorithmic structure of the method appears to be highly suitable for the framework of unstructured hexahedral non-conformal meshing. It has shown exceptional performance in certain cases.

The main disadvantage of the present approach is its computational cost. Namely, generation of initial solutions is currently based on distances to closest boundaries calculated for

each internal node. The direct distance calculation appears to be the most time-consuming part of the procedure. This drawback can be addressed by applying more sophisticated algorithms for computation of distance functions.

The presented work promotes the methodology of unstructured hexahedral non-conformal mesh generation a few steps forward. The high degree of automation of the meshing process as well as the efficient quality improvement tools enable to successfully mesh a wide class of complex geometries. Meshes for simulations of viscous flows can be obtained by means of the developed method for viscous mesh generation. High quality of these meshes allows for efficient modeling of various flows exhibiting viscous effects. Finally, numerous non-steady state problems that involve moving boundaries can be addressed using the presented mesh deformation procedure.

Bibliography

[1] MpCCI: Mesh-based parallel Code Coupling Interface. Specification of MpCCI v1.3. Fraunhofer Institute for Algorithms and Scientific Computing, Germany, April 2002.

[2] A. Aksenov, A. Dyadkin, and A. Gudzovsky. Numerical Simulation of Car Tire Aquaplaning. In *Computational Fluid Dynamics*, 1996.

[3] S. Altmann. *Rotations, Quaternions, and Double Groups*. Clarendon Press, 1986.

[4] C. Armstrong, D. Robinson, R. McKeag, T. Li, S. Bridgett, R. Donaghy, and C. McGleenan. Medials for Meshing and More. In *4th International Meshing Roundtable*, pages 277–288, Sandia National Laboratories, October 1995.

[5] T. Baker. Automatic Mesh Generation for Complex Three-Dimensional Regions Using a Constrained Delaunay Triangulation. *Engineering with Computers*, 5:161–175, 1989.

[6] T. Baker. Triangulations, Mesh Generation and Point Placement Strategies. In *Frontiers of Computational Fluid Dynamics*, Ithaca, New York, November 1994.

[7] T. Baker. Mesh Movement and Metamorphosis. In *10th International Meshing Roundtable*, pages 387–396, Sandia National Laboratories, October 2001.

[8] T. Baker and P. Cavallo. Dynamic Adaptation for Deforming Tetrahedral Meshes. In *14th AIAA Computational Fluid Dynamics Conference, Norfolk*, Norfolk, USA, June 1999.

[9] T. Baker and J. Vassberg. Tetrahedral Mesh Generation and Optimization. In M. Cross, B. Soni, J. Thompson, J. Hauser, and P. Eiseman, editors, *Numerical Grid Generation in Computational Field Simulations*, pages 337–349, 6th International Conference, University of Greenwich, July 1998.

[10] J. Batina. Unsteady Euler Spring Method for Three-Dimensional Unstructured Dynamic Meshes. *AIAA Paper 89-0150*, 1989.

[11] M. Bern and D. Eppstein. Flipping Cubical Meshes. In *10th International Meshing Roundtable*, pages 19–29, Sandia National Laboratories, October 2001.

[12] T. Blacker and R. Meyers. Seams and Wedges in Plastering: A 3D Hexahedral Mesh Generation Algorithm. *Engineering with Computers*, 2:83–93, 1993.

[13] T. Blacker and M. Stephenson. Paving: A New Approach to Automated Quadrilateral Mesh Generation. *International Journal for Numerical Methods in Engineering*, 32:811–847, 1991.

[14] F. Blom. Consideration on the Spring Analogy. *International Journal for Numerical Methods in Fluids*, 32:647–668, 2000.

[15] P. Boehmann, S. Wittchen, M. Shephard, K. Grice, and M. Yerry. Robust Geometrically-based, Automatic Two-Dimensional Mesh Generation. *International Journal for Numerical Methods in Engineering*, 24:1043–1078, 1987.

[16] E. Boender. Reliable Delaunay-Based Mesh Generation and Mesh Improvement. *Communications in Numerical Methods in Engineering*, 10:773–783, 1994.

[17] H. Borouchaki and P. George. Aspects of 2-D Delaunay Mesh Generation. *International Journal For Numerical Methods in Engineering*, 40:1957–1975, 1997.

[18] H. Borouchaki, P. George, F. Hecht, P. Laug, and E. Saltel. Delaunay Mesh Generation Governed by Metric Specifications. Part I. Algorithms. *Finite Elements in Analysis and Design*, 25:61–83, 1997.

[19] H. Borouchaki, P. George, and B. Mohammadi. Delaunay Mesh Generation Governed by Metric Specifications. Part II. Applications. *Finite Elements in Analysis and Design*, 25:85–109, 1997.

[20] A. Bowyer. Computing Dirichlet Tesselations. *The Computer Journal*, 24(2):162–166, 1981.

[21] A. Browne. Mathematical Analysis for Pneumatic Tire Hydroplaning. In *Surface Texture versus Skidding: Measurements, Frictional Aspects and Safety Features of Tire-Pavement Interactions*, pages 75–94. American Society for Testing and Materials, 1975.

[22] N. Calvo and S. Idelsohn. All-hexahedral Element Meshing: Generation of the Dual Mesh by Recurrent Subdivision. *Computer Methods in Applied Mechanics and Engineering*, 182:371–378, 2000.

[23] N. Calvo and S. Idelsohn. All-hexahedral Mesh Smoothing with A Node-Based Measure of Quality. *International Journal for Numerical Methods in Engineering*, 50(8):1957–1967, 2001.

[24] S. Canann. *Plastering and Optismoothing: New Approaches to Automated, 3D Hexahedral Mesh Generation and Mesh Smoothing*. PhD thesis, Brigham Young University, Provo, UT, 1991.

[25] S. A. Canann, M. B. Stephenson, and T. D. Blacker. Optismoothing: An Optimization-Driven Approach to Mesh Smoothing. *Finite Elements in Analysis and Design*, 13:185–190, 1993.

[26] S. A. Canann, J. R. Tristano, and M. L. Staten. An Approach to Combined Laplacian and Optimization-Based smoothing for Triangular, Quadrilateral, and Quad-Dominant Meshes. In *7th International Meshing Roundtable*, pages 479–494, Sandia National Laboratories, October 1998.

[27] R. Cass, S. Benzley, R. Meyers, and T. Blacker. Generalized 3D Paving: An Automated Quadrilateral Surface Mesh Generation Algorithm. *International Journal For Numerical Methods In Engineering*, 39:1475–1489, 1996.

[28] M. Castro-Diaz, F. Hecht, and B. Mohammadi. New Progress in Anisotropic Mesh Adaptation for Inviscid and Viscous Flow Simulations. In *4th International Meshing Roundtable*, Sandia National Laboratories, 1995.

[29] M. Castro-Diaz, F. Hecht, B. Mohammadi, and O. Pironneau. Anisotropic Unstructured Mesh Adaptation for Flow Simulations. *International Journal for Numerical Methods for Engineering*, 25(4):475, 1997.

[30] G. Coussement. Euler-Lagrange Formulation of A Mesh Optimization and Adaptation Method and Unification with Some Other Popular Methods. In *5th International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, Mississippi, USA, April 1996.

[31] S. Davis. NACA 64A010 (NASA Ames Model). Oscillatory Pitching. Compendium of Unsteady Aerodynamic Measurements, Data Set 702, AGARD, August 1982.

[32] H. de Cogny, M. Shephard, and M. Georges. Explicit Node Point Smoothing within The Octree Mesh Generator. Technical Report 10-1990, SCOREC, 1990.

[33] C. Degand and C. Farhat. A Three-Dimensional Torsional Spring Analogy Method for Unstructured Dynamic Meshes. *Computers and Structures*, 80:305–316, 2002.

[34] M. Delanaye, Ch. Hirsch, and K. Kovalev. Untangling and Optimization of Unstructured Hexahedral Meshes. *Journal of Computational Mathematics and Mathematical Physics*, 43(6):845–853, 2003.

[35] M. Delanaye, A. Patel, K. Kovalev, B. Leonard, A. Lani, and Ch. Hirsch. From CAD to Flow Solution With Adaptive Unstructured Hexahedral Meshing. In *5th World Congress of Computational Mechanics*, Vienna, Austria, 2002. Vienna University of Technology.

[36] M. Delanaye, K.-F. Tchon, A. Patel, and Ch. Hirsch. All Hexahedra Unstructured Grid Generation. In *ECCOMAS 2000 Conference*, Barcelona, Spain, 2000.

[37] B. Delaunay. Sur la Sphere Vide. *Bulletin of Academy of Sciences, USSR*, 117:793–800, 1934.

[38] C. Farhat, C. Degand, B. Koobus, and M. Lesoinne. Torsional Springs for Two-Dimensional Unstructured Fluid Meshes. *Computer Methods in Applied Mechanics and Engineering*, 163:231–245, 1998.

[39] D. Field. Laplacian Smoothing and Delaunay Triangulations. *Communications and Applied Numerical Methods*, 4:709–712, 1988.

[40] N. Folwell and S. Mitchell. Reliable Whisker Weaving via Curve Contraction. In *7th International Meshing Roundtable*, pages 365–378, Sandia National Laboratories, October 1998.

[41] L. Freitag. On Combining Laplacian and Optimization-Based Mesh Smoothing Techniques. In *Trends In Unstructured Mesh Generation*, volume 220, pages 37–43, ASME, July 1997.

[42] L. Freitag, M. Jones, and P. Plassmann. An Efficient Parallel Algorithm for Mesh Smoothing. In *4th International Meshing Roundtable*, pages 47–58, Sandia National Laboratories, 1995.

[43] L. Freitag and P. Knupp. Tetrahedral Element Shape Optimization via The Jacobian Determinant and Condition Number. In *8th International Meshing Roundtable*, pages 247–258, Sandia National Laboratories, September 1999.

[44] L. Freitag and P. Knupp. Tetrahedral Mesh Improvement via Optimization of The Element Condition Number. *International Journal For Numerical Methods In Engineering*, 53:1377–1391, 2002.

[45] L. Freitag and C. Ollivier-Gooch. Tetrahedral Mesh Improvement Using Swapping and Smoothing. *International Journal For Numerical Methods In Engineering*, 40:3979–4002, 1997.

[46] L. Freitag and P. Plassman. Local Optimization-based Simplicial Mesh Untangling and Improvement. *International Journal in Numerical Methods in Engineering*, 49(1):109–125, 2000.

[47] W. H. Frey and D. H. Field. Mesh Relaxation: A New Technique for Improving Triangulations. *International Journal For Numerical Methods In Engineering*, 31(6):1121–1133, 1991.

[48] R. Garimella. *Anisotropic Tetrahedral Mesh Generation.* PhD thesis, Rensselaer Polytechnic Institute, 1998.

[49] P. L. George. *Automatic Mesh Generation*, pages 234–236. Masson, Paris, 1991.

[50] P. L. George and H. Borouchaki. *Delaunay Triangulation and Meshing: Application to Finite Elements*, page 413. Hermes, 1998.

[51] P. J. Green and R. Sibson. Computing the Dirichlet Tesselation in the Plane. *The Computer Journal*, 21(2):168–173, 1977.

[52] H. Grogger and M. Weiss. Calculation of the Three-dimensional Free Surface Flow Around an Automobile Tire. *Tire Science and Technology*, 24(1):39–49, 1996.

[53] H. Grogger and M. Weiss. Calculation of the Hydroplaning of a Deformable Smooth-Shaped and Longitudinally-Grooved Tire. *Tire Science and Technology*, 25(4):265–287, 1997.

[54] P. Hansbo. Generalized Laplacian Smoothing Of Unstructured Grids. *Communications in Numerical Methods in Engineering*, 11:455–464, 1995.

[55] O. Hassan, E. Probert, K. Morgan, and J. Peraire. Unstructured Tetrahedral Mesh Generation for Three-Dimensional Viscous Flows. *International Journal for Numerical Methods in Engineering*, 39:549–567, 1996.

[56] B. Helenbrook. Mesh Deformation Using the Biharmonic Operator. *International Journal for Numerical Methods in Engineering*, 56:1007–1021, 2003.

[57] W. Horne and R. Dreher. Phenomena of Pneumatic Tire Hydroplaning. Technical Note D-2056, NASA Langley Research Center, 1963.

[58] S. Ivanenko. Harmonic Mappings. In *Handbook of Grid Generation*, chapter 8, pages 1–43. CRC Press, 1998.

[59] S. Ivanenko. Optimality Principles for Nondegenerate Grids. In *Grid Generation: New Trends and Applications in Real World Simulations*, Russian Academy of Sciences, Computational Center, 2001.

[60] O.-P. Jacquotte. A Mechanical Model for a New Mesh Generation Method in Computational Fluid Dynamics. *Computer Methods in Applied Mechanics and Engineering*, 66(3):323–338, 1988.

[61] O.-P. Jacquotte. Recent Progress On Mesh Optimization. In *3rd International Conference on Numerical Grid Generation in Computational Fluid Dynamics*, Barcelona, Spain, June 1991.

[62] O.-P. Jacquotte. Structured Grid Generation: Algebraic Method, Optimization and Adaptation. In *Grid Generation, Lecture Series 1994-2002, Von Karman Institute for Fluid Dynamics*. 1994.

[63] O.-P. Jacquotte and G. Coussement. Structured Mesh Adaptation: Space Accuracy and Interpolation Methods. *Computational Methods in Applied Mechanics and Engineering*, 101:397–432, 1992.

[64] N. Jones and S. Wright. Algorithm For Smoothing Triangulated Surfaces. *Journal of Computing in Civil Engineering*, ASCE 1:85–102, 1991.

[65] Y. Kallinderis, A. Khawaja, and H. McMorris. Hybrid Prismatic/Tetrahedral Grid Generation for Complex 3-D Geometries. *AIAA-95-0211*, 1995.

[66] Y. Kallinderis and S. Ward. Prismatic Grid Generation for Three-Dimensional Complex Geometries. *AIAA Journal*, 31(10):1850–1856, 1993.

[67] A. Khawaja, H. McMorris, and Y. Kallinderis. Hybrid Grids for Viscous Flows Around Complex 3-D Geometries Including Multiple Bodies. *AIAA-95-1685*, 1995.

[68] D. Kholodar, S. Morton, and R. Cummings. Deformation of Unstructured Viscous Grids. In *43rd AIAA Aerospace Meeting and Exhibit*, Reno, USA, January 2005.

[69] P. Kinney. CleanUp: Improving Quadrilateral Finite Element Meshes. In *6th International Meshing Roundtable*, pages 437–447, Sandia National Laboratories, October 1997.

[70] P. Knupp. Winslow Smoothing on Two Dimensional Unstructured Meshes. In *7th International Meshing Roundtable*, pages 449–457, Sandia National Laboratories, October 1998.

[71] P. Knupp. Achieving Finite Element Mesh Quality via Optimization of The Jacobian Matrix Norm and Associated Quantities, Part I - A Framework for Surface Mesh Optimization. Technical Report SAND99-0110J, Sandia National Laboratories, 1999.

[72] P. Knupp. Achieving Finite Element Mesh Quality via Optimization of The Jacobian Matrix Norm and Associated Quantities, Part II - A Framework for Volume Mesh Optimization. Technical Report SAND99-0709J, Sandia National Laboratories, 1999.

[73] P. Knupp. Hexahedral Mesh Untangling & Algebraic Mesh Quality Metrics. In *9th International Meshing Roundtable*, pages 173–183, Sandia National Laboratories, October 2000.

[74] P. Knupp. A Method for Hexahedral Mesh Shape Optimization. *International Journal for Numerical Methods in Engineering*, 58:319–332, 2003.

[75] K. Kovalev, M. Delanaye, and Ch. Hirsch. Untangling and Optimization of Unstructured Hexahedral Meshes. In S. Ivanenko and V. Garanzha, editors, *Grid Generation: Theory and Applications*, Moscow, Russia, July 2002. Computational Center of the Russian Academy of Sciences.

[76] K. Kovalev and Ch. Hirsch. Automatic Generation of High-Quality Unstructured Hexahedral Meshes for High Reynolds Number Computations. In P. Diez and N.-E. Wiberg, editors, *Adaptive Modeling and Simulation*, pages 369–382, 2005.

[77] M. Leatham and J. Chappell. On The Rapid Regeneration of Hybrid Grids Due To Design Drivern Geometry Perturbation. In *6th International Conference on Numerical Grid Generation in Computational Field Simulations*, pages 553–542, July 1998.

[78] D. Levy, T. Zickuhr, J. Vassberg, S. Agraval, R. Wahls, S. Pirzadeh, and M. Hemsh. Summary of Data from the First AIAA CFD Drag Prediction Workshop. *AIAA Paper 2002-0841*, 2002.

[79] T. Li, R. McKeag, and C. Armstrong. Hexahedral meshing using midpoint subdivision and integer programming. *Computer Methods in Applied Mechanics and Engineering*, 24:171–193, 1995.

[80] R. Lohner. Matching Semi-structured and Unstructured Grids for Navier-Stokes Calculations. *AIAA-93-3348-CP*, 1995.

[81] R. Lohner. Generation of Unstructured Grids Suitable for RANS Calculations. In S. Idelsohn et al., editor, *Computational Mechanics - New Trends and Applications*, pages 1–11, CIMNE, Barcelona, 1998.

[82] R. Lohner, K. Morgan, and O. Zienkiewicz. Adaptive Grid Refinement for The Compressible Euler Equations. In I. Babuska et al., editor, *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, pages 281–297. Wiley and Sons, Ltd., 1986.

[83] D. Lynch and K. O'Neill. Elastic Grid Deformation for Moving Boundary Problems in Two Space Dimensions. In S. Young, editor, *Finite Elements in Water Resources*, University of Mississippi, 1980.

[84] L. Marechal. A New Approach to Octree-Based Hexahedral Meshing. In *10th International Meshing Roundtable*, pages 209–221, Sandia National Laboratories, October 2001.

[85] D. Martineau and J. Georgala. A Mesh Movement Algorithm For High Quality Generalized Meshes. In *42nd AIAA Fluid Dynamics Conference and Exhibit*, 2004.

[86] D. Mavriplis. Adaptive Mesh Generation for Viscous Flows Using Delaunay Triangulation. *Journal of Computational Physics*, 90:271–291, 1990.

[87] M. Mendenhall, R. Childs, and J. Morisson. Best Practices For Reduction of Uncertainties in CFD Results. In *41st AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, January 2003. AIAA-2003-0411.

[88] C. Mitchell and T. Tautges. Pillowing Doublets: Refining a Mesh to Ensure That Faces Share atMost One Edge. In *4th International Meshing Roundtable*, pages 231–242, 1995.

[89] MSC Software Corporation. *Theory and User Information*, 2003.

[90] N. Mukherjee. A Hybrid, Variational 3D Smoother for Orphaned Shell Meshes. In *11th International Meshing Roundtable*, pages 379–390, Sandia National Laboratories, 11th International Meshing Roundtable 2002.

[91] P. Murdoch and S. Benzley. The Spatial Twist Continuum. In *4th International Meshing Roundtable*, pages 243–251, Sandia National Laboratories, October 1995.

[92] Y. Nakajima, E. Seta, T. Kamegawa, and H. Ogawa. Hydroplaning Analysis by FEM and FVM: Effect of Tire Rolling and Tire Pattern on Hydroplaning. In *FISITA-2000 Conference*, Seoul, 2000.

[93] A. Oddy, J. Goldak, M. McDill, and M. Bibby. A Distortion Metric for Isoparametric Finite Elements. In *Trans. CSME*, number 38-CSME-32, 1988.

[94] T. Okano and M. Koishi. A New Computational Procedure to Predict transient Hydroplaning Performance of a Tire. Technical report, The Yokohama Rubber Company, 2001.

[95] V. Parthasarathy and S. Kodiyalam. A Constrained Optimization Approach to Finite Element Mesh Smoothing. *Journal of Finite Elements in Analysis and Design*, 9:309–320, 1991.

[96] A. Patel. *Development of an Adaptive RANS Solver for Unstructured Hexahedral Meshes*. PhD thesis, Universite Libre de Bruxelles, March 2003.

[97] A. Patel, B. Leonard, M. Delanaye, and Ch. Hirsch. Unstructured Unsteady Adaptive Simulations for External Aerodynamics. In *ECCOMAS 2000*, Barcelona, September 2000.

[98] W. Philips, C. Hailey, and G. Gerbert. Review of Attitude Representation Used for Air Kinematics. *Journal Aircraft*, 38(4):718–737, 2001.

[99] J. Pilliod and E. Puckett. Second-order Accurate Volume-Of-Fluid Algorithms for Tracking Material Interfaces. Technical Report LBNL-40744, Lawrence Berkeley National Laboratory, 1997.

[100] S. Pirzadeh. Viscous Unstructured Three-Dimensional Grids by the Advancing Layer Method. In *32nd Aerospace Sciences Meeting & Exhibit*, Reno, Nevada, 1994. AIAA-94-0417.

[101] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in FORTRAN. The Art of Scientific Computing.* Cambridge University Press, 1992.

[102] M. Price and C. Armstrong. Hexahedral Mesh Generation by Medial Surface Subdivision: Part I. Solids with Flat and Concave Edges. *International Journal for Numerical Methods in Engineering*, 40:111–136, 1997.

[103] M. Price, C. Armstrong, and M. Sabin. Hexahedral Mesh Generation by Medial Surface Subdivision: Part I. Solids with Concave Edges. *International Journal for Numerical Methods in Engineering*, 38(19):3335–3359, 1995.

[104] J. Samareh. Application of Quaternions for Mesh Deformation. Technical report, NASA/TM-2002-211646, 2002.

[105] R. Schneiders. Automatic Generation of Hexahedral Finite Element Meshes. In *4th International Meshing Roundtable*, pages 103–114, Sandia National Laboratories, October 1995.

[106] R. Schneiders. A Grid Based Algorithm for the Generation of Hexahedral Element Meshes. *Engineering with Computers*, 12(3-4):168–177, 1996.

[107] R. Schneiders. An Algorithm for the Generation of Hexahedral Element Meshes based on an Octree Technique. In *6th International Meshing Roundtable*, pages 183–194, Sandia National Laboratories, October 1997.

[108] R. Schneiders, R. Schindler, and F. Weiler. Octree-based Generation of Hexahedral Element Meshes. In *5th International Meshing Roundtable*, pages 205–216, Sandia National Laboratories, October 1996.

[109] D. Sharov and K. Nakahashi. Hybrid Prismatic/Tetrahedral Grid generation for Viscous Flow Applications. *AIAA Journal*, 36(2):157–162, 1998.

[110] J. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. http://www.cs.cmu.edu/ quake/triangle.html.

[111] K. Shoemake. Animating Rotation with Quaternion Curves. In *SIGGRAPH*, pages 245–254, July 1985.

[112] P. Spalart. Young-Person‘s Guide To Detached-Eddy Simulation Grids. Technical report, NASA/CR-2001-211032, 2001.

[113] M. Stephenson, S. Canann, T. Blacker, and R. Meyers. Plastering Progress Report I. Technical Report SAND89-2192, Sandia National Labs, 1992.

[114] A. Talbert and A. Parkinson. Development of an Automatic Two-Dimensional Finite Element Mesh Generator Using Quadrilateral Elements and Bezier Curve Boundary Definition. *International Journal for Numerical Methods in Engineering*, 29:1551–1567, 1990.

[115] T. Tam and C. Armstrong. 2D Finite Element Mesh Generation by Medial Axis Subdivision. *Advances in Engineering Software*, 13:313–324, 1991.

[116] T. Taniguchi, T. Goda, H. Kasper, and W. Zielke. Hexahedral Mesh Generation of Complex Composite Domain. In *5th International Conference on Grid Generation in Computational Field Simmulations*, pages 699–707, Mississippi State University, April 1996.

[117] G. Taubin. A Signal Processing Approach to Fair Surface Design. *Computer Graphics*, 29(Annual Conference Series):351–358, 1995.

[118] T. Tautges, T. Blacker, and S. Mitchell. The Whisker Weaving Algorithm: A Connectivity-Based Method for Constructing All-Hexahedral Finite Element Meshes. *International Journal for Numerical Methods in Engineering*, 39:3327–3349, 1996.

[119] K. Tchon, C. Hirsch, and R. Schneiders. OctreeBased Hexahedral Mesh Generation For Viscous Flow Simulations. In *13th AIAA Computational Fluid Dynamics Conference*, AIAA-971980, 1997.

[120] O. Ushakova. Conditions of Nondegeneracy of Three-dimensional Cells. A Formula of a Volume of Cells. *SIAM Journal on Scientific Computing*, 23:1274–1290, 2001.

[121] D. F. Watson. Computing The Delaunay Tesselation With Application To Voronoi Polytopes. *The Computer Journal*, 24(2):167–172, 1981.

[122] D. White and P. Kinney. Redesign of the Paving Algorithm: Robustness Enhancements through Element by Element Meshing. In *6th International Meshing Roundtable*, pages 323–335, Sandia National Laboratories, October 1997.

[123] A. Wick. Generation of Dynamic Grids Using Structural Analogy. In *17th GAMM Seminar Leipzig on Construction of Grid Generation Algorithms*, Leipzig, Germany, 2001.

[124] A. Wick. Grid Deformation Techniques for Two-Dimensional Hybrid Grids. In *10th International Meshing Roundtable*, pages 261–267, Sandia National Laboratories, October 2001.

[125] A. Wick, M. Buffat, and F. Thiele. A Novel Method for Generation of Dynamic Meshes. In *7th International Conference on Numerical Grid Generation in Computational Field Simulation*, Whistler, Canada, 2000.

[126] M. Wierse, J. Cabello, and Y. Mochizuki. Automatic Grid Generation with HEXAR. In M. Cross., B. K. Soni, J. F. Thompson, J. Hauser, and P. R. Eiseman, editors, *6th International Conference on Numerical Grid Generation in Computational Field Simulations*, pages 843–852, University of Greenwich, July 1998.