



Vrije Universiteit Brussel

Control Methodologies in Unstructured Hexahedral Grid Generation

Özgür Uğraş BARAN

A dissertation submitted
to obtain the degree of
Doctor of Applied Sciences
of the
Vrije Universiteit Brussel.

Promotors
Chris LACOR
Charles HIRSCH

Department of Mechanical Engineering
Vrije Universiteit Brussel

November 2005



Pirpura...

Acknowledgments

I would like to express my gratitude to my advisor Prof. Charles Hirsch, who graciously gave me the opportunity to realize the present thesis. I am also grateful to Prof. Hirsch for sharing his wisdom with me, not only in technical issues but also the life itself. Without the inspiration he has given to me, this study cannot be possible.

I would like to thank Marc Tombroff in the name of Numeca International, giving me the opportunity of using their computing facilities and their excellent software, which makes this study available. I also like to express my appreciation Michel Pottiez for his precious support on the software and for everything he taught me about efficient software development.

I would like to thank Cem Ozan Asma, for his over three years of unconditional support and unforgettable friendship.

I am thankful to Benoît Leonard, Pierpaolo Borrelli and Konstantin Kovalev not only for their support to compile this thesis but also for their great friendship. Without their support everything would be much more difficult. I also like to thank former Numeca employees Alpesh Patel and Miles Elsdon for the same reason.

And I cannot forget my friends Ghader Ghorbaniasl, Sergey Smirnov, Murat Gürpınar, Colinda Francke and the others.

Finally, I express my love to Berna for whom this work is dedicated. Her warmest love, unlimited support and unconditional optimism makes everything much easier.

Contents

1	Introduction	1
2	Unstructured Mesh Generation Technology	5
2.1	Triangular/ Tetrahedral Mesh Generation	7
2.1.1	Delaunay triangulation	7
2.1.1.1	Mesh control methodologies from Delaunay Triangulation	10
2.1.2	Advancing Front Triangulation	10
2.1.2.1	Mesh control methodologies from Advancing Triangulation	11
2.2	Hexahedral mesh generation	12
2.2.1	Ideal Hex-meshing Algorithm	13
2.2.2	Indirect methods	13
2.2.2.1	Mesh control with Indirect Methods	14
2.2.3	Advancing front methods	15
2.2.3.1	Paving and Plastering	15
2.2.3.2	Whisker-Weaving	16
2.2.3.3	Mesh control with Hexahedral Advancing Front Technique	18
2.2.4	Geometric Decomposition Methods	19
2.2.4.1	Medial Axis Methods	19
2.2.4.2	Mesh control with the Medial Axis Methods	21
2.2.4.3	Sweeping and mapped meshing	22
2.2.4.4	Automatic Decomposition Methods	25
2.2.4.5	Mesh Control with Geometric Decomposition Methods . .	26
2.2.5	Overlay Grid Methods	26

2.2.5.1	Mesh control with Overlay Grid Technique	29
2.3	Mesh Control Techniques	29
2.3.1	Laplacian Smoothing	30
2.3.2	Physics Based Methods	31
2.3.3	Optimisation Based methods	33
2.3.4	Mesh Refinement	33
2.4	Conclusion	35
3	Mesh Control Algorithm	39
3.1	Spring Analogy	40
3.1.1	Boundary conditions	42
3.1.2	Adaptation of method for non-conformal meshes	43
3.2	The notion of metric	47
3.2.1	Riemannian Metric	49
3.2.2	Anisotropic mesh adaptation using spring analogy	49
3.2.3	Application	50
3.3	Conclusion	52
4	Natural Metrics	55
4.1	Metric for Natural Layout Representation	56
4.2	Bubble Growth Technique	58
4.3	Theoretical Foundation	59
4.3.1	Notations	59
4.3.2	Elliptic Interpolation	61
4.3.3	Correction of metric size	63
4.4	Numerical Implementation	64
4.4.1	Adaptation of the Method to Hex Meshes	64
4.4.2	Bubble Quality Criterion	66
4.4.3	Algorithm for successive bubbles	67
4.4.4	Algorithm for Metric Extraction for single cell.	68
4.4.5	The Basic Algorithm For Natural Metric Generation	69
4.5	Improvements on the Algorithm	70

4.5.1	Skeletonisation for Natural Metric Extraction	71
4.5.2	Other Improvements	72
4.5.3	The Final Algorithm for the Bubble Growth	74
4.6	Results	75
4.7	Conclusion	76
5	Distance Map Metric	81
5.1	Distance Maps	82
5.1.1	Properties of the distance function	83
5.2	Hessian Matrix	86
5.2.1	Relations Between Hessian Matrix and Local Extrema	86
5.2.2	From Hessian Matrix to Riemannian Metric	87
5.3	Distance Map Metric	87
5.4	Application	89
5.4.1	Computation of the Distance map	89
5.4.2	Computation of the Hessian	90
5.4.3	Skeletonisation via distance maps	92
5.5	Conclusion	93
6	Mesh generation control	95
6.1	Application of the mesh generation control	96
6.1.1	Selection of metrics and stiffness functions	96
6.1.2	Selection of Distance Map Function	98
6.1.3	Numerical scheme	99
6.2	Meshing Applications	100
6.2.1	Mesh Generation Control for Compact Geometries	100
6.2.2	Mesh generation control for external aerodynamics domains	105
6.3	Conclusion	108

7	Numerical simulations	111
7.1	Turbomachinery application	111
7.1.1	Test Conditions	111
7.1.2	Computational grids and solution procedure	112
7.1.3	Results	113
7.2	External aerodynamics	115
7.2.1	Numerical simulation	116
7.3	Conclusion	117
8	Mesh control in simulation	123
8.1	The segment springs approach and moving boundary problem	123
8.2	Application of the method	125
8.2.1	Mesh movement	125
8.2.2	Numerical simulation	126
8.3	Conclusion	127
9	Conclusions and Recommendations	131
A	Skeletonisation by Thinning	135
B	Skeletonisation by Distance Transform	139
	Notation	143
	Bibliography	143
	Index	155

List of Figures

2.1	Voronoi Diagram around a set of points.	8
2.2	Delaunay triangulation	9
2.3	Advancing Front Triangulation	11
2.4	Schneiders' open problem [1]	12
2.5	Indirect hexahedral mesh generation methods	14
2.6	Paving algorithm as it was redesigned by White.	16
2.7	Spatial Twist Continuum (STC) and Whisker-weaving type hexahedral mesh generation	17
2.8	Medial Axis in two dimensions	19
2.9	Medial surface of a three-dimensional object	19
2.10	Hexahedral mesh generation by Sweeping	23
2.11	Mesh generation steps in overlay grid technique	27
2.12	Lineal Spring Analogy	32
2.13	Torsional Springs Analogy	32
3.1	The Spring Analogy	40
3.2	Mesh regularisation test for spring analogy.	43
3.3	Boundary treatment for a spring system	44
3.4	Two and four connected hanging nodes	45
3.5	Regularisation test for octree spring analogy point relocation algorithm.	47
3.6	The physical representation of spherical metric	48
3.7	Demonstration for metric based vertex relocation method.	51
3.8	Vertex relocation technique based on distance information of an image.	52
4.1	Spherical metric located at the center of an object.	57

4.2	Diverging parallel walls	58
4.3	The bubble growth algorithm.	59
4.4	Notation for bubble growth	60
4.5	The effect of point density on the tensor product	61
4.6	Metric calculation on a bubble.	62
4.7	Bubble growth in a volume	64
4.8	Skeletonisation process for natural metric extraction	73
4.9	Basic test case for Natural metrics extraction	77
4.10	Toy dinosaur test case for natural metrics extraction	78
5.1	Distance map calculated by city block type distance function	83
5.2	The equi-distance lines and metric definition	84
5.3	Medial axis from distance maps	85
5.4	Equi-distance surfaces of δ^2 from the distance map around the toy dinosaur geometry	90
5.5	Eigenvectors of the largest eigenvalues from the Hessian of the distance map	92
6.1	Comparison of the deformed meshes on boundary of the adapting mesh. . .	101
6.2	Comparison of the final meshes on moulding geometry	102
6.3	Orthogonality and the aspect ratio distribution for test cases applied on moulding geometry.	103
6.4	Comparison of the final meshes on volute geometry	104
6.5	Detail view of the final meshes at the diffuser section on the volute geometry	105
6.6	Orthogonality and the aspect ratio distribution for test cases applied on volute geometry.	106
6.7	Skeleton computed for DLR-F4 test case	107
6.8	Coarse grid generated around the DLR-F4 geometry.	107
6.9	Grid at the wing tip of the DLR-F4 configuration.	108
6.10	Orthogonality and the aspect ratio distribution for test cases applied on DLR-F4 geometry.	108
6.11	Cross-section of the optimised mesh at the wing section of DLR-F4 geometry.	109
7.1	Rotor-37 geometry	112

7.2	Structured and the unstructured meshes for Rotor-37 test case.	113
7.3	Rotor-37 test case, comparison of anisotropic efficiency	114
7.4	Rotor-37 test case, comparison of pressure ratio	114
7.5	Relative Mach number plots for Rotor-37 case for structured and unstructured meshes at the midspan.	115
7.6	Relative total pressure plots for unstructured Rotor-37 mesh.	116
7.7	Pressure distribution on DLR-F4 geometry for different mesh densities. . .	118
7.8	C_p distribution in various wing span sections for DLR-F4 geometry	119
7.9	Bottom view of DLR-F4 geometry for original, controlled and refined grids	120
7.10	Top view view of DLR-F4 geometry for original, controlled and refined grids	121
8.1	Deformed mesh via segment-spring analogy.	126
8.2	Pressure plot over oscillating NACA-0012 airfoil at different time steps. . .	128
8.3	Lift coefficient C_L versus angle of attack plot for oscillating NACA0012 wing	128
A.1	Skeletonisation via thinning	137
B.1	Skeletonisation with distance transform for Toy dinosaur test case.	141
B.2	Skeleton of volute geometry with distance transform.	142

List of Tables

4.1	The elliptic interpolation error for the surface mesh for the bar geometry .	66
4.2	The elliptic interpolation error for the surface mesh for the rotated bar geometry	67
4.3	The elliptic interpolation error for the surface mesh for the diamond shaped front.	67
7.1	Flow conditions for DLR-F4 test case for the numerical test.	116
8.1	Flow conditions for NACA0012	127

List of Algorithms

3.2	Spring analogy for non-conformal hexahedra	46
4.2	Algorithm for growth for the bubble $\mathcal{N}_k$ of order k around cell $\mathcal{S}$	68
4.3	Metric Calculation algorithm for single cell	69
4.4	The basic algorithm for natural metric extraction in mesh $\mathcal{T}$	70
5.2	Distance map calculations	90
5.3	Hessian of the distance function	91
6.2	Final algorithm for mesh generation control	100
A.2	Skeletonisation by thinning	136
B.2	Skeletonisation with distance map.	140

Chapter 1

Introduction

With the remarkable progress accomplished by countless researches in the last decades, the numerical simulation techniques in different scientific and engineering fields are being accepted and widely adopted by the industry. With the introduction of accurate new numerical methods, progress in the meshing techniques, reduced cost of computational power, increasing expertise in practice and finally efficient user interfaces combining them, these techniques have become a reality as a standard step of design process in almost every engineering field. The practical applications may include one of the different physical problems such as fluid mechanics, heat transfer, solid mechanics etc, or combination of them like fluid structure interaction problems, in very different scales ranging from micro scales as it is in electronics to very large scales like earth simulations.

Despite the differences in scales of the problems, dissimilarities in algorithmic features and physical characteristics of the problems, the common element in all these simulations is the computational grid. Almost all of the computational methods rely on a computational grid to perform the simulation, where the computational domain is subdivided into simpler elements, such as triangles, tetrahedra, hexahedra etc. A usually disregarded fact is, the computational mesh is as important as the simulation algorithm itself in numerical modelling. Computational grid often plays an important role in the accuracy of the solution, simulation time, and even the computational requirements. Hence, a high quality mesh generation algorithm is a priority for industry.

The hexahedral meshes are considered as superior to the tetrahedral ones for different reasons. First, many researches in literature indicate that hexahedral meshes give unsurpassed accuracy in similar configuration. The hexahedron keeps accuracy with directional sizing, which is very important in boundary layer calculations. Hexahedral meshes reduces the cell count four to ten fold compared to tetrahedral meshes, which saves the computational time and computer resources. All these features draw the attention of the industry on the hexahedral methods.

However, with the increasing industrial demand on hexahedral methods, new constraints are introduced. A decade ago, the only industrial solution for hexahedral meshing was

based on multiblock structured meshes, which provide highest quality meshes, however, requires enormous amount of user intervention. As the complexities of geometries to be meshed for real world simulations grow enormously, these kinds of methods become impractical, since the human effort to mesh them becomes very expensive and small changes in the geometry during design process is very difficult to handle. As the trend is to make the simulations with the finest possible details with the increased availability of the computational resources, these mesh generators are not practical anymore.

The interest on the more efficient hexahedral meshing led to an expansion on the academic research in all automatic full hexahedral mesh generation techniques. However, it is realized that, automatic hexahedral meshing is an extremely complex topic, and despite the expectations, most of the tetrahedral techniques cannot be extended to the hexahedral problems. After fifteen years of research, a couple of new algorithms are introduced in the field. Yet, most algorithms came with their sets of problems. First, most of these algorithms are not universal. Although they are automated methods, they can be applied to a limited sets of problems. Many of the researches have diverted into hex-dominant methods to overcome these difficulties. Some of them keep some degree of user intervention. After all, automatic all hexahedral meshing remains a hot topic after many years.

Among all those methods, the overlay grid method seems to be the only industrially accepted method. In this method, an initial mesh encompassing the geometry is locally adapted to the geometry by refining the cells at necessary points, then the cells intersecting the boundary as well as the ones outside of the geometry are removed, and then the topology is captured. The algorithm can be applied to the geometries with any complexity while minimising the human effort. In fact, it is possible to eliminate the human effort. The present thesis is conducted on such an overlay-grid based mesh generator named HEX-PRESS. The mesh generator is part of a complete fluid mechanics software of the Belgian company Numeca International. This thesis is conducted with a close collaboration with Numeca, which is a well established company in turbomachinery and other aerodynamics simulations.

The overlay grid method has its own sets of inefficiencies as it is discussed in the next chapter. One of the most significant drawbacks of this method is its directional insensitivity. The algorithm is very sensitive on the selection of the initial mesh. Therefore, in order to capture the sections of the geometry where the initial mesh cells don't share the same orientation, many refinement iterations are necessary. In addition, very finer details require many adaptation steps. The extra refinement effort at those locations results in very high cell counts in final mesh.

Although the industrial trend is to perform simulations on denser meshes, it is wiser to keep the cell count at a minimum and allow the user to increase the cell density only in the regions required for a physically correct simulation. Therefore, reduction of the number of cells produced by the mesh generator becomes a requirement, as more complex simulations are made on extremely complex geometries. This study addresses this issue and tries to implement the orientation and size control on overlay grid full automatic hexahedral mesh

generation.

Therefore, in this thesis control methodologies on hexahedral mesh generation is developed to produce optimal meshes. Due to the enormous complexity of the hexahedra, the main interest in automatic hexahedral meshing research has been on producing valid meshes. Control methodologies for hexahedral meshes is a relatively new topic, and this thesis is aimed to fulfil this requirement. It should be noted that, mesh control methodologies are not employed for mesh generation only, but also used for controlling the mesh during the simulation. Therefore, in this thesis, a general mesh control technique, which can be extended for a wider range of meshing and remeshing requirements is presented.

The approach taken in this work consists in making an existing non-conformal octree hexahedral grid generation algorithm more sensitive to local sizes and directions. To achieve this goal, a spring analogy based algorithm using different local size metrics around an arbitrary geometry is developed. Different metrics are tested for this purpose, including metrics obtained from distance maps information and Natural metrics. These metrics are introduced to solve the problem of local size information. Once this information is obtained, the mesh is adapted considering this information using a spring-analogy based algorithm, where the mesh is converted to a network of springs with certain stiffness. The mesh is then adapted dynamically at each refinement step applied to fit the mesh to the geometry, therefore the overlay grid is better oriented with the geometry and cell sizes adapted to the local sizes, hence less refinement steps are necessary.

The thesis is divided into nine chapters.

In Chapter 2, a literature review on unstructured hexahedral mesh generation is presented with an emphasis on mesh control techniques. The chapter starts with a short review of the tetrahedral meshes. Unlike the hexahedral ones, tetrahedral meshes have good control features, so the possibility of extending such techniques for the hexahedral meshes are discussed. Then the mesh hexahedral mesh generation techniques are presented, in order to give an overview on the research field. In the same chapter, mesh control features for such methods are discussed. The chapter completes with a review of general mesh control techniques and a conclusion.

In Chapter 3, a mesh control technique, named spring-analogy is introduced. Spring analogy relies on replacing meshes with a network of springs. The chapter starts with a theoretical introduction of the technique. Then the notion of the metric is introduced. A metric is a symmetric positive definite matrix, which can be dissembled to its eigenvalues and eigenvectors. These metrics can be used to control the mesh cell sizes using the spring analogy. At the end of the chapter, some numerical tests are shown.

In Chapter 4, a metric, named *the natural metric*, is introduced to be used with the spring analogy based mesh generation control technique. The method relies on expanding an elastic ball starting from the medial axis of the geometry and fitting an elliptic metric for the shape of this bubble. Then this metric can be used to control the mesh. The chapter starts with the presentation of the idea, that is followed by the theoretical development

and numerical implementation. The chapter concludes with examples and the discussion of the results.

In Chapter 5, another metric which is based on distance maps is presented. A distance map is defined on a background grid which contains the data of distance to the closest boundary of the geometry for each cell on the background grid. The distance map gives precious information on the local thickness of a given geometry. The information about local orientation inside the geometry can also be extracted from the same distance map using the Hessian matrix of the distance function.

Chapter 6 presents the application of the mesh control methodologies on the unstructured hexahedral mesh generation applications. In this chapter, the theory developed in the Chapters 3-5 is compiled into a complete mesh generation control system. The different anisotropic metrics are introduced to be used as means of automatic mesh manipulation. The proposed algorithm is verified on different geometric configurations, including compact geometries and external aerodynamics solution domains. The performance of the controlled mesh generation algorithm is compared with the uncontrolled one.

Chapter 7 shows numerical results of some simulation of fluid mechanics problems on the meshes generated by the method presented in this thesis. An internal and an external aerodynamics configurations are given as validation test cases. Rotor-37 test case, which is a validation test case for a high-pressure compressor inlet stage of aircraft turbine engine. This test case is designed by NASA as a validation case for turbomachinery flow solvers, and experimental data is available for comparison with numerical solution. The simulation is performed on the mesh produced by the algorithm developed in this thesis and results are compared with the solution on a structured grid. The external aerodynamics test case is DLR-F4 wing-body configuration, which is a validation test case used for the AIAA drag prediction workshop. Experimental data is also available for this test case. The numerical simulations are performed on a very coarse mesh produced by the methods described in the thesis, then the mesh is adapted. The flow solutions are compared with the result obtained by a reference mesh of similar size, produced using conventional methods.

Chapter 8 extends the mesh control methodologies to numerical simulation-stage. As a model problem, moving boundary problems are considered. In such problems, the boundary shape evolves in time and the mesh vertices should be moved accordingly to comply with this movement. The updated mesh should preserve the features of the original mesh in terms of quality and anisotropic size metric. To achieve this goal, the segment-spring analogy is introduced. Large boundary movement handled with this method is demonstrated. The quality of the updated mesh is verified by numerical simulation around an oscillating NACA0012 airfoil. The numerical solution is compared with experimental data.

The thesis concludes with the general conclusions and recommendations.

Chapter 2

Unstructured Mesh Generation Technology

Computational simulation techniques for physical problems have become an important part of the industrial design process. Almost all of those simulations are conducted on a computational grid, where the numerical approximations of the physical laws are simulated on the elements of the mesh. These meshes consist of simple shapes like tetrahedra, hexahedra or their combination. In recent years, industrial interest on the hexahedral grid generation is increasing steadily. There are various reasons for this, such as much improved accuracy compared to tetrahedral meshes [2–4], reduced total cell count up to 4 to 10 fold [5] and directional sizing without losing accuracy, which is a very important property in boundary layer simulations.

As the expertise on numerical simulation techniques increases, it is seen that for an accurate simulation, quality of the mesh is as important as the simulation algorithm itself. There is no universal definition of the mesh quality. Instead, quality metrics of the mesh can be dependent on the simulation itself. The mesh quality for a simulation consists of different measures. For many algorithms, total cell count of the mesh is the limiting factor, as it is in implicit computational structural mechanics codes. Many computational fluid mechanics codes require imposed anisotropy in regions that are close to the solid wall to simulate the boundary-layer. For almost all codes, cell orthogonality improves the solution quality.

These quality requirements can be extended depending on the numerical simulation and they bring together the idea of controlled mesh generation. Especially, strong industrial demand on numerical simulation techniques brings the need for highest quality simulation on smallest number of cells. Therefore, meshing the domain is not sufficient; but the meshing process itself should be controlled to meet these demands.

Structured meshing technology, which responds the demand to hexahedral mesh generators of the industry exists for almost three decades. The quality of the structured multi-block Cartesian meshes is unsurpassed, where the control of the mesh features are

also possible. Unfortunately, as the simulations on the more complex geometries are required, these mesh generators are not viable alternatives, since they require great amount of human effort. For real-world problems, this process can take months, which is not acceptable according to industrial standards. Halpburn [6] reports that, for an industrial finite element simulation cycle, time required for building finite element models dominates the total analysis time, accounting up to 90 percent. Therefore, a significant amount of research is conducted on fast automatic mesh generation technologies for both tetrahedral [7] and hexahedral [8, 9] cases. All automatic mesh generation techniques show a significant progress in the recent years and made these techniques available for the industry. Nowadays, the use of all-automatic hexahedral mesh generation methods is increasing rapidly, as they become mature.

On the other hand, the same thing cannot be said for control techniques for these methods. Although, there are plenty of studies on the tetrahedral/ triangular mesh control, hexahedral mesh control technologies are very rare. The control of the automatic hexahedral mesh generation is very crucial to meet industrial demand on mesh quality. Therefore, development of the control methodologies for all-automatic hexahedral mesh generation techniques is aimed in this study.

There are various difficulties to fulfil this objective. The main challenge is the enormous complexity of hexahedral meshes. Hexahedral meshes are known to be significantly more complex than tetrahedral ones, and the number of possible operations for the alteration of the mesh connectivity is limited. Mesh conformity is difficult to maintain, which is the main reason why the research on the hexahedral mesh generation is focused on the generation of valid meshes rather than controlling the process. Control methods developed for the structured meshes don't have desired automation for the unstructured case, and even if they are usable, they should be altered significantly for the more complex connections of the unstructured case.

The most extensive research in the subject of mesh generation control methodologies is conducted in the triangular-tetrahedral mesh generation field. Due to the very different characteristics of the two kinds of meshes, these techniques cannot be used directly in the quadrilateral-hexahedral case. Nevertheless, controlled tetrahedral mesh generation is a very active research field due to their relative simplicity, great flexibility and additionally, numerous finite element simulation codes available for many different engineering problems, which depends on these meshes. Therefore, the experience on the control methodologies on tetrahedral mesh generation is too extensive to be ignored. Although the control algorithms may not be used directly, many ideas used to control tetrahedral meshes may be transferred to hexahedral case.

There are a couple of options to apply control methodologies on hexahedral mesh generation, such as mesh relocation methods, or mesh optimisation algorithms. The hexahedral control techniques are generally similar to that of Cartesian mesh generators, additionally, some of the tetrahedral methods, which don't utilise the alteration of the mesh topology can be used for this purpose. The selection of the control algorithm should also be made

carefully to achieve a reliable and robust mesh control mechanism.

Therefore, in this chapter, the triangular/tetrahedral mesh generation algorithms will be presented to investigate the possibility of extending the control methodologies developed on such meshes to the hexahedral meshes. Then, state of the art on hexahedral mesh generation techniques will be presented after discussing properties of hexahedral meshes. In this section, a flexible algorithmic basis for developing a fully controlled automatic mesh generation system will be investigated. This section is followed by a survey on the mesh control methodologies applied on the hexahedral meshes. The chapter concludes with the proposal of an algorithm that applies control methodologies on hexahedral meshes.

2.1 Triangular/ Tetrahedral Mesh Generation

Tri/ Tetrahedral mesh generators are by far the most popular unstructured grid generators. One of the main reasons of this popularity is the wide availability of unstructured finite element codes. Finite element algorithms are much easier to write on triangles/tetrahedron due to their special geometric features. Also conformal triangular-tetrahedral mesh generation in complex geometries is much easier than quadrilateral-hexahedral meshes. For these reasons, research on conformal tetrahedral mesh generation algorithms are expanded and industrial applications exist by early 80s. Today, although application specific implementations are sometimes necessary, these methods are well studied, reliable and widely adopted [7].

Algorithms can be divided into three categories: octree type, Delaunay type and advancing front type. The Octree method, similar to the hexahedral version, depends on local subdivision of the cell at cell-boundary intersections. The method is non-body conforming and not very much accepted by industrial applications, rather used by academic purposes. They do not present good mesh control features, therefore, in this review the latter methods are analysed.

These two methods are often compelled as competing approaches. The Delaunay triangulation offers attractive mathematical properties and merely refers to a particular connectivity associated with a given set of vertices. The advancing front approach offers a high quality point placement strategy while imposing a specific ordering of the element generation process and the integrity of the boundary.

2.1.1 Delaunay triangulation

Delaunay triangulation is undoubtedly the most popular and most profoundly studied mesh generation technique. A very extensive and detailed literature can be found on the topic [10, 11]. The concept is based on Delaunay criterion illustrated by Boris Delaunay [12] in 1934. The notion is not directly associated to mesh generation, but is a principle that is used to form high quality triangulation.

The Delaunay triangulation is correlated to Voronoï diagrams, which associates regions of the space to the closest point in a set of points. The Voronoï diagram around a set of nodes is shown in the Figure 2.1. The non-boundary nodes have their closed Voronoï regions, which are shaded in the same figure. Every point inside this region is closest to the associated node. Consider the point O at the intersection of three Voronoï edges, marked with the circle in the mentioned figure. This point, by definition should be at the same distance to three points. Consider a triangle is drawn between the nodes, the Voronoï edges of which are intersected at the point O . Therefore, point O is located at the barycentre of this triangle. The circumcircle around the triangle (or simplex) S_i does not contain any other nodes, since its centre is located at the intersection of the Voronoï edges and consequently, it is positioned at the centre of the circle. Moreover every other point in the set cannot be contained by this circle, due to the Voronoï property. Applying this attribute to every intersection of edges yields a unique triangulation called Delaunay Triangulation.

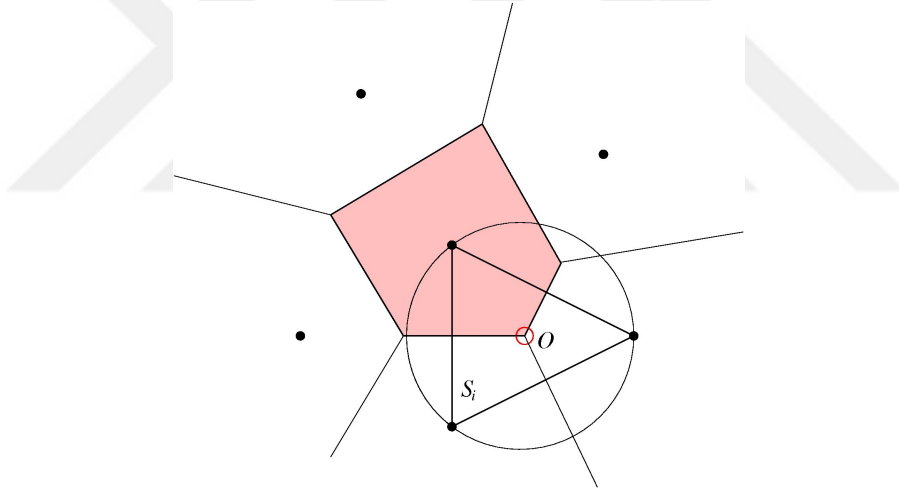


Figure 2.1: Voronoï Diagram around a set of points. Regions of space are associated to the points, and the equidistance edges separates these regions.

The concept also extends in higher dimensions, for example, in 3D, a unique Delaunay tetrahedralization exists, where circumspheres of the tetrahedra can be proven not to contain any other nodes. This analogy can be applied in reverse manner as well. One can examine a triangulation in a set of point for Delaunay property. An example of the Delaunay criterion in two dimensions can be seen in Figure 2.2. On the left, the non-Delaunay type of triangulation is shown. Flipping the edge in the middle yields another set of triangles, which complies the Delaunay property as it is seen in the right. This operation is called swapping and this technique is the most common tool for creating Delaunay triangulation. Swapping operation also extends for three dimensions, where, three non-Delaunay tetrahedra, which share one common edge, are translated to Delaunay pairs by swapping

this edge in any directions through non-shared faces.

Figure 2.2: Delaunay Triangulation. The triangulation on the left is not Delaunay type, since circumcircle of the triangles contains other points. Flipping the edge, the triangulation on the right is Delaunay type.

As it is mentioned above, the Delaunay triangulation is not a meshing method, but a criterion to connect the nodes as triangles. This connection criterion is utilised in meshing first in the early 80s with the pioneering works of Watson, Tanemura and Lawson [13–15]. These algorithms are extended and perfected by various researchers including Baker [16], Wright [17], Chew [18]. Today the Delaunay triangulation is a well-studied topic and, although application specific implementations would be diverse, the industrial codes based on them are automated and robust. The use of the Delaunay triangulation is not limited to the mesh generation but it finds itself a wide range of application fields from data interpolation to computer animation.

The Delaunay mesh generation starts with meshing of a set of surface points. The initial simplexes are changed to the Delaunay ones. Nevertheless, the mesh obtained in that way is nothing but a set of simplexes connecting a boundary to the opposing boundary, in other words there are no interior cells. Thus, to achieve a good quality triangulation inside the mesh should be refined. To achieve this, new points should be inserted inside the geometry. The point insertion strategy actually makes the difference of each algorithm. A good point insertion method enhances the overall quality of the mesh and considered as the most important step of the Delaunay meshing. The point insertion strategy can be very application specific and the literature on them is enormous. Review of this topic is beyond the intention of this study. A good overview, however, can be found in the excellent book of Georges and Borouchaki [10].

Point insertion may alter the Delaunay property so that it should be coupled with the edge swapping. The operation shown in Figure 2.2 is the most basic one, yet, it is the main operation in two dimensions. In three dimensions, swapping is more complex and there are a number of swap operations for different connections. For more detailed information, see [10, 19]

The most difficult issue in Delaunay mesh generation is to keep the surface integrity, since the Delaunay triangulation does not necessarily have edges and surfaces on the boundary of the geometry. Some extra work is necessary to capture the mesh. Several alternative algorithms exist in the literature that addresses this problem [19, 20].

2.1.1.1 Mesh control methodologies from Delaunay Triangulation

Delaunay triangulation is a very well studied method. The application areas are very wide and are not limited to the mesh generation, therefore the method offers a great code reuse. This feature attracted many researchers from different fields, resulting in tremendous amount of research on the topic, and many novel ideas to appear. Nowadays, countless implementations based on Delaunay triangulation are available commercially, as well as in public domain.

The main research on Delaunay type methods has been on the mesh control side. Indeed, the Delaunay criterion is merely a method to detect the features triangulation. The meshing basically consists of the control of inserting new nodes, and arranging the mesh connection with Delaunay criterion. In other words the control of meshing is the main meshing process itself. The control options are very wide and one can select the control mechanism of the meshing process for very different needs and with very different constraints. With a careful control on the point insertion relocation, very high quality meshes that fit the requirements of the numerical simulation can be obtained. That explains the popularity of Delaunay methods.

Unfortunately, it is not possible extend the Delaunay property for the hexahedron. In addition, many operations that are crucial in the Delaunay type methods, like edge swapping, are not possible in hexahedral meshes. Therefore, the method itself is not applicable for our needs.

However, some of the control mechanisms that are used in these algorithms can be extended to the hexahedral meshes by adapting them to the hexahedral control mechanisms. There is huge amount of literature on the local feature extraction and control mechanisms applied on the Delaunay triangulation, some of which can be utilised for the purpose of controlling the hex-meshing. The metric based control mechanism is one of those. In these methods, a continuous metric field is defined inside the domain, which is used as a Delaunay criterion in Riemannian space. Of course, the Delaunay property cannot be used for hexahedron; however, the metric can be used in a point relocation algorithm applied on the hexahedral meshes. Therefore, a metric that will help us to control the hexahedral mesh generation process can be found from the literature on the metric based tetrahedral meshing methods.

2.1.2 Advancing Front Triangulation

The second most popular group of tetrahedral mesh generation algorithm is the advancing front type. There is a significant amount of research on advancing front techniques, for example see [21–26]. The advancing front method consists in marching into as yet ungridded space by one element at a time. As it is seen in Figure 2.3, the triangles are inserted into the geometry by layers propagating from the boundary to the interior of the domain. The faces separating the gridded region from the ungridded part is called *the front*. The

advancing front algorithms rely on finding new triangulation based on local size functions and front intersection control. The collapsing fronts are connected to each other to ensure the mesh topology.

Advancing front algorithms contain many good properties like the local size control. The size of the cells can be imposed very easily; since triangles can be created in any size only condition of non-intersecting fronts are satisfied. This size control makes the method very popular in CFD, where boundary layer cell size control plays a crucial role.

As a side note, the advancing front marching methods can be used as point insertion algorithms in Delaunay type methods. Several authors combined two methods [27–29], which is indeed a very wise concept. Using advancing front type point insertion constraints, meshes of desired size distribution are created, while keeping the Delaunay criterion for the high quality cells.

Figure 2.3: Advancing Front Triangulation

2.1.2.1 Mesh control methodologies from Advancing Triangulation

Advancing front approach is a geometrical method. The mesh is generated from the surface to the interior by easily controllable logical criteria that are selected for the requirements of the problem that will be solved on the grid. Although they are not mathematically as well-defined as Delaunay methods, they still offer a very good mesh control mechanism. Unlike Delaunay type methods, they don't have a problem of boundary constraint. They also offer excellent mesh at the locations close to the boundary.

Hexahedral meshing still has very limited mathematical basis compared to tetrahedral meshing. This is the main struggle for researches to build a robust scheme for hexahedral meshing. For that reason, advancing front triangulation promoted researchers to develop similar hex-meshing algorithms. Therefore, the control methodologies for advancing front algorithms can be extended to hex-meshes.

The analogy is then extended to advancing front quadrilateral mesh generation, which is successful. Unfortunately, the method could not be extended to the hexahedral meshes due to the extremely complex front matching conditions. It is seen that, direct extension of advancing front methods to the hex meshes is not feasible and only some very advanced methods, which are quite different from the original method could be applied with a limited success.

2.2 Hexahedral mesh generation

It is previously stated that, hexahedral meshes are superior to tetrahedral ones, in terms of simulation accuracy, cell count and simulation speed, those creates an industrial demand on automatic hexahedral grid generators. However, building a robust and automatic hexahedral mesh generator is much more difficult than the tetrahedral case. The reason is, hexahedron is much more stiff than tetrahedron in geometrical sense, which introduces additional difficulty in making topological operation on hexahedral meshes. For example, point insertion, in other words inserting a node to the domain and updating a valid connectivity is a difficult operation in quadrilateral meshes. This operation is even not possible for hexahedra. Similarly, swapping operations is extremely difficult in hexahedra.

This stiffness also results in difficulty of seaming hexahedral fronts with valid hexahedra, which is the main obstacle for advancing front type methods. A famous example for this is demonstrated by Schneiders' problem [1]. Consider the simple pyramid shown in figure 2.4. The surface mesh is the simplest quadrilateral mesh and the hexahedral mesh that fills the volume using this surface mesh is questioned. Surprisingly, even for this simple problem, a valid solution cannot be found, yet.

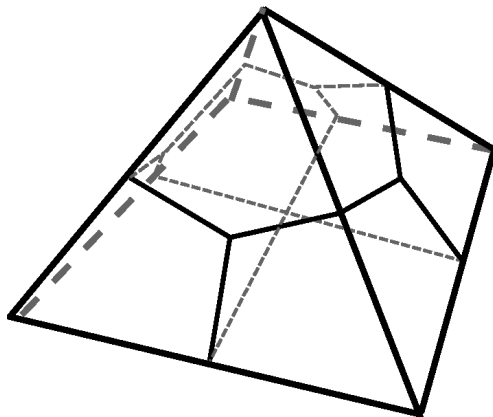


Figure 2.4: Schneiders' open problem [1]

That is the reason why the number of different hexahedral mesh generation algorithms is much higher than that of tetrahedral meshes. Considering the expectations from an ideal hex-meshing algorithm that will be given in the next section, the problem of robust hex meshing is not solved for the time being. In this section, the overview of the state of the art in hexahedral meshing is given with an emphasis on mesh control potential of the methods during the generation of the grid.

2.2.1 Ideal Hex-meshing Algorithm

As novel algorithms for hexahedral meshing emerge, it is seen that all these algorithms have their shortcomings. Properties of an ideal hex-meshing method for ultimate reliability and robustness are identified by Sabin [30] for general meshing and this definition is extended by Tautges [31] and Blacker [5] after the hex-meshing experience in the last decade. Some of these properties are obvious, such as meshing speed [6] and mesh quality. Below, the most important of those requirements are given.

Geometric Generality

The algorithm should be available for every kind of geometry and shouldn't be limited to a certain kind of geometries.

Orientation Insensitivity

The mesh for an object should be the same regardless of the orientation of the geometry in the space. In other words, if the geometry is translated or rotated, the mesh produced for these conditions should be the same as the unmodified one.

Boundary Sensitivity

The most important region of the grid for the numerical simulation is the vicinity of the boundary. The quality of the cells in this region is extremely important in order to apply an accurate simulation.

Cell size control

The optimum mesh may be defined as the grid that contains least cells to obtain most accurate results. This is achieved by introducing denser cells in the regions where more accurate are calculations needed. The control of cell size should be automatic in the mesh generator.

2.2.2 Indirect methods

Indirect methods are based on producing hexahedra with topological operations applied on the elements of tetrahedral mesh. These operations include merging triangles to tetrahedra as well as subdividing the simplexes to quadrilaterals-hexahedra as it is shown in Figure 2.5. These methods are employed as initial attempts to generate hexahedral meshes, however it is difficult to claim that those attempts addressed the advantages of the quad-hex meshes.

First, the quality expectation from hexahedral meshes have not been met, since triangle and tetrahedron have very different quality criteria and different connection conditions. It is not possible to produce a hexahedral-quadrilateral mesh with cells of high orthogonality, using these operations.

Figure 2.5: Indirect hexahedral mesh generation methods

The first method consists in subdividing the simplexes. In the first row of Figure 2.5, the subdivision operation in 2-D is shown. As it is seen here, the number of cells are increased three fold for triangular meshes, whereas this number increases to four-fold in 3-D with tetrahedral elements as it is seen in the last row. This increase is not desired and conflicts with the major expectations of reduction of cell count by employing hexahedral meshes.

The second approach, which consist of merging triangles into quadrilaterals, may produce cells of slightly better quality and reduces the cell count, however, it may not be possible to produce all-quadrilateral meshes. In most cases there are triangles left unglued in the domain, as it is seen in Figure 2.5. Moreover, 3-D extension is not obvious, since merging two tetrahedra does not produce a hexahedron. There are a couple of hexahedron subdivision schemes to produce tetrahedra, however, finding these combinations in reverse order in a real mesh is a matter of chance and is very rare.

However, one of the methods is worth mentioning here. In their very interesting two part study, Owen et. al. [32, 33] presented an indirect hex-dominant mesh generation technique. In these studies, an advancing front type tet-meshing scheme is designed to be used in conjunction with hexahedral merging. See section 2.2.3.1 for a detailed explanation of this study.

2.2.2.1 Mesh control with Indirect Methods

Although it was criticised heavily in the previous section, the method holds attention in mesh control point of view. Indirect methods potentially have all the features their base tetrahedral meshing algorithm has. This is very exciting, since the control methodologies are very well studied and very robust for tetrahedral meshing. To be able to transfer the control possibilities of tetrahedral methods to hexahedral meshes automatically is a very tempting idea.

Unfortunately, in reality, subdivision based methods do not preserve anisotropic features in hexahedral cells. Instead of stretched rectangles, one obtains sliver cells, which do not contain any positive features of hexahedral meshes. In three dimensions, merging based methods are mostly impossible to implement. Nevertheless, the hex-merging controlled advancing front tet-meshing type methods seem promising. However, the constraints on tetrahedralisation part of these methods are stricter than that of direct tet-meshing methods, since the tetrahedral mesh should allow the merging operation. In addition, all-hexahedral meshing is still not possible with this kind of algorithms.

2.2.3 Advancing front methods

When the all-hexahedron mesh generation problem began to attract academic interest, a number of researches started importing the triangulation/ tetrahedralization techniques for new generation quadrilateral/ hexahedral generator. Of course, Delaunay class of methods was eliminated immediately, since Delaunay criterion is strictly simpleton feature, and cannot be extended to any other shapes. In addition, point insertion, and many refinement operators are unusable in quads and hexes. As a result, many researchers started to implement advancing front type algorithms for hexahedra. As it is the case in tetrahedral meshes, these techniques are designed to work from geometry surface to its interior.

After the initial optimism about advancing front mesh generation implementations on quadrilateral meshes, it was realized that extension of those methods to hexahedrons is extremely difficult. The fundamental problem is the difficulty of finding the matching criteria at the meeting surface fronts due to the constraints of the conformal connection of the hexahedral elements. The topic drew a lot of attention from the academic world, and after fifteen years of continuous research, today there is a couple of very complex algorithms, which can be classified into two categories. The first one is following the classical approach and called *Paving* and *Plastering*. The second group of algorithms, mainly developed by a group at Sandia National Laboratories is called *Whisker-Weaving*, is an extremely complex algorithm, which is still evolving. Both approaches are still not mature enough for the industrial use.

2.2.3.1 Paving and Plastering

Paving is direct extension of the advancing front methods in surface meshes. In this method, a topological surface is projected towards inside of the geometry by rows of quadrilaterals for each topological edge, until the entire surface is covered with mesh. Most of the time, there are one or more degenerate faces, which then are subdivided into quadrilaterals using template subdivisions and these subdivisions are projected through cells to the edges in order to obtain conformal surface meshes. One of the first studies in this area is presented by Blacker [34]. In this study, paving in a planar surface is formulated. Researcher designed the algorithm where edges are projected row-by-row. Cass [35] generalized Blacker's method for an arbitrary 3-D surface. However, this row-by-row projection approach is not very robust, since the control of advancing front rows sometimes becomes extremely difficult, due to the complexity of matching front. To relax this hurdle, White [36] introduced cell-by-cell paving instead of row-by-row paving.

Paving is considered as a successful method on surface meshing, however its natural extension to 3-D called *plastering* is not as successful. Plastering starts with meshing the topological surface of the geometry with paving (or any other surface quad meshing algorithm). Then hexahedral cells start to fill the volume progressively by projecting from the surface mesh to interior domain. However, connecting the mesh where the advancing fronts meet is not possible every time. Most of the time, a very large unmeshed volume is

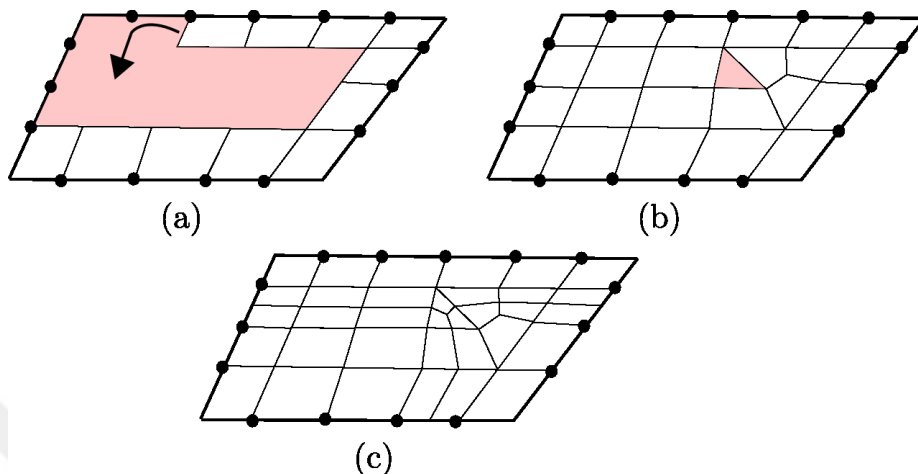


Figure 2.6: Paving algorithm as it was redesigned by White.

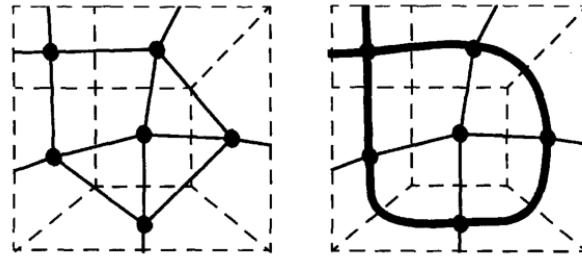
left in the middle of the geometry, which should be meshed by hybrid elements. Currently only very basic geometries can be meshed with plastering, and for real world cases the method is still not robust enough.

There are not many attempts in literature to explore plastering. As in paving, Blacker has formulated plastering [37], defining seam and wedge operations to connect meeting fronts. However, in most of the practical cases it is seen that seaming of the mesh is not possible, and a huge volume inside the geometry left unmeshed as it is described above. The complexity of the problem is discouraging and left researchers to focus on hybrid methods. Leland [38] employed a tet-hex mixed element paving algorithm, which relaxes many shortcomings of the plastering.

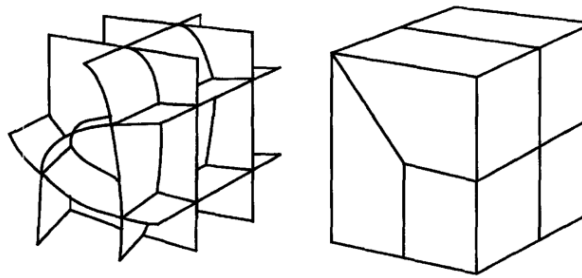
An interesting series of studies are presented by Owen. Owen implemented an indirect approach for both paving [32] and plastering [33]. In *H-Morph* technique, a background triangular mesh is followed from edge to interior cells, by creating quads analysing triangles, modifying the connectivity if necessary with basic operations like flipping and merging them into quadrilaterals. Nodes of this created quadrilateral are relocated to ensure high quality cells. This procedure is repeated until all the triangles are converted to quadrilaterals. In the *Q-Morph*, starting from the surface quadrilateral mesh, which can be created by *H-Morph*, a tetrahedral background mesh is created, and following similar, but more complex procedure with the *H-Morph* technique, hexahedral cells are created by merging tetrahedral meshes.

2.2.3.2 Whisker-Weaving

A more successful approach to surface to volume strategy for hexahedral meshing is introduced by Tautges, Benzley and Blacker [39–41]. The idea is build around the concept of *Spatial Twist Continuum* (STC) described by Murdoch in [42, 43]. The idea is to use



(a) dual and STC of surface mesh



(b) STC sheets of 3-D volume

Figure 2.7: Spatial Twist Continuum (STC) and Whisker-weaving type hexahedral mesh generation

the dual mesh of the surface mesh to construct the volume mesh. First, the dual of the surface mesh is built. To obtain the surface mesh, whisker weaving in 2-D or any other surface meshing method can be employed. An example STC of a surface mesh is shown in following figure 2.7. This STC is extended as dual surface strip on the volume. As the Whisker weaving name implies, the STC or the surface dual is used as a guide to *weave* cell edges from surface to the inside of the geometry and the mesh is constructed by following the STC. This is a wiser idea than the original algorithm, since the constraint on the matching hexahedron fronts is relaxed by analysing the dual. However, the method is extremely complex. Moreover, due to the nature of the algorithm leads to low quality cells while following the STC.

Several alternative methods are implemented by researchers. Müller-Hannemann [44, 45] uses the dual transform of the graph of the surface grid to insert layers of hexes covering the boundary. The layer is removed from the boundary, leaving the mesh with one less dual cycle.

Another study on the topic is presented by Calvo [46, 47], where researcher used the dual mesh as subdivision means. The algorithm is indeed applying the subdivision in layers.

Dual based methods employ highly sophisticated algorithms. Basic source of the complexity is the dual. Construction of the dual from sheets and wires is a difficult task. Additionally, most of the attention should be given to constructing a good dual, which

is not self-intersecting and which produces high quality hexahedrons. Methods to achieve these goals vary from surface mesh reconstruction to dual mesh cleaning [40, 45, 47, 48]

Unlike Plastering, Whisker Weaving has some degree of success. Geometries that are more complex can be meshed with the method at least in hex-dominant fashion; however, the method is still not robust enough for general use.

2.2.3.3 Mesh control with Hexahedral Advancing Front Technique

Advancing front meshing ideally offers very good mesh control features. First, advancing front algorithms are orientation insensitive, since the meshing starts directly from the topological primitives. Although in hexahedral implementations available in the literature are designed to produce isotropic meshes, there is no real obstacle for anisotropic meshing implementations. Moreover, although the volume mesh cannot always be achieved, surface mesh is generally high quality and can be reused.

However, in practice, this potential cannot be exploited up to now in three-dimensional case. The plastering type methods are not robust enough to be utilized in meshing process, yet, it is too early to implement and control feature on them.

Whisker-Weaving is much more promising than the plastering type advancing front techniques. Actually, whisker-weaving is essentially a control process. The idea is evolved in time from utilizing the dual mesh to control the Spatial Twist Continuum, or dual sheet itself. Nevertheless, on the negative side, the automatic control of the STC is an extremely complex process. Three research groups are known to be working on the subject for last ten to fifteen years; however, due to the overwhelming complexity of the method, the progress is rather slow to allow the method to become available for general geometries in near future. Therefore, this fact discourages researchers to implement any control methodologies on the whisker-weaving type algorithms.

Even if the control methodologies would be implemented for these methods, there are other obstacles that would not let these techniques to be accepted by the industry: First, due to computationally very expensive projection calculations, the algorithms are extremely slow. Therefore, only small sized meshes can be generated. Parallelisation should be relatively easy for plastering type, but it is not very clear how to parallelise the whisker-weaving type algorithms. In addition, although many improvements are made, still, only a limited set of geometries can be meshed. Last, all-hexahedral meshes are difficult or impossible to generate. Whisker-weaving produces Hex-dominant meshes, and it seems that the research proceeds in this direction. For complex cases, even hex dominance cannot be guaranteed by plastering type algorithms.

2.2.4 Geometric Decomposition Methods

2.2.4.1 Medial Axis Methods

Medial Axis methods are based on skeletonisation concept mainly used in computer image recognition. In his pioneering work, Blum [49] defined a medial axis as the locus of the centre of an inscribed disc of maximal diameter as it rolls around the domain interior expanding and contracting to maintain contact with the domain boundary. The medial axis of the 2-D shape is shown in Figure 2.8. The combination of the medial axis and the radius function, which describes the radius of the inscribed disc at any point on the medial axis, is known as the Medial Axis Transform (MAT). The Medial Axis Transform is an invaluable tool in computer graphics. The object boundary can be reconstructed using this information, since all the boundary information is embedded in the medial. Therefore, the medial axis can be considered as a dimensionally reduced model of the geometry. This process is valid also in three-dimensional objects. In 3-D the equivalent construction is the locus of the centres of all inscribed spheres of maximal diameter, which is sometimes called as medial surface.

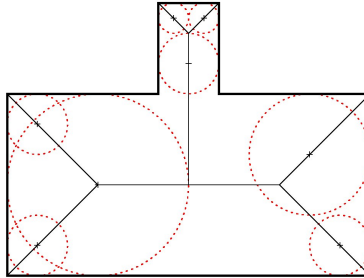


Figure 2.8: Medial Axis in two dimensions. Circles are maximal tangents to any point on the surface indicated with the bold line. The centres of the circles construct the Medial Axis.



Figure 2.9: Medial Surface of a three-dimensional object. Left is the object, right is its medial.

Although the initial study [49] was conducted quite early, use of the method in com-

puter graphics expanded much later. Application areas of the medials spread from image recognition to medical imaging also robotic motion planning and mesh generation. Because of this, the computation of the medial is a popular research topic in computer graphics. There are various of approaches to subtract medials in the literature, which are, thinning algorithms, that are based on topological thinning of the geometry up to skeleton, Voronoï based methods, analytical methods, grass-fire transform and medial axis extraction from a distance map. For a good review of the medial axis transform tools refer to the paper of Leymarie [50].

In the hexahedral mesh generation field, medial axis transform is regarded as a tool with lots of potential, simply because of its ability to give a simpler, dimensionally reduced description of the geometry, without losing any of the features of it. Extending Sampl's description, [51] medial transform is a very powerful tool because it contains:

- information on the local feature size, which is the radius function,
- information on opposite boundary points by relating the nearest boundary points of a medial axis point, i.e. two tangents of the maximal sphere,
- information on the region's topology by medial axis topology.
- information on branching of the topology

If these items are investigated carefully, it is seen that, local feature-size function (or local thickness), gives a hint about the maximum local cell-size information, which is much desired information for controlled mesh generation. Second, connection to the opposite face provides the subvolume meshing. Preservation of the topology of the geometry offers a full-detail meshing opportunity. Finally, simplified representation of the topology offers a tool for automatic decomposition.

These features, of course, are very attractive from mesh generation point of view. Then, starting from early nineties, medial axis based mesh generation methods started to appear. One of the first studies in the literature using Medial Axis transform is from Gürsoy and Patrikalakis [52–54]. In these studies, researchers introduced a two-dimensional quadrilateral mesh generator based on medial axis of planar geometries. In [52], they introduced the algorithm decomposition of the geometry, whereas [53] is dedicated to meshing applications. The algorithm can be applied on two-dimensional, planar geometries, with success.

However, one of the major problems of medial axis methods started to appear. The medial axis is very sensitive to the surface detail, yielding to an extra dense and complex medial axis, which in turn produces extra dense mesh. Armstrong [55], in his 1995 paper, introduced the idea of geometric detail suppression, to have simpler medials. The main modifications and simplifications applied on medials for mesh generation practice are defined in this paper. Some initial 3D meshing results are also presented.

The research for three-dimensional implementation of the medial axis based mesh generators started after this study. Price introduced a Medial Axis toolkit [56] for meshing and other applications. The primary concern was to provide the cleanest possible medial surface (and axis) for mesh generation. Toolkit provides means of midpoint division algorithms and medial algorithm based on an analytical skeletonisation method. This study then paved the way for a much detailed study by Armstrong, published in two parts [57, 58]. This study refers to simplification of the model for mesh generation and handling of different kinds of corners and topological features in medial axis meshing in three-dimensions.

Sampl [51] is one of the first researchers who addressed the Medial Axis quality, which is one of the main blocking factors of the medial axis hex meshing. In practice, medial axis calculation produces a cloud of medial points, which are scattered and approximate. Producing a usable medial surface is a difficult process. To solve this, Sampl introduced a Voronoi edge based skeletonisation algorithm, followed by a medial cleaning process. Sampl uses a different process to produce mesh. Instead of a subdivision process, he prefers meshing first medial edges, then medial surfaces. This medial mesh is then projected to the surfaces. The resulting mesh is Hex-dominant.

Using a more robust method for Medial Axis extraction, Sheffer [59] introduced an embedded Voronoi graph based Medial Axis transform tool. Theoretically, Voronoi diagram of a closed object is an extension of the Voronoi graph for set of surface points. Voronoi graph inside the geometry is equal to medial axis in shape. The embedded Voronoi graph (EVG) that Sheffer used has several advantages over ordinary medial axis. The EVG in three dimensions contains the *Voronoi surfaces*, which are the medial surfaces, as well as the *Voronoi edges* and *Voronoi vertices*. These elements of the EVG are directly related to the topological features of the geometry. This extra information is used to implement a better subdivision scheme.

2.2.4.2 Mesh control with the Medial Axis Methods

The medial axis is not a specific tool for mesh generation. The medials are used in almost every field in computational geometry including medical imaging, volume segmentation, surface reconstruction and others, and they are becoming more and more popular. There are several reasons for this. First, the mathematical foundation is well defined, yet very flexible. There is more than one definition of the skeletons, like grass-fire analogy, or rolling ball analogy or Voronoi graphs, etc. This makes a number of different algorithms available, but more important than that, gives the opportunity to implement the analogy best suited to the specific needs.

The second reason is that, the medial axis transform provides a very good, yet simple enough descriptor of the shape. The dimensionally reduced dual of the shape that contains information for every feature of the basic shape is much easier to deal with. Moreover, the amount of information about the geometry that is represented on the medial can be controlled for specific needs.

Despite their popularity, there are a couple of problems that researchers had faced while developing medial axis based hexahedral mesh generators. First, controlling the amount of information embedded on the medial is an open problem. Most of the times, small details affect the medial, and simplification of the medial sometimes ends up with the loss of important details. The medial is also affected very easily from the surface impurities, which give excessively branched medial. This obstacle prevents the medial based methods to be a general method for dirty geometries as well as geometries with high detail level. The second problem is, it is not always possible to get a clean medial. Generally, in practice, the medial in the beginning consists of a cloud of points in the space, from which the medial surfaces should be built. The medial surface reconstruction is not necessary for many applications, but it is a strict need for high quality mesh generation. This operation may be very complex for detailed geometries.

The last but not the least, the medial computation can be very expensive. Faster methods like background map thinning and distance map skeletonisation provide too rough medials to be used in real life meshing. On the other hand, Embedded Voronoi Graph type detailed methods require excessive computational type, far too expensive to be used in geometries with high detail.

However, one should admit that, a clean medial produces one of the highest quality meshes inside the domain. The methods are also boundary sensitive and orientation insensitive. With the increasing computational power and introduction of novel skeletonisation algorithms, medial axis methods can be next generation meshing tools.

The problems above prevented the medial based methods to be successful. However, it is difficult to ignore the potential of the medials as a grid control tool. Medials contain much information about local sizes and anisotropy on them, which are essential for mesh control. Therefore, even if the medials are not utilised as a decomposition tool, they can be utilised in controlling the mesh generation process. In this study, the medials are used extensively to extract information about the geometry, and control the mesh generation process.

2.2.4.3 Sweeping and mapped meshing

Sweeping type algorithms became one of the most popular approaches in Hexahedral meshing in recent years. Sweeping may not be considered as an unstructured approach, since it is based on mapping procedure of the structured techniques. To be able to apply the technique, two opposite faces with similar topologies connected with a *connection surface* are necessary. The first opposing surface, which is called source surface is meshed with quadrilaterals and swept through the connection surface to meet the target surface. Sweeping is considered as $2\frac{1}{2} - D$ meshing due to this semi-structured feature. Since the existence of the target, swept and source surfaces is required to apply the method, the applicability of the method is limited. However, in practice solid mechanics applications, a good percentage of the geometries can be classified as *sweepable*. Because of this, sweeping

algorithms find themselves a way into commercial mesh generators, and the research to improve their capability is expanded. Of course, a single sweepable volume is not interesting in terms of applications, since single sweep can be done using a structured mesh generator. Therefore, sweeping algorithms try to subdivide the volume into sweepable sub-volumes, while providing connections between them. For that reason, they are classified as geometric decomposition tools.

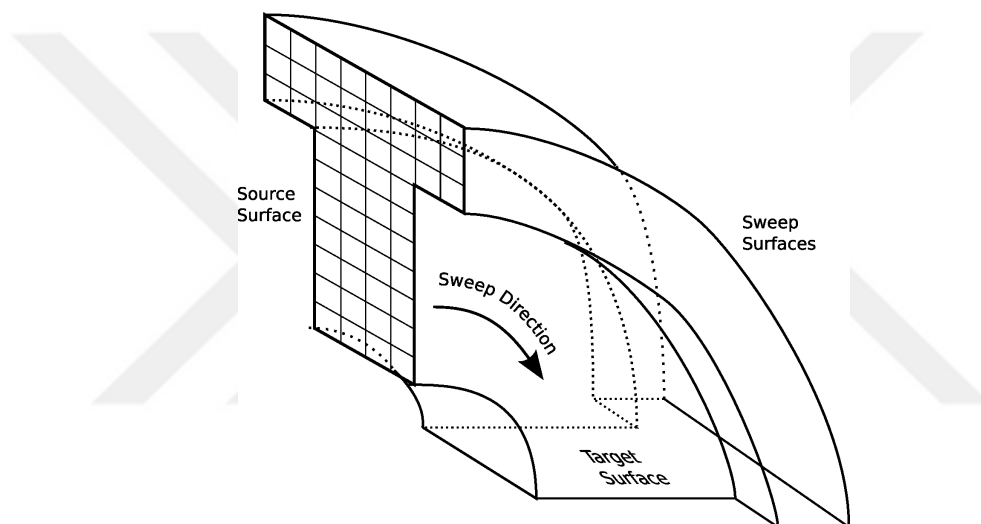


Figure 2.10: Hexahedral mesh generation by Sweeping

Blacker [60] introduced the *cooper tool* as such a decomposition scheme. A “cooper” is an artisan who makes barrels. *Cooper tool* is applicable to geometries that can be considered as “barrels” laying the same axis. A barrel is defined as a volume that is swappable from the base in any shape through an axis, which is not necessarily linear. Blackers algorithm allows meshing in multiple barrels or sweep surfaces, however, sweeping can be done in single sweep direction. In other words, an object consisting sweepable volumes that cannot be matched in the same axis cannot be meshed. The topology is allowed to branch or split along the sweep direction.

The main drawback of sweeping algorithms is that, although they allow a quite good tool partitioning and connecting the swept volumes, the sweep operation can be done in one direction only. In other words, any extrusion connecting to the swept volume cannot be meshed. This constraint limits the number of solids that can be meshed since they automatically exclude solids with imprints or protrusions on the linking surfaces. Miyoshi and Blacker [61] extended the *cooper* algorithm to be able to sweep through multiple axes. In their Multi-axis cooper algorithm, they first create a hierarchy of the branches after

classifying the source and swept faces. Then, the connection between target face patches on the swept surface is tried to be constructed. The geometry that is meshable with cooper algorithm extends to any arbitrarily branching object with this method.

Sweeping algorithms provide structured mesh in the sweep direction and they require the linking surfaces connecting source surface to the target to be mappable. For a multi-axis toll, the connection between the linking surface mesh to the mesh on a footprint of extrusion volume should be made available. To solve this problem, Jankovich [62] introduced a new tool that can connect mapped linking surfaces to any imprints of the branches, which constitutes a target for a source surface on the connected linking surface. The tool creates conforming connections and extends the usability of the sweeping algorithms.

Another study on sweeping operations is presented by Mingwu [63], which is then extended in [64]. Unlike Blacker, they choose a topological method to apply sweeping, introducing the idea of swept surface matching. Once the swept surfaces are projected onto each other and matched, overlapping surfaces are detected and meshed; therefore, sweeping becomes an easier operation. This method relaxes some shortcomings of the cooper algorithm.

The major advantage of the sweeping type methods is that; once the swept surfaces are identified and meshed, the interior mesh connectivity is also determined. However, placement of these interior nodes plays an important role. For complex geometries, there are no very well defined coordinates of inner nodes. For these kind of geometries, point placement algorithm plays an important role. Knupp [65] offers a good point placement strategy for interior nodes of swept volumes using a linear transformation obtained from the loop data. Staten [66] also points out this issue, proposing an algorithm called BMSweep, depending on a background mesh. The interpolation information is calculated on this mesh and positions of the interior nodes are calculated by interpolation operation on the background mesh data. Then, interior elements are calculated layer by layer from source to target surfaces.

This structured behaviour of the linking surfaces creates another bottleneck for the method. In certain geometries, structured connection can create large difference in cell sizes of the opposing curves of these surfaces. This creates a quality problem, and to improve the overall quality of the mesh, Borden [67] introduced some pillow cells a in the linking surface with his *Cleave and Fill* tool. This tool uses the pillowing idea introduced in [68].

As the last point, selection of the sweepable surfaces is a manual process and it should be automated for industrial use. White and Tautges have addressed this issue [69] and introduced an automatic toolkit for detection of extruded or swappable volumes. Their algorithm is based on topological and local geometric criteria.

2.2.4.4 Automatic Decomposition Methods

The ultimate goal of automatic hexahedral meshing is to be able to mimic multi-block structured meshing without human intervention. The most direct route to achieve this goal is to repeat the operations of an engineer meshing a geometry. An expert in the area first subdivides the object into meshable pieces. This requires correct identification of the features. Imitating the human conception of the geometric feature by computers is called feature recognition. Feature recognition is the process of extracting design or manufacturing information from a solid model. Extensive research has been performed in the field [70]. The basic idea behind the automatic decomposition tools is to detect the features of an object and to utilise this information in any meshing.

Sweeping is a tool to give a degree of automation, however, in practice, geometries are very complex to be meshed with one sweep operations, hence, the volume should be decomposed into mappable subvolumes. Manual decomposition of geometry into swappable subvolumes is a very time consuming process. To solve this problem, White and Mingwu [71] came up with the idea of automatic decomposition of sweep volumes introducing two methods, namely volume submapping and n-surface sweeping. Volume submapping uses pseudo or virtual geometry to decompose complex volumes into mappable sub-volumes. Mappable regions are generally limited to volumes that can be parameterised into logical hexahedra. To decompose volumes into mappable subvolumes, the mesh connectivity of the surfaces bounding the solid is used to parameterise the surface nodes in a local integer i - j - k space.

Shih and Sakurai [72] further experimented on the idea of using sweeping technique as a volume decomposition tool. They used linear but non-rotational swept volumes. Their model is limited to linearly swappable models, and cannot be extended to general geometries.

Liu and Gadh [73] proposed an algorithm using the notion of *Basic LOGical Bulk shapes BLOBs*. A *BLOB* is defined as a simple object of solid model decomposition and generally is a result of absence or presence of a material on a surface. This information is extracted by following the C-Loops, which are sets of loops that follow topological edges. Depending on the features of these edges, protrusions and extrusions on a surface are determined investigating the corner surface relationship, and the shape is decomposed using this information. The initial algorithm was limited with certain geometries and the same researchers presented an enhanced algorithm in [74, 75]. The improved algorithm addresses the possibility of accomplishing different decomposition alternatives for a complex geometry. The important point is to make the decomposition in a *meshable* manner.

This idea is further used in the study of Lu [76]. Lu employed similar tools with Liu [75], using feature recognition technique called CLoop, which is defined as a closed set of linked edges, the geometry is decomposed in simpler pieces. Lu employed subdivision technique in recursive manner, so the geometry is subdivided until there is no decomposable volume left. The classification of geometric primitives that are considered as the features of the object is extended compared with the previous studies.

Another attempt using the feature recognition is presented by Chiba et. al. [77, 78]. In their method, the geometry is decomposed into single-connected sub-models in such a way that, decomposed geometries can be meshed with a mapped Cartesian mesh block. These decomposed geometries are then converted to simple cubical blocks called recognition geometries and converted to mapping model, which consist of simple Cartesian meshes on those blocks. The mapped models are matched and reassembled in recognition geometries. The last step is to transform the mapped mesh to the original geometry with the reverse transformation applied to obtain recognition geometries.

A similar approach is presented recently by Su and Kumar [79]. In this study, geometry is tried to be decomposed into block Cartesian mappable subvolumes using fuzzy logic techniques. Instead of following the edge directions, the surface normal vectors are analysed to calculate the orientation of the sub-entries. The fuzzy-logic method is also used to calculate the reverse orientation of the geometric entities.

2.2.4.5 Mesh Control with Geometric Decomposition Methods

The sweeping type mesh generation, is a semi-structured method, hence partially shares the control features of the structured mesh generation methods. In the swept direction, the grid is essentially a structured one, and in this direction the features of the mesh are easily controlled similar to the structured methods. The swept surface can also be controlled, by controlling the surface meshing method. However, this is valid for single axis sweeping. In the multi-axis sweeping, stitch operations make the mesh control extremely difficult. Meshing is based on stitching of the branches; therefore, mesh cells should be located such that the branches are connectable. Moreover, the sweeping is only applicable for certain types of algorithms and cannot be generalised.

Automatic decomposition methods are very promising for the controlled mesh generation, since they present all information about the geometry in simpler pieces. Nowadays, feature recognition and volume segmentation concepts are very popular topics in computational geometry, and in the near future, they will be utilised in a very wide area of applications. However, currently time they are not mature enough to be used in mesh generation. Current automatic decomposition based methods can segment only very simple volumes into meshable subvolumes. As expertise on the automatic segmentation area increases, these methods would be serious alternatives to the current meshing algorithms.

2.2.5 Overlay Grid Methods

One of the few industrially accepted methods is the overlay grid technique, in which the grid is adapted to the geometry with local adaptations. This method relies on an initial grid that encompasses the geometry. This grid is refined locally at the locations where the mesh intersects the boundary. In other words, the intersecting cells are subdivided in smaller cells in order to capture finer details. After sufficient cycles of refinements,

the background grid is intersected with the geometry again, and the inside cells are kept. However, the mesh generated with this approach does not fit the geometry, instead, it is a staircase mesh that intersects the geometry. This reduces the quality of the solution at the boundary, which is the most important region to ensure the accuracy of the solution.

The unique part of these algorithms compared to simpler octree based algorithms is that, they allow the projection of the surface cells to the boundary, with the introduction of buffer cells. Therefore, a body fitting mesh, where all the surfaces, edges and the corners of the geometry are captured is obtained with very high quality cells at the boundary [80, 81]. The meshing process is summarised in the Figure 2.11. Consider the spherical body shown at (a). An initial mesh that encompasses the body is inserted (b), then the mesh is refined and the cells left outside the domain are removed (c). The boundary is captured with the introduction of new cells filling the gap between refined mesh and the geometry, so that body conforming mesh is obtained (d), and last, cells at the boundary of the domain are refined into the normal direction of the surface to have the boundary layer mesh (e).

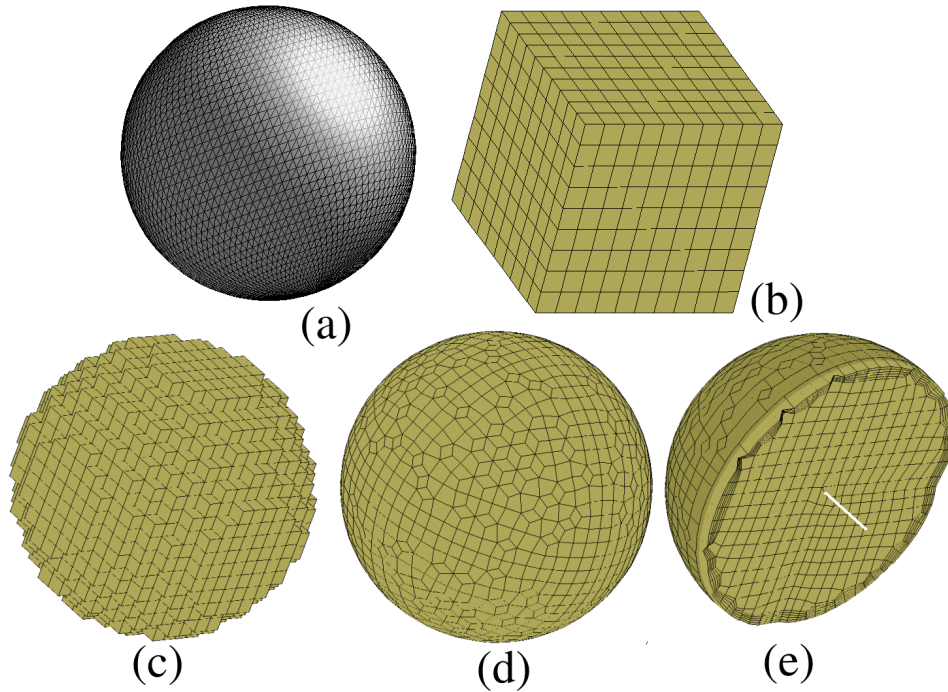


Figure 2.11: Mesh generation steps in overlay grid technique

One of the pioneering studies is conducted by Schneiders [81, 82]. In these studies, the researcher introduced the basis of the method including the boundary fitting, which makes the method the most industrially accepted one. Schneiders' method relies on the

non-conformal refinement operations, followed by trimming operation, which eliminates intersecting and outside cells.

Another such study, which perfects the original algorithm comes from Delanaye [80]. This study together with Tchou's study [83] prepared the basis of industrial mesh generation software *HEXPRESS*, which is one of the most robust all-automatic hexahedral mesh generators in industry. This algorithm is similar to the Schneiders' with a greater emphasis on efficiency on local refinement operations and the robustness on topology capturing. The mesh generator also focuses on the boundary layer insertion and mesh quality improvements, which are generally regarded as two pitfalls of the method.

One of the biggest criticisms on the method is the weakness of the method to require too many refinement operations in order to capture thin sections of the geometry. These refinement operations increase the cell count significantly. To solve this problem, in his sculpturing algorithm, Walton [84] suggested not eliminating the cells that cut the geometry, and apply the snapping operation from outside of the geometry to the inside. This is a good idea to solve the problem, since, there are always cells left in the geometry unlike the methods that use inside to outside snapping, where void sections may appear due to the excessive trimming of the cells. However, since the cells are squeezed too much inside the geometry, the boundary cells can be of lesser quality.

Most of the overlay grid methods rely on the non-conformal refinements. Therefore, many numerical simulation softwares that rely on conformal meshes cannot utilise these meshes. There are studies that apply conformisation by subdividing the connecting cells into the tetrahedron pyramids etc [85]. However, solvers that can make use of the mixed elements are also not so widespread.

The solution for this problem is to apply conformal full-hexahedral refinement operations. Schneiders introduced the directional refinement method [82], where conformal refinements are produced by pillowing layers in alternating i, j and k directions. The drawback of this method is that it requires a Cartesian initial octree. This study is extended to arbitrary initial meshes by Tchou [86, 87]. In this study, anisotropic refinement operations are followed by the multi-dimensional pillowing step, which is accepted to be very robust. Another unique feature of Tchou's study is its use of Riemannian metric as point relocation mechanism to apply better refinement operations. Tchou's refinement operations are extended for arbitrary mesh cells, faces and elements by Harris [88] using the twist planes as a guide.

Flexibility of the method makes it possible to develop applications even with hexahedral remeshing, which is considered as a quite difficult subject. In a recent work conducted by Kwak [89], where the researchers applied mesh generation and control for metal forming analysis with overlay grid technique using conformal refinements. In this study, the domain is continuously remeshed as the metal forming develops.

Another interesting application is presented by Su [90] with Kumar. The same research group has many studies in automatic decomposition based mesh generation methods. Ap-

parently, in this new study, the grid-based analogy is used as a basis of geometric decomposition, as well as multi-domain meshing with conforming interfaces.

2.2.5.1 Mesh control with Overlay Grid Technique

Albeit it is a very efficient and automated method, this algorithm tends to produce many cells in order to capture finer details. Although, the industrial trend is to perform the simulations on denser meshes, it is wiser to keep the cell count to a minimum and allow the user to increase the cell density only in the regions required by the simulation. Therefore, reduction of the number of cells produced by the mesh generator is becoming a requirement, as more complex simulations are made on extremely complex geometries.

The major inefficiency is the orientation insensitivity of the method. The final mesh obtained by the geometry is dependent on the initial grid that encapsulates the geometry. An initial grid that follows the geometry well will result in a very high quality mesh. However, in practical application this is not the case. To ensure user independency, initial grids are generally Cartesian box-meshes that are inserted automatically. The user-independent nature of the technique is the most appreciated side of the method.

So that, automatic mesh control on these methods is not an option, but a must in terms of reliability and robustness. The greatest advantage of the method is its unsurpassed flexibility to deal with any kind of meshes. The method is independent on the cell size, cell orientation and cell isotropy. Therefore, any control mechanism that can be applied on hexahedral meshes can be used in this method.

A recent study proves the applicability of the mesh control techniques on the overlay grid based mesh generators. In their paper, Tchon et. al. [87] have also introduced the control of the mesh with a metric field defined around the geometry. Cell sizes and directions are controlled with the help of this metric.

Apparently, due to its ability to handle arbitrary meshes, the overlay grid technique offers the greatest opportunity of the control of the grid in the meshing process.

2.3 Mesh Control Techniques

Computational grids are as important as the numerical scheme itself to achieve a reliable and efficient simulation of physical problems. The requirements on the mesh generation schemes vary with physical problem and numerical scheme. For example, mesh size requirements are completely different for internal and external flows. Unlike inviscid flows, viscous flow simulations require very thin cells close to the boundary. In addition to these meshing-stage requirements, handling of the meshes during the simulation-stage is often necessary. The examples are adaptive meshing techniques that are applied to increase the accuracy of the solution at necessary locations and handling of mesh-movement.

Therefore, for a robust numerical simulation, computational grid topology should be controlled at mesh generation stage and/or at simulation time. The main demand is on the cell-size control and assured mesh-quality. For this purpose, smoothing and optimisation based methods are employed, which are used to enhance the mesh locally to fit the requirements of the numerical schemes, such as cell edge orthogonality and aspect ratio. The other category of methods are mesh relocation methods, which are used to keep control of the mesh movement or apply the adaptation by reorganising the node positions with adaptation criteria.

In the following section, such algorithms are explained. Mesh improvement methods are categorised in three groups, however, most of the time the distinction between them become less clear. Many studies involve hybrid applications of these methods. Moreover, with an appropriate modification, all methods can be used for almost every purpose with certain efficiency. For example, it is possible to use a point relocation algorithm in mesh optimisation and vice versa. It should be noted that, point relocation algorithms are designed to be more flexible than the quality enhancement methods, which are aimed of obtaining highest mesh quality.

2.3.1 Laplacian Smoothing

Laplacian smoothing is one of the oldest optimisation methods for grid generation. The method is originally developed for structured grids. Researchers made a visual analogy between Cartesian meshes and potential flow solutions, observing the orthogonal and well-distributed iso-lines of the stream function and velocity potential. Potential flow is based on the famous Laplace equation given as $\nabla^2\psi = 0$, $\nabla^2\phi = 0$, where ψ is the stream function and ϕ is the velocity potential. Analogous to this, an orthogonal and well-distributed curvilinear grid can be defined in the domain solving,

$$\nabla^2\xi = 0, \quad \nabla^2\eta = 0 \quad (2.1)$$

where, ξ and η are constant *potential* lines for upper/lower and left/right boundaries in body fitted Cartesian meshes. ξ and η can also be considered as the primary directions of computational domain, which is a unit square. Therefore, Laplace equation can also be considered as a space transformation tool and reverse equation, that solves the coordinates of x and y in $\xi - \eta$ plane for uniformly distributed grid lines.

Laplace equation by nature produces the smoothest possible meshes, as it can be shown from calculus of variations. For its most basic form, Laplace equation tends to give the average points at the neighbouring points. However, this distribution brings two problems. First, although it produces very good cell size distribution in convex regions, point-averaging behaviour tends to produce very low quality, even negative cells in concave regions. Therefore, extra care is necessary in controlling these regions. Second, the cell size distribution is uniform, and in order to concentrate coordinate lines in certain regions,

basic equations should be modified. To control the Laplacian smoothing, source functions are introduced in the right hand side of the equation (2.1), and the new equation is called Poisson equation. The literature on these control functions is enormous and a good overview of the methods for orthogonality and the size control can be found in [91, 92].

Laplacian smoothing is not limited to structured Cartesian grids. Especially with the newly developed size control techniques, which reduce the difficulty of algebraic nature of control functions, the method is also applied to triangular grids by many researchers such as Jones [93], Field [94] and Hansbo [95].

Due to their mathematical properties, Laplacian smoothing techniques are a good basis for optimisation techniques, which are described in Section 2.3.3. Laplacian system is generally computationally less costly than many optimisation schemes. Such algorithms are presented by [96] and Canann [97, 98].

Laplacian systems provide the mathematically most optimal meshes, however, controlling the mesh in extreme conditions, such as large mesh movements, adaptive meshing can be cumbersome.

2.3.2 Physics Based Methods

Physics based method relies on analogies with continuous or discrete pseudo-physical systems and meshes. For the continuous case, the most obvious example is the analogy between moving boundary meshes and fictitious deforming solid system. For example, for a moving boundary system, with a displacement field of u on a given mesh boundary $\partial\Omega_b$ of the cells, the mesh is optimally relocated by solving the mechanical system,

$$\nabla \cdot (\mathbf{E}_{ijkl}\epsilon_{kl}(u)) = 0 \quad (2.2)$$

where, ϵ is the strain tensor and $\mathbf{E}$ is the elasticity tensor.

This system is, however, very expensive to solve. This is a severe problem especially for Computational Fluid Dynamics (CFD) problems, where the cell-count is very high. Therefore, the pseudo-mechanical analogy is largely simplified by introducing a discrete system instead of continuous one by Batina [99, 100]. In the new set-up, which is named as spring analogy, the mesh edges are replaced by a network of springs with certain stiffness k as it is shown in Figure 2.12. The springs applies forces F on nodes determined by Hooke's Law and under these loads, nodes move to the equilibrium points given as,

$$\sum_{I=1}^n \vec{F}_I(x_J) = \sum_{I=1}^n (x_I - x_J)k_I = 0 \quad (2.3)$$

This system is much simpler than the continuous system given in equation 2.2, since only the solution of a linear system is necessary. In addition, spring analogy offers a robust

method especially for mesh deformation problems. Because of these, the spring analogy is one of the most popular methods in mesh control. The applications include mesh moving meshes [99, 101], store separation problems [102], fluid-structure interaction [103], free surface flows [104] and mesh adaptation and optimisation [105].

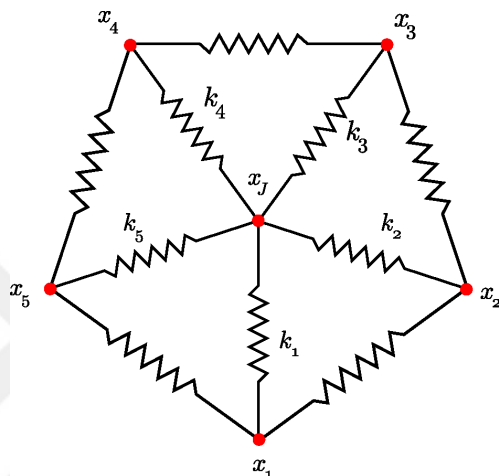


Figure 2.12: Lineal Spring Analogy

However, the method is not failure-proof for very large deformation and complex geometric configurations as it is demonstrated by Chakravarthy[106]. There is the guarantee that the points will be located far enough; however, the node can cross over the edges for very dense meshes. To solve this problem, Farhat et. al. [107, 108] proposed a new method that is named torsional springs, where fictitious torsional springs are located at mesh nodes as it is shown in Figure 2.13 for one cell. Here, torsional springs do not allow too small, and negative angles between mesh edges, hence provides the mesh quality.

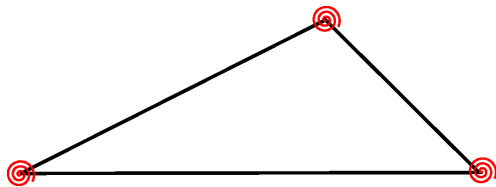


Figure 2.13: Torsional Springs Analogy

The spring analogy is preferred over the other methods, not only for its simplicity, but also its flexibility. The user is free to choose two parameters depending on the problem. The first one is the spring stiffness. With the correct selection of the stiffness of the edge-springs, cell-size control and anisotropy control is applied to the mesh easily. For example, for a

uniform distribution, stiffness is set to reverse of the edge length. The second parameter is the base length, where the spring is in neither compression nor extension. The base length is a very good tool to control mesh quality.

2.3.3 Optimisation Based methods

Optimisation methods are based on solving quality metrics through mesh cells and nodes. Many optimisation methods may also contain ideas and procedures from smoothing and physics-based methods. These methods implement optimisation techniques based on a quality criteria around each cell. The difference between optimisation methods is the choice of the quality criterion. Extensive research about these metrics is available in the literature.

The basic optimisation metrics are solid-body deformation methods [109], which are based on optimisation of deformation energy necessary to provide the best-shaped elements, minimisation of aspect ratio [110] and maximising the orthogonality [111]. These methods are mostly tetrahedral based, however, hexahedral extensions of them are possible. Canann [97] has presented a method for hexahedral elements, which depends on tetrahedral quality metrics.

Optimisation based methods provide very high quality meshes. However, these methods are so expensive that it is almost impossible to use them as a general tool. Many researchers have addressed to this issue and suggested simpler quality metrics such as Laplacian metric [97, 98]. Therefore, these methods should be applied as a final step in mesh generation process, in order to ensure highest quality mesh, and mesh control process should be handled with a more efficient method.

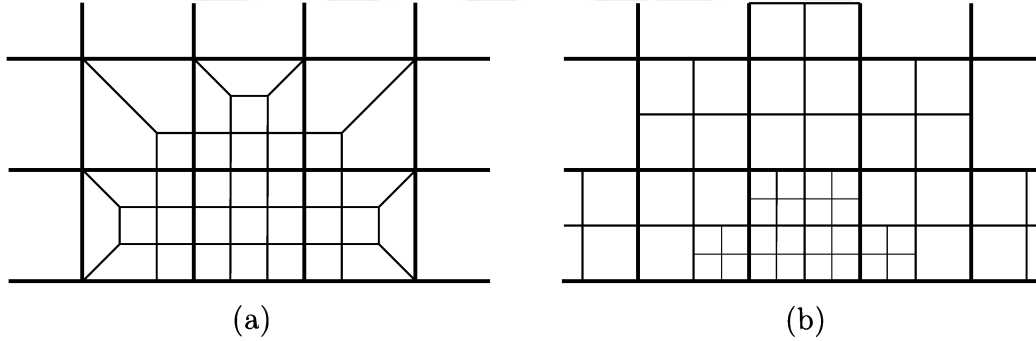
2.3.4 Mesh Refinement

Another procedure to control mesh generation and adaptation is the refinement of the mesh. In tetrahedral meshes, the main mesh refinement technique is the point insertion. In this method, new nodes are inserted at locations where finer mesh cells are necessary. Then, these nodes are connected introducing new mesh-edges to the closest points. Then, using the techniques described in Delaunay triangulation section, mesh quality is improved. Unfortunately, point insertion techniques are not appropriate for hexahedral-quadrilateral mesh generation. Point insertion is possible to a certain degree in quadrilateral grids but resulting mesh is very low quality and often it is difficult to improve this quality, since many topological operations like edge-swapping is not applicable to quadrilateral meshes. Point insertion in hexahedral case is even not possible.

Apparently, the only method that can be applied to refine hexahedral meshes are cell sub-division type refinements, where cells are cut into smaller hexahedron. There are two main types of sub-division schemes, namely conformal and non-conformal refinements as

it is seen in Figure 2.3.4. In this figure, the boundary cell in the middle is refined for both cases. For the conformal case, the refinement operation is seen in first sub-figure. Here, the cell to be refined is subdivided into nine children, (27 children for the hexahedral case) and surrounding cells are subdivided with conforming templates. These templates differ depending on type of neighbourhood¹ and refinement distribution. This kind of refinement provides unique connectivity.

Alternatively, for the non-conformal case the target cell is divided into four octants (or 8 for hexahedral case). There is a significant gain in the number of the cells in this case, however, hanging nodes at the edge and face centres are introduced. This yields a multiple connection between cells with sudden change of the cell size. To maintain the accuracy of the numerical scheme, this refinement should *diffuse* after a second one is applied in order to prevent too large size change containing non-conformal connections. In Figure 2.3.4.(b) two refinement operations and refinement diffusion is shown. Therefore, the method becomes less efficient and produces too many cells.



caption[Conformal and non-conformal refinement operations for quadrilaterals]Conformal (a) and non-conformal (b) refinement operations for quadrilaterals

Application of the non-conformal refinement is considered to be easier than the conformal one. Moreover, the final mesh consists of perfect shaped hexahedron, whereas conformal refinement introduces skew cells in the connecting primitives; and for a higher quality conformal mesh, an optimisation step should be applied after the refinement process. On the contrary, non-conformal refinement introduces hanging nodes, which are not usable in many numerical simulation codes.

Refinement operations are the primary tools used in overlay grid methods and many adaptive solvers. Both refinement methods produce enormous number of cells, which creates many difficulties in terms of computer resources. Habashi [105] suggests anisotropic refinement operation coupled with a mesh relocation algorithm for the adaptive methods. Similarly, Tchon [86] applied a similar methodology for an overlay-grid based method.

¹i.e. depending on node, edge or face neighbourhood

2.4 Conclusion

In the previous sections, the state of the art of Hexahedral Grid generation together with grid control methodologies in general is presented. The grid control technology for tetrahedral grid is very mature and very robust, and can be used as a starting point for developing hexahedral grid control methodologies. Unfortunately, due to different characteristics of the two different meshes types, what we can directly use from the tetrahedral mesh generation experience is limited. However, some of the ideas developed to help for controlling tetrahedral mesh generation can be utilised in hexahedral meshing process. One of such concepts that can be utilised in the hexahedral mesh generation is the metric fields. A metric is a local transformation from the Eulerian space to the Riemannian space, which can be utilised for control of the local features of the mesh such as cell size and anisotropy.

The metric field can be applied to control hexahedral mesh generation process using a suitable mesh relocation algorithm. The mesh relocation algorithm to be used in this study is chosen as the spring analogy, due to its flexibility and simplicity. The flexibility of the method lets us develop spring analogy based system using Riemannian metrics. Due to the simplicity of the spring analogy, a minimum effort is needed to apply new metrics for mesh relocation. This makes development of new methods based on different metrics possible by letting developer only on the metric, not the mesh relocation system. The detailed discussion of the spring analogy based algorithms is given in the next chapter.

The selection of an appropriate metric becomes the most important part of the mesh generation control process. The metric should be chosen concerning either requirements of the numerical simulation or to improve the inefficient parts of the mesh generation algorithm. With the intention of the mesh generation control, the latter, improving the inefficiencies of the mesh generation, is chosen as our target. Therefore, in order to proceed towards this direction; the mesh generation algorithm should be identified.

After a decade of research, there are a handful of hexahedral mesh generation techniques in the literature, some of which are already being used by industry. Unfortunately, many hexahedral grid generation technologies are either not robust or their application areas are limited to certain classes of geometries. Others need significant amount of human intervention to mesh the domain. Among many methods, the grid-based methods appear to be the most flexible and robust one. The method is proven useful for meshing for every field of application. Indeed, the generality of the overlay grid based algorithm is unsurpassed by any other method. Unlike most of the hexahedral meshing algorithms, there are no known shortcomings of the method; therefore, it is one of the few all-hexahedral meshing methods. Moreover, the algorithm needs almost no human intervention, what makes the method one of the best candidates for full-automatic mesh generators. Therefore, overlay grid based methods can be a very good foundation for this study.

We are grateful to Numeca Software International, who generously opens the source code of their excellent software for us to implement our ideas. Their unstructured mesh generation software, *HEXPRESS*, is an overlay grid based mesh generator that relies on

non-conformal refinements. The software provided an excellent development environment, allowing us to focus on the control mechanisms for hexahedral mesh generations. The software offers an excellent automation on the mesh generation process, which is a perfect as *click-and-mesh* system. Many industrial customers rely on this mesh generator; therefore, there is a good feedback about the shortcomings of the method. Consequently, HEXPRESS is an excellent basis for development of control methodologies on hexahedral mesh generation, and made this study become available.

As a result, a grid-based system is chosen as the mesh generation system and the control methodologies on mesh generation process are chosen accordingly. Our method should pinpoint the inefficiencies of the algorithm and solve them using automatic mesh control. The major inefficiency of the grid based mesh generation systems is their orientation insensitivity. The whole mesh generation system depends on the initial grid that encompasses geometry. As it is discussed in respective section, the selection of the initial mesh greatly affects the quality of the mesh and the number of the cells in the final mesh. Especially the cell count in the final mesh produced by the overlay grid based algorithm should be controlled.

The problem can briefly be described as follows: since grid based methods adapts the initial mesh for the only by successive refinements of the mesh cells, so many iterations are necessary to capture the finer details of the geometry. Otherwise, after trimming out the intersecting and exterior cells, there exist no cells inside these details and capturing the topology of the geometry fails. Anisotropic refinement seems like a solution at the first glance; nevertheless, if the detailed section does not share the same orientation with the initial mesh, isotropic refinements become necessary.

This inefficiency can be solved by adapting the mesh to respect the topology of the geometry. In other words, the mesh generation algorithm should be aware of the features of the geometry, in such a way that the size and the orientation of the local features should be known everywhere in the geometry. Then this information can be utilised to control mesh generation process to build a more efficient algorithm that respects those local features of the geometry. In other words, a blind, orientation insensitive meshing algorithm becomes more *geometry-aware*.

Consequently, the problem can be solved by controlling the mesh generation process using a metric that represents the local features of the geometry, like the orientation and the thickness of the subdomain. The control mechanism, then, will consist of a mesh relocation algorithm, apparently the spring analogy that uses this metric information as an input to modify the mesh dynamically in the meshing process, such that the initial and intermediate meshes respect the local size and orientation. In the following chapters, such metrics are presented.

Nonconformity is another point to challenge. HEXPRESS relies on the non-conformal refinements for the adaptation process while capturing the details of the geometry. Non-conformal meshes are considered as more difficult to apply control methodologies compared to the conformal ones, due to the existence of the hanging nodes. Therefore, an extra care

is necessary to handle such nodes.

To summarise, in the following three chapters several Riemannian metrics that aim to solve these inefficiencies of the overlay grid based algorithms will be introduced together with spring-analogy based point relocation algorithm for non-conformal meshes, developed as the main mesh control tool to apply these metrics.



Chapter 3

Mesh Control Algorithm

Control methodologies for computational grids, in broad terms, can be described as any operation that will affect the quality of the mesh in terms of simulation requirements. This includes geometry pre-processing for better meshing such as topological cleaning or feature detection, post-processing, or run time processing such as mesh adaptation or mesh movement and direct control of mesh generation. An efficient mesh control methodology should contain at least one of these steps. In this thesis, the control of mesh generation process is focused.

Thanks to the extensive research on the mesh generation control methodologies mainly conducted on tetrahedral and Cartesian meshes, there are a number of reliable techniques for mesh control. These methods rely on topological modification of the mesh, optimisation of the cell shapes, or relocation of the nodes. Topological alteration of the mesh consists in edge/face swapping and conformal or non-conformal refinements. Unfortunately, the application of the former for hexahedral meshes is very limited [112] and is not usable for the time being. The non-conformal refinement operations are the basis of the octree overlay-grid type mesh generation algorithms, which is the basis algorithm of *HEXPRESS*, the software which Numeca International kindly permit the researcher to apply the ideas mentioned in this thesis. The refinement operations are quite efficient in *HEXPRESS* [113]; therefore, one can apply more advanced techniques for this purpose.

The mesh relocation algorithms are used for changing the position of the mesh nodes, while preserving the validity and topology of the mesh. These systems are designed such as, with the forced movement of the mesh nodes, the rest of the nodes are relocated to the most optimal positions. Among a couple of these algorithms, the spring analogy is chosen due to its simplicity, robustness and flexibility in adaptation to any meshing problem.

One of the main advantages of the spring analogy, which is also worth mentioning, is that it is not only utilised in the controlled mesh generation step, but it is also very popular in mesh post-processing, especially in moving mesh problems. Indeed, the use of the method in meshing stage is less frequent. The spring analogy controlled hexahedral mesh generation is implemented for conformal meshes [86, 87]. The application of the

method in non-conformal mesh generation is not popular, but as it will be presented in this chapter, it is very easy to adapt the method for such meshes.

Therefore, this chapter is dedicated to the spring analogy, the method which will be the foundation of mesh control techniques developed through this study. In the following two chapters, mesh control tools, which are applied using spring-analogy based mesh relocation algorithm, will be presented.

3.1 Spring Analogy

The spring analogy consists of replacing the mesh edges with a network of springs. The idea is introduced for triangular grids by Batina [99, 100] for two-dimensional moving meshes. The use of the method has been expanded to other fields of computational mechanics from free surface problems [104] to store separation analysis [114] and moving boundary problems, wherever handling of the mesh movement is necessary. The method is also used successfully in mesh adaptation problems, where the mesh is relocated such that mesh cells are aligned with the solution field and mesh anisotropy that respects the rate of change of the solution field is introduced. For the applications of the adaptive techniques on structured and unstructured meshes using spring analogy, papers of Habashi et. al. [115–117] are comprehensive references.

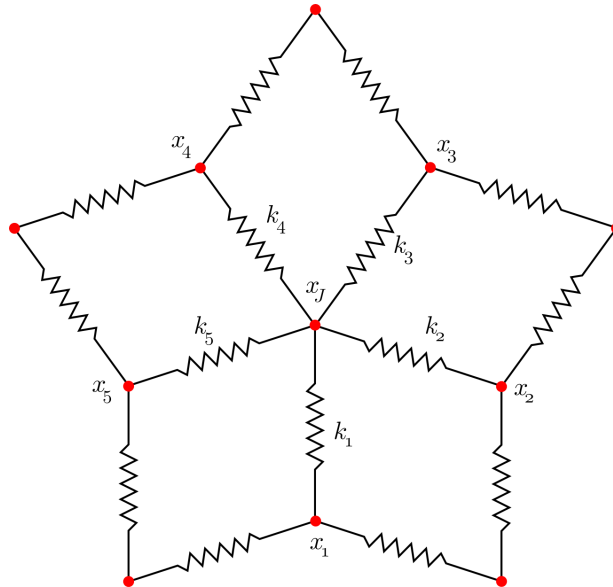


Figure 3.1: The Spring Analogy

For a detailed analysis of the spring analogy, one can refer to papers of Blom [118] for mesh movement applications and Habashi [105] for adaptative meshing. Consider the mesh

edges as a network of springs. These springs are linear, hence, Hooke's law is valid and they have a certain stiffness of $k_{I,J}$. I and J denote two nodes of the edge that is replaced by the spring as it is seen in Figure 3.1. The ideal positions of the vertices are then interpreted by the solution of an energy minimisation problem. This yields the position of node x_J , while all other nodes, $I = 1 \dots N$ connected to this node being fixed,

$$\min_{x_J \in \mathcal{R}^d} E(x_J) = \min_{x_J \in \mathcal{R}^d} \sum_{I=1}^n (x_I - x_J)^2 k_I(x_J) \quad (3.1)$$

The vertex will move to the position, where the spring energy on the node x_J is minimised. This position can be found by taking the gradient of equation (3.1), which means finding a point x_J that satisfies the equation,

$$\sum_{I=1}^n \vec{F}_I(x_J) = \sum_{I=1}^n (x_I - x_J) k_I(x_J) = 0 \quad (3.2)$$

where, by Hooke's law, $(x_I - x_J) k_I(x_J)$ is the spring force $F_I(x_J)$ related to the edge $\overline{x_I x_J}$. After regrouping the equation (3.2),

$$x_J = \frac{\sum_{I=1}^n x_I k_I(x_J)}{\sum_{I=1}^n k_I(x_J)} \quad (3.3)$$

Therefore, the new position x_J as calculated in equation (3.3) is the equilibrium location, to where the original point is moved under the forces of surrounding springs. Considering this system of springs replacing cell edges and connected at mesh vertices, this equation can be written for each node, which is free to move. This implies the solution of the system of the n_f number of equations, where n_f is the number of free (or non-fixed) nodes. This system can be easily solved by a Gauss-Seidel type iterative formulation, which is given by:

$$x_J^{m+1} = x_J^m + \omega \frac{\sum_{I=1}^n (x_I^m - x_J^m) k_I(x_J^m)}{\sum_{I=1}^n k_I(x_J^m)} \quad (3.4)$$

where, ω is a relaxation factor, given as $0 < \omega < 2$. For an over-relaxation scheme one should choose $\omega > 1$. Selection of the most efficient ω depends on the problem.

The springs in the system demonstrated here are always under tension. The zero-force position is at the location that the spring length is zero and if the boundary nodes are not constrained, then the system shrinks to a single point. Compression formulation, in turn, yields infinite expansion and cannot be used. Indeed, the formulation is not physical, since it assumes the equilibrium length of the spring as zero. A more generalised formulation of the spring analogy takes the equilibrium length of the springs into consideration and is

called segment-spring analogy. Rewriting equation (3.3) for a given equilibrium length l_{IJ} and solving for the system, one has following general formula:

$$x_J^{m+1} = x_J^m + \omega \frac{\sum_{I=1}^n (x_I^m - x_J^m - l_{IJ}) k_I(x_J^m)}{\sum_{I=1}^n k_I(x_J^m)} \quad (3.5)$$

For most smoothing or optimisation based applications, this length is simply taken as zero, to be able to control the length of the mesh edge. However, for moving boundary problems, this length is taken as the initial length of the edge to guarantee that the optimised edge sizes do not change significantly.

What makes the spring analogy very flexible compared to the other methods is the selection of the spring stiffness. With slight modifications on the formula to ensure stability, user is free to select any stiffness function. Therefore, selecting appropriate equilibrium length and stiffness functions, the mesh can be controlled easily. For example, in the most common usage in mesh regularisation, the spring stiffness is chosen either as unity, or better as reverse edge length given as:

$$k_I(x_J) = \frac{1}{\|x_I - x_J\|} \quad (3.6)$$

According to Blom [118] this version of spring-analogy based methods is actually analogous to the Laplacian smoothing. The resulting system is very simple to solve and analyse, yet, very robust. For a demonstration of the method, a uniform mesh is randomised with an allowable mesh movement of up to twice the size of the edge length. Then the mesh is regularised with the formulation given by equations (3.5) and (3.6).

The results can be seen in Figure 3.2. As it is seen here, even though the randomised mesh is self-intersecting, the result is close to perfect. The test is repeated with very high random movement of the nodes with up to a thousand times of edge length, the result is still very successful. The convergence is also very fast. For example, for the set-up used in the test given in the figure, which is a 10000 cell mesh with twice edge length randomisation and an over-relaxation factor $\omega = 1.5$, the computation time for convergence to a maximum normal error of 10^{-5} is less than 20 seconds on a 1GHz Pentium class computer. In 3-D configurations the convergence is even faster, since the node-relocation information is more rapidly transmitted in Gauss-Seidel system.

3.1.1 Boundary conditions

In the solution of any physical system, correct identification of the boundary condition is of greatest importance. Although the spring network is an artificial system, the boundary conditions should be set carefully to achieve body-conforming mesh with desired edge sizes also at the boundary. The boundary nodes should be kept on the boundary, otherwise the

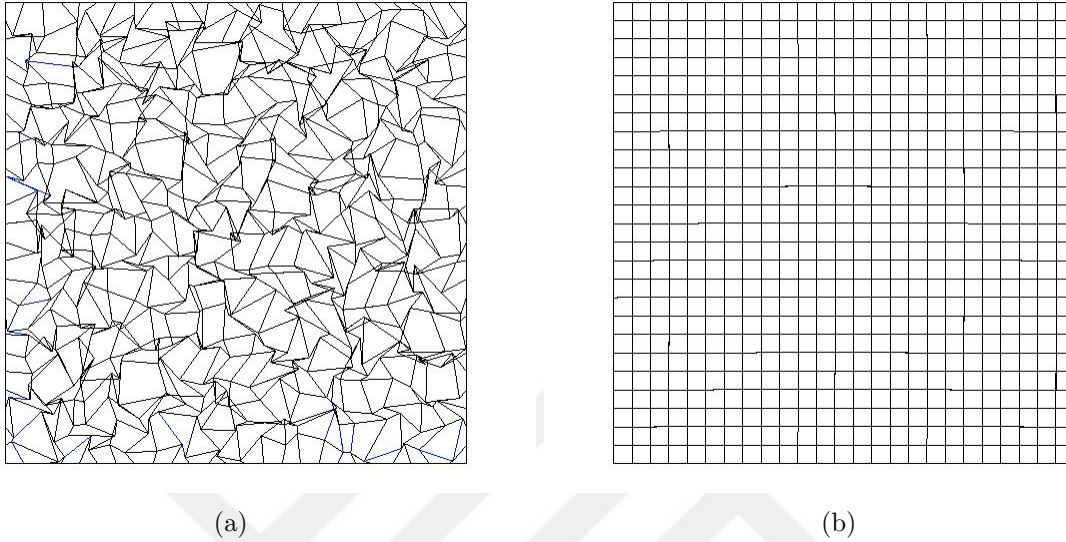


Figure 3.2: Mesh regularisation test for spring analogy.

spring system shrinks to a single point. However, fixing the nodes on the boundary results in a very stiff system, that doesn't allow nodes to move to the most optimal position, and also results in non-orthogonal, low quality cells near boundary. Boundary nodes should be allowed to move on the boundary that they belong to.

To solve this problem, springs, whose only one end are connected to surface nodes are mirrored with respect to the vectors normal to the boundary, introducing new springs. Stiffness of these springs are the same with the mirrored ones. Then this new surface spring system, which consists of surface springs, surface-connecting springs and the mirror springs are solved by letting only the boundary nodes move. The new system is shown in figure 3.3. The new system keeps surface nodes on the planes tangent to the surface at the initial location. New locations of the surface mesh vertices may not be on the surface, if the surface is curved. Therefore these nodes should be re-snapped to the closest surface, after point relocation completed.

This boundary treatment should be applied with care, in order to keep mesh topology valid. This method should be adapted for nodes on the topological edges, so that these vertices keep on the same edges. Additionally, vertices, which reside on the topological corners should be fixed on those nodes. To ensure simulation consistency, the surface vertices shouldn't change the surfaces they are associated to.

3.1.2 Adaptation of method for non-conformal meshes

As it is mentioned before, the study is conducted on the code of overlay-grid based mesh generator *HEXPRESS*. The grid generator relies on the non-conformal refinements of the

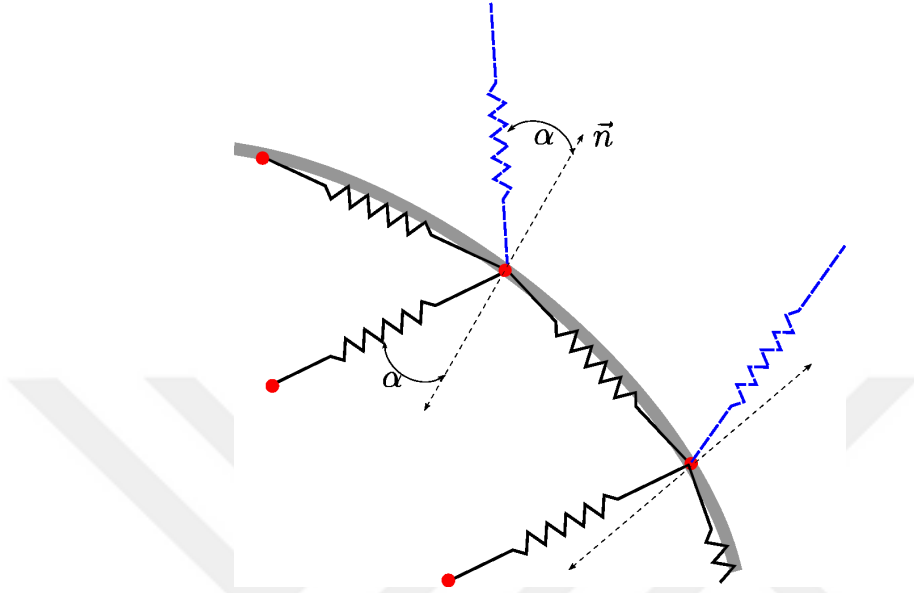


Figure 3.3: Boundary treatment for a spring system

mesh cells in order to capture the geometric details. Therefore, the mesh contains so-called hanging nodes, which is located at the centre of refined edge or the face. Existence of these nodes together with the refined cell edges poses two problems for spring-analogy based point relocation algorithm.

The hanging nodes cannot be handled as conformal ones, since they will be *pulled in* by the unbalanced spring force due to the non-existence of edges leaving the node hanging. The imaginary springs as it is employed for the boundary solutions seem to be a viable solution, however, this solution complicates the algorithm unnecessarily. Instead, a simple geometrical approach is chosen in this study. The hanging nodes are categorised into two, *two-connected* hanging nodes, which resides hanging on a parent edge and *four-connected* hanging nodes, that is hanging on a parent face, dividing it into four children faces. The definition is shown in Figure 3.4. These nodes are frozen through an iterative cycle. At the end of the cycle, first two-connected hanging nodes are relocated such that they are at the geometric centre of vertices of their parent edge. Next, four-connected nodes are relocated at the geometrical centre of the corresponding two-connected nodes, hence at the centre of their parent face.

The existence of the refinement operations is the second problem to tackle with. The refinement operation produces cells in different sizes, and applying the simple spring methods results in equally distributed edge sizes. This is not an desired effect, the cell refinement is applied in order to obtain smaller cells where necessary. Therefore the algorithm should respect these sizes. There are two options to solve this problem. First, since the algorithm is applied in the mesh generation process, the refinement operation is applied recursively. Therefore, at any moment of operation, all cells but that belongs to a particular refinement

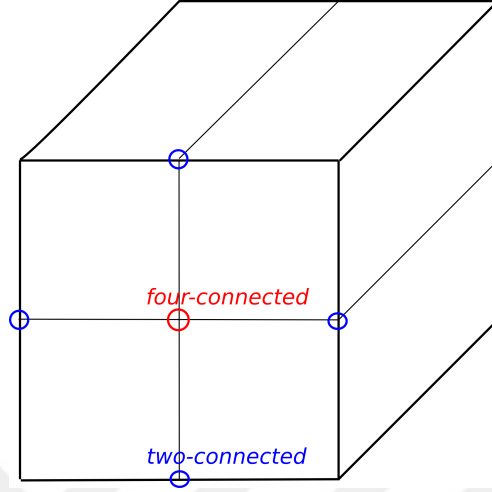


Figure 3.4: Two and four connected hanging nodes

level are ignored, and the spring system is built from these edges of particular refinement level¹. However, especially for the metric based versions of the method, initial experimentations conducted using this idea suggest that the multi-spring system approach results in too stiff spring system. As a result, it is decided to solve the complete spring system containing all unrefined edges in every refinement level.

In order to obtain the correct edge size, a more physical approach is followed. It is decided to correct the stiffness of the edges such that, the refined edges have half of the sizes of their parents, since the meshing algorithm applies non-conformal refinement operations, where each refined edge is divided exactly into two². Therefore, a weight factor for each spring stiffness is introduced to ensure such an edge-size distribution among refined and un-refined edges.

The Gauss-Seidel iterative algorithm used in Spring analogy is updated as following:

$$x_J^{m+1} = x_J^m + \omega \frac{\sum_{I=1}^n W_I (x_I^m - x_J^m) k_I(x_J^m)}{\sum_{I=1}^n W_I k_I(x_J^m)} \quad (3.7)$$

where, W_I is the stiffness weight factor for the edge, given as

$$W_I = 2^{2l_I} \quad (3.8)$$

where, l_I is the refinement level of the edge. At this moment, two points should be underlined. It should be noted that this weight factor is only applicable for the non-conformal

¹For the octree refinement based algorithm, the edges of finest level is the best candidate for this purpose.

²See section 2.3.4

mesh refinement operations as described in section 2.3.4. For conformal refinements, this weight should be altered to satisfy the required edge-size ratio between parent and children cells. The second issue is that, this factor is defined for the family of stiffness functions as given in equation (3.6). This is a very common form of stiffness, and the usage of the weight function given in equation (3.8) may be extended to many applications, however, this formulation should be altered when necessary

The final spring-analogy based vertex relocation algorithm for the non-conformal, octree, hexahedral mesh used through this study is given in the Algorithm 3.2. This basic algorithm is used through this study with slight modifications on stiffness function and related parameters, which will be explained in respective chapters.

Algorithm 3.2 Spring analogy for non-conformal hexahedra

```

1: repeat
2:   for all  $\chi_I \in (\mathcal{T} - \partial\mathcal{T} - \mathcal{T}^{hang})$  do  $\triangleright$  for each non-hanging and boundary vertices
3:      $E = UE\{\chi_I, \chi_J\}$   $\triangleright$  get the edges surrounding the node  $\chi_I$ 
4:      $k_{IJ} = k(E_{IJ}, \dots)$   $\triangleright$  get the stiffness of edge  $E_{IJ}$ 
5:      $W_{IJ} = 2^{l(E_{IJ}, k_{IJ})}$   $\triangleright$  get the correction of edge sizes of edge  $E_{IJ}$ 
6:      $\Delta\chi = \omega \frac{\sum_{I=1}^n W_I(x_I - x_J)k_I(x_J)}{\sum_{I=1}^n W_I k_I(x_J)}$   $\triangleright$  calculate the mesh movement
7:      $\chi = \chi + \Delta\chi$   $\triangleright$  move the node.
8:   end for
9:    $\triangleright$  Solve boundary system  $\triangleleft$ 
10:  for all  $\chi_I \in \partial\mathcal{T}$  do  $\triangleright$  for each boundary vertices
11:     $E = UE\{\chi_I, \chi_J\}$   $\triangleright$  get the edges surrounding the node  $\chi_I$ 
12:    for all  $\chi_J \notin \partial\mathcal{T}$  do
13:       $E = E \cup E_{IJ}^{mirror}$   $\triangleright$  find the mirror edges
14:    end for
15:     $\Delta\chi = \omega \frac{\sum_{I=1}^n W_I(x_I - x_J)k_I(x_J)}{\sum_{I=1}^n W_I k_I(x_J)}$   $\triangleright$  calculate the mesh movement
16:     $\chi = \chi + \Delta\chi$   $\triangleright$  move the node.
17:    Project the node to the closest surface
18:  end for
19:   $\triangleright$  Correct the hanging nodes  $\triangleleft$ 
20:  for all  $\chi_I \in \mathcal{T}^{hang}$  do
21:     $\chi^{hang} = UE\{\chi_I, \chi_J\} - \chi_I$   $\triangleright$  get the surrounding nodes
22:     $\chi_I = \frac{1}{n} \sum_{j=1, n} \chi_J^{hang}$   $\triangleright$  evaluate the new location
23:  end for
24: until  $\max(\Delta\chi) < TOL$   $\triangleright$  stop if the maximum node movement is less than tolerance
    
```

In order to test the new algorithm, a simple regularisation test, which is similar to the test applied on Cartesian meshes, is applied on a non-conformal mesh. The test is applied on a three dimensional mesh that contains multiple refinement levels. An initial mesh is

refined twice around a volute geometry. The resulted multi-level mesh is randomised as it is done in section 3.1 on page 42 with a randomisation radius of twice the edge length. In Figure 3.5, randomised (a) and regularised (b) meshes are shown from the top-view for clarity. The randomised mesh containing 114876 cells is shown at the left sub-figure (a). This mesh is regularised with the updated algorithm that is given above, with error tolerance of 10^{-5} of longest edge length and shown in the right sub-figure (b). Convergence required 38 seconds on a Pentium class 1.GHz computer.

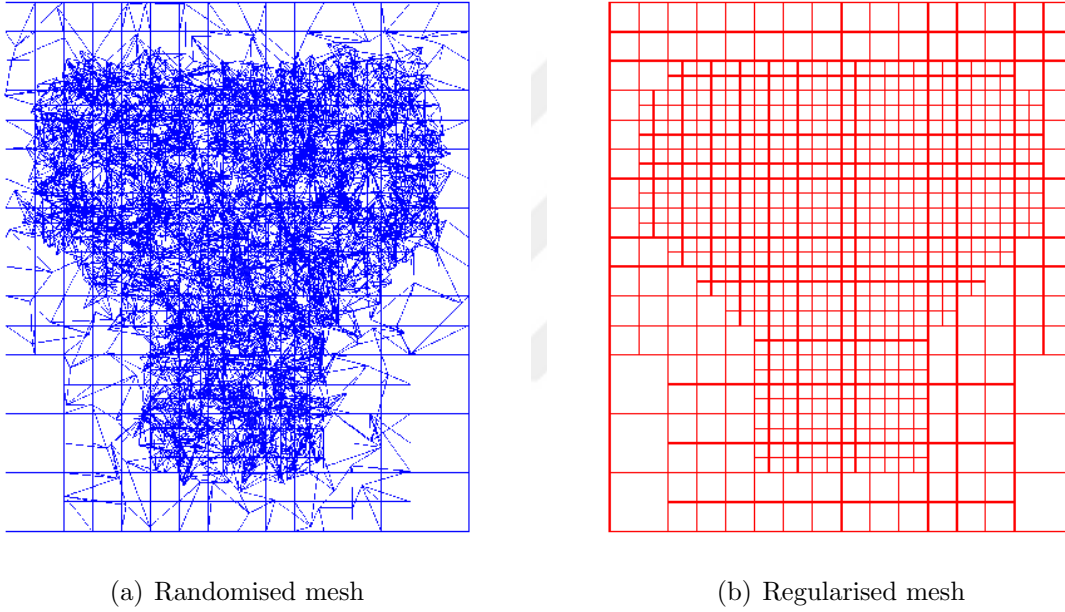


Figure 3.5: Regularisation test for octree spring analogy point relocation algorithm.

The convergence rate and speed of the method is very good and proves that the method is very efficient, hence applicable to any practical case. The randomised mesh has converged to the original mesh within a reasonably short time; therefore, it can be concluded that spring analogy based mesh vertex relocation algorithm is efficient enough to be used in mesh control process.

3.2 The notion of metric

Let Ω in $\mathbb{R}^d$ be considered as our domain and let χ be an arbitrary point in Ω . A metric or a metric tensor, $\mathcal{M}(\chi)$, is a $d \times d$ real symmetric positive definite matrix specified at any point χ in Ω . Since the metric $\mathcal{M}$ is positive definite, $\lambda_1, \dots, \lambda_d$, the eigenvalues of $\mathcal{M}$ are

positive and since $\mathcal{M}$ is symmetric, its eigenvectors are orthogonal, thus the metric can be written in the form,

$$\mathcal{M} = R\Lambda R^T \quad (3.9)$$

where, R is the matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues, given as:

$$\Lambda = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_d \end{pmatrix} \quad (3.10)$$

To give a more physical perspective to the definition of metric, given that the eigenvalues of $\mathcal{M}$ are always positive, Λ can be rewritten by replacing $\lambda_i = \frac{1}{h_i^2}$,

$$\Lambda = \begin{pmatrix} \frac{1}{h_1^2} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{h_d^2} \end{pmatrix} \quad (3.11)$$

where, $h_1, \dots, h_d$ can be defined as the anisotropic sizes in respective eigenvectors.

Briefly, the metric is split into d directions ($R_{1,\dots,d}$) and d sizes, introducing a more visual standpoint. The metric defined at a point can be considered as an ellipsoid around this point χ with a length of principal axes $h_1, \dots, h_d$, lying in the primary directions determined by eigenvectors R . The boundary of the domain encapsulated by this ellipsoid defined by these sizes and this rotation matrix and centred at χ will be denoted as $\chi + \mathcal{E}(\mathcal{M})$.

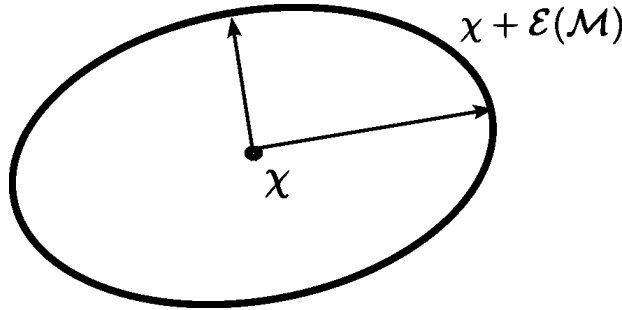


Figure 3.6: The physical representation of spherical metric

Therefore, $\mathcal{M}$ is called elliptic metric. This analogy can be extended to the other primary shapes, in other words, cylindrical or rectangular definition of the metrics are also possible. The properties of these primary shapes can be identified from the metric information and

details can be found in [119]. Regardless from the analogy, a metric contains a rotation and respective anisotropic size information, that might be used as a control tool for the local size and direction control.

3.2.1 Riemannian Metric

The field $\mathcal{M}(\chi)_{\chi \in \Omega}$ is assumed to be continuous. This means a *Riemannian* structure is formed in Ω . If $\mathcal{M}(\chi)$ is independent of χ , then problem reduces to the Euclidian case.

Let A and B be two points in $(\Omega, \mathcal{M}(\chi)_{\chi \in \Omega})$ and M be a point lying on a geodesic which is connecting A and B , denoted as Γ . Let $\gamma(t)$ be a parameterisation for Γ , such as $\gamma(0) = A$ and $\gamma(1) = B$. Then, $d_{\mathcal{M}}(A, B)$, the distance in Riemannian domain between points A and B is $l(\Gamma)$, the length of Γ , which is given by:

$$d_{\mathcal{M}}(A, B) = l(\Gamma) = \int_0^1 \sqrt{\gamma(t)^T \mathcal{M}(\gamma(t)) \gamma(t)} \quad (3.12)$$

In practice, this equation is quite difficult to use. However, if one reduces the geodesics to be the straight lines between A and B , the equation 3.12 is reduced to

$$d_{\mathcal{M}}(A, B) = l(\Gamma) = \int_0^1 \sqrt{\vec{AB}^T \mathcal{M} \vec{AB}} \quad (3.13)$$

In brief, $d_{\mathcal{M}}(A, B)$ is a distance relation between points A and B in Riemannian (Metric) space. This function also implies a transformation from Eulerian space to Metric space. Therefore, distance between two points on a geodesic can be calculated in locally transformed Riemannian space.

Reversing this analogy, by introducing a correct metric, the length of an edge can be calculated in any continuous Riemannian space. Therefore, by controlling the metric space, one can control sizes of the edges in any anisotropic direction.

3.2.2 Anisotropic mesh adaptation using spring analogy

Anisotropic mesh adaptation with spring-analogy based vertex relocation method is implemented by many researchers. For a good introduction one can see a series of papers of Habashi et. al. [105, 115, 116] and Tam [117], where such a technique is presented theoretically and practically for both structured and unstructured grids.

Suppose that, a Riemannian structure exists in domain Ω . Also consider a mesh topology $\mathcal{T}$ existing in Ω in this Riemannian space. Consequently, it is possible to apply any mesh regularisation-adaptation process in this domain with spring analogy. The regularised mesh will have optimised edge-length distribution in this space.

As it is depicted at the previous section, a transformation from the metric edge length $e_{\mathcal{M}}$ to the Eulerian edge length exists. Thus, the Eulerian mesh can be optimised in such a way that, its Riemannian counterpart is regularised. This can be achieved by adjusting stiffnesses of the spring such that the edge lengths are stretched in accordance with the local metric information. For a metric field $\mathcal{M}$ and a mesh edge between points x_I and x_J the metric length for this edge is given as:

$$e_{\mathcal{M}}(x_I - x_J) = \int_0^1 \sqrt{(x_I - x_J)^T \mathcal{M}(x_I - x_J)} \quad (3.14)$$

Therefore, $e_{\mathcal{M}}$ can be considered as the necessary edge length for uniform distribution in metric space. This length can be achieved by modifying the stiffness function given in 3.2 with a dimensionless form, such as:

$$k_I(x_J) = \frac{e(x_I - x_J)}{\|x_I - x_J\|} \quad (3.15)$$

The integral in equation (3.14) can be calculated approximately by any quadrature rule.

3.2.3 Application

In order to demonstrate the metric based relocation method two-dimensional tests are shown in this section. For the first test the following local size and rotation information are used:

$$h_1 = 1/240 + |0.25 - r| \quad (3.16)$$

$$h_2 = 1/60 + |0.25 - r| \quad (3.17)$$

$$R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \quad (3.18)$$

where, r is the distance from the centre point, and $\theta = \tan(x/y)$. The cell sizes h_1 and h_2 indicates a size distribution changing linearly through a circle with a diameter of 0.5 and centred at (0.5, 0.5). The point relocation algorithm is applied on a uniform 25×25 mesh. The metric based relocated mesh is shown in figure 3.7. As it is seen here, the mesh cells are accumulated around a circle with a diameter of 0.5, as expected. Resulted mesh has considerably good quality.

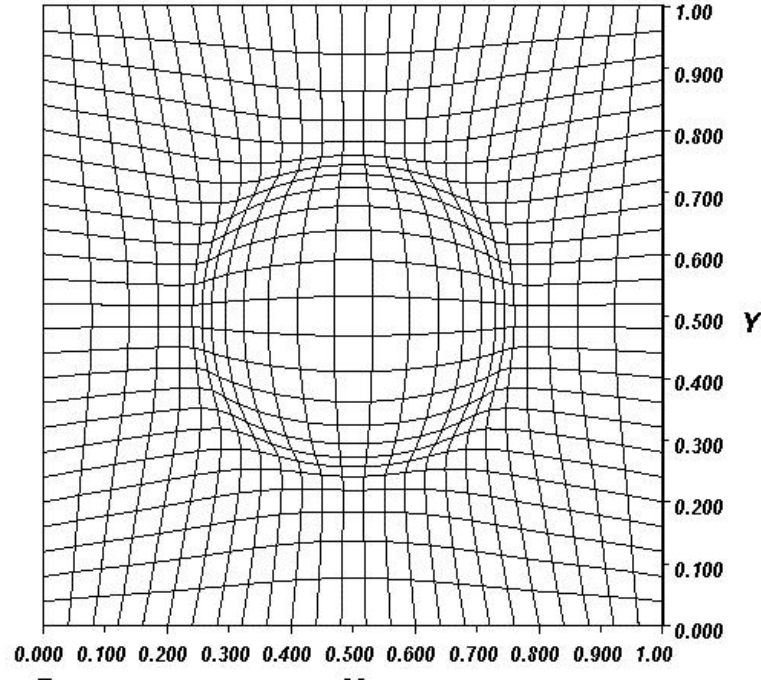


Figure 3.7: Demonstration for metric based vertex relocation method.

Another test case, is shown in figure 3.8. On the left corner of the figure, a reptile figure, which is used as a basis of the metric information, is shown in red colour. The rotation matrix and the size functions are calculated as function of the distance to the nearest boundary. Using this metric, a 100×100 grid is altered. The modified mesh is also presented in the same figure. As it seen here, the base shape is captured with the mesh successfully. The method used to evaluate the size function will be explained in Chapter 5.

As it is seen in these tests, using a metric field, one can easily control the properties of the mesh cells. In both figures, the base shapes, which are a circle for the first and the reptile figure in the second, can be identified easily from the adapted meshes. Although the cells are modified aggressively, there are no negative cells in the mesh. However, it should be noted here that, continuous metric fields are used in these tests. A discontinuity in the metric field may cause severe distortions and prevents the convergence of the code and should be avoided.

The presented method is also very fast. The latter mesh containing 10^4 cells converges to the displayed shape only in 18 seconds in 1 GHz Pentium class machine. Metric based version is significantly more expensive than the plain spring-analogy based method in

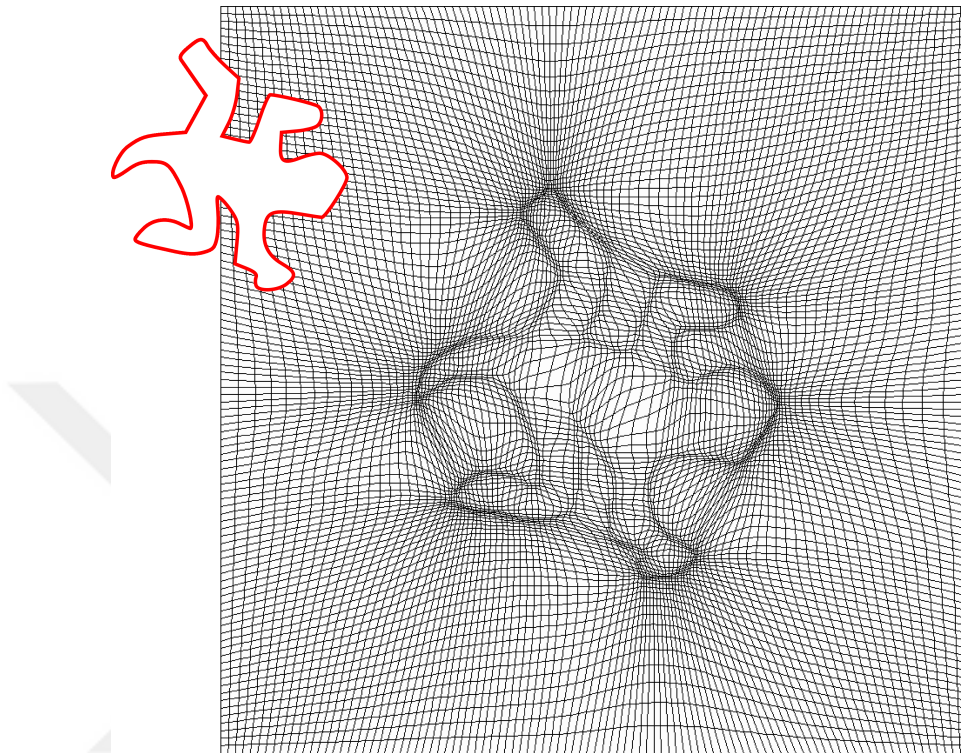


Figure 3.8: Vertex relocation technique based on distance information of an image.

terms of computational cost due to the matrix multiplication contained in every stiffness calculation. But still, the method is considered as extremely efficient, especially compared to the deformation analysis based methods and optimisation based methods.

3.3 Conclusion

In this chapter a robust and flexible method named spring analogy is discussed. Spring analogy is a very popular physics based vertex relocation method, which relies on the replacement of the mesh edges with a network of imaginary springs. These springs have certain stiffness and base lengths. The spring system is allowed to move under the spring forces to an equilibrium condition. For a certain selection of the stiffness function, adapted mesh is considered as optimal one in terms of edge length and the process is analogous to Laplacian smoothing. This process is demonstrated for both conformal and non-conformal hexahedral meshes.

The vertex relocation process can be used as mesh control process, by controlling the stiffness of the springs. In other words, the edge sizes can be controlled by altering the stiffness associated to that edge. However, controlling only the size doesn't mean controlling

the mesh, instead, the control of the anisotropic sizes is required.

To achieve this goal, Metric based mesh control process is defined. A metric is real, symmetric positive-definite matrix; therefore, it can be dissembled to its positive eigenvalues and orthogonal eigenvectors. For a given continuous metric field, a Riemannian structure exists. Therefore, if a mesh is adapted as a smooth mesh in Riemannian space using a appropriate metric that will contain desired anisotropic sizes and directions, the Eulerian mesh will contain such features. This idea is demonstrated with two two-dimensional test cases. The results are very successful and proves that the method can be used in mesh control process.

The only problem remaining is to choose an appropriate metric for the control process. Actually this selection has an uttermost importance to achieve a good mesh control medium. In this study, the main focus will be on the effective mesh generation control, keeping the inefficiencies of the underlying mesh generation algorithm in mind. The mesh generation algorithm is overlay grid based, therefore, it is very sensitive to the change of the orientation of the geometry. The algorithm is not aware of the local sizes and local orientations. The only means of capturing the geometry is applying mesh refinement based adaptation, which results in excessive number of elements in the mesh.

The solution of the problem is controlling the mesh with vertex relocation algorithm presented here with a metric that contains information about the local features of the geometry. In the forthcoming chapters, such metrics will be introduced.

Chapter 4

Natural Metrics

As it is discussed earlier, the major inefficiency of the grid based hexahedral mesh generation algorithms can be identified as the lack of orientation insensitivity of the method. One can include the effect of local thickness as a source of inefficiency in these methods, observing that, when there exists a thin section in the geometry many refinement operations should be applied in order to fill this thickness with cells in appropriate sizes. These operations increase the local cell count enormously. Anisotropic refinement at these regions seems to be a good solution, however, real life geometries barely shares the same orientation with the cells of adapting grid, leaving anisotropic refinements as an ineffective tool.

This observation leads us to the conclusion that a good meshing algorithm should not only be orientation sensitive, but also take the local thickness of the geometry into consideration. Then, the local feature inside the geometry can be identified by at least two parameters, namely the local thickness, and the local orientation of the geometry.

This conclusion immediately brings the idea of making use of the medial axis transform as a local shape descriptor. The medial axis transform contains the information of the distance to the nearest wall, therefore, naturally contains the thickness information. The local layout of the medial also contains (at least in 2-D) the directionality information. This makes the medial axis transformation an immediate candidate for the shape descriptor.

However, while considering the thickness, one refers to the ratio between the smallest dimension of a geometry to the largest one. This is the how a *thin* object is distinguished from a *small* object. The medial axis transform, gives a hint about the local size, however, it does not contain any information about *local length*, hence, the local anisotropy.

So, following the conclusions above, one can define the *Natural Layout* of the geometry, which is a concept to describe the local features of the domain, that may be exploited to construct a *geometry-aware* mesh generator. The natural layout of the geometry consists of the information about:

- local thickness, in other words the distance between two closest walls,

- local anisotropic sizes, which means the relative size information between length, width and thickness,
- local orientation, i.e. the anisotropic directions.

The solution of the orientation insensitivity of the grid based hexahedral mesh generation algorithms, is then achieved by detection of the natural layout of the geometry to be used as an input in mesh control process. For tetrahedral meshes, similar problems are also encountered by the researchers dealing with thin geometries. Plastic moulding problems contain such geometries, which consist of very thin sections in different directions. Meshing these geometries using traditional methods with cells of high isotropy, results in an enormous number of cells while attempting to put enough number of cells in thickness, which is a necessity for numerical simulation. Therefore, generating highly anisotropic meshes with cells elongated parallel to the geometry boundary is crucial in these applications. A couple of researchers addressed this issue and proposed some methods to generate such meshes.

Such a study has also been conducted recently by Gruau at Material Processing Center at l'École des Mines de Paris on tetrahedral meshes [119, 120]. As it is indicated by the researcher, if one can identify the local direction of anisotropy everywhere in the geometry rather than the bulk body, then very successful anisotropic modifications on the mesh topology can be applied within the geometry. The direction of local anisotropy is an indication of the natural directions of the geometry, therefore is an invaluable information for introducing the directional insensitivity. Combined with the correct anisotropic sizes, this information can be used to control the mesh generation process respecting anisotropic directions, with methods that are used by other researchers [121, 122].

The idea introduced by Gruau is calculating a metric at every point inside a domain by locating the biggest possible ellipsoid at every node on a background mesh. This metric, which is named by the author as natural metric, is shown to be successful on tetrahedral meshes with thin sections. In the following sections, the development of the method adapted to hexahedral meshes and more general geometries has been presented.

4.1 Metric for Natural Layout Representation

In the Section 3.2 the notion of metric is presented. In this chapter, it is concluded that a metric is a powerful tool for mesh control. So that, a metric that can represent the Natural Layout of the geometry can be directly used to control the mesh. In this section, some speculation about such an anisotropic metric will be made.

As it is discussed, the medial axis transform may be a good starting point for this metric, since the properties of the geometry at least in one direction are known. Let Ω in $\mathcal{R}^d$ be considered as our domain. The medial axis, denoted as $\partial_{\frac{1}{2}}\Omega$, is defined as the axis lying inside the geometry and located at the loci of maximal spheres. Then, let's define

the metric $\mathcal{M}_{\frac{1}{2}}$ that is located at any point χ on this medial axis. Using the analogy between the metric and the spheroid, made in section 3.2, one may consider this metric as a spheroid in Euclidian space, whose centre lies on medial axis. The boundary of the metric is denoted as $\chi + \mathcal{E}(\mathcal{M})$.

What we are searching for is an anisotropic measure of the geometry. So the local anisotropy of the geometry can be modelled with a stretched spheroid, primary axes of which lie on the metric eigenvectors. Let's locate the biggest spheroid centred at point $\chi \in \partial_{\frac{1}{2}}\Omega$ in the domain, as it is shown in Figure 4.1. For two parallel walls, the vector from the medial axis to the closest wall should be equal to the biggest eigenvector of this metric; that represents the smallest size from equation (3.11). This is true by the definition of the medial axis, which is defined as located at the centre of tangent spheres. This condition may not be true for non-parallel cases as in the corners. However, this will be speculated later.

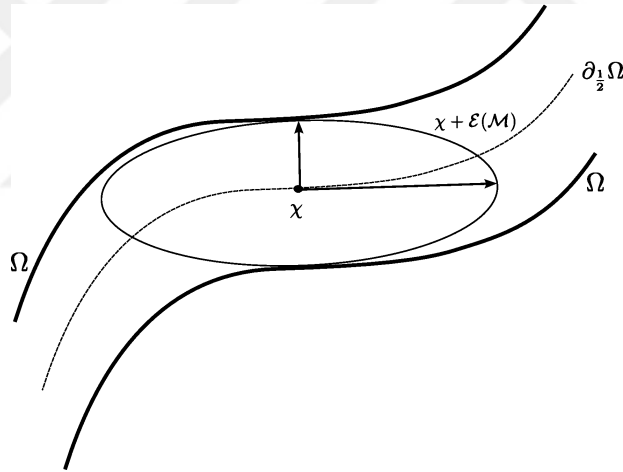


Figure 4.1: Spherical metric located at the center of an object.

Now, at least one axis of the spheroid is known, which is the shortest one. This means one anisotropic direction is known, which is on the direction of biggest eigenvector by equation (3.11). It is indicated that, the smallest eigenvector, (or largest axis) should reside on an orthogonal direction to the largest one, since the metric is defined as positive definite and parallel $d * d$ matrix. For the condition of parallel surfaces, this direction resides at the same plane of medial surface (for the two dimensional case, it is on the same direction of the medial). From the figure it seems like this direction and size can be selected as the distance to the farthest wall. However, this conclusion is not always true. Let's consider the geometry shown in Figure 4.2.

In this figure it is seen that, if the parallel walls are not bounded by any curvature or a closed wall, then the biggest spheroid fit into the volume can be too large in longitudinal direction. Therefore, using the medial axis information as a direct measure of

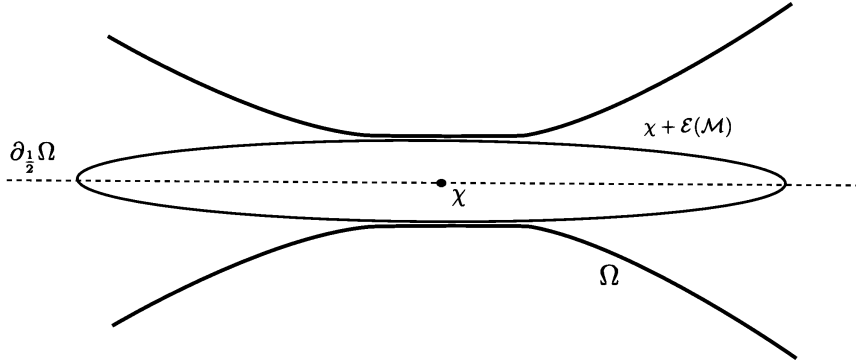


Figure 4.2: Diverging parallel walls

local anisotropy is not possible. The size of the metric in the lateral directions should be controlled in order to prevent erroneous anisotropic measures.

Actually, the detection of the anisotropic size information on the medial plane is a very challenging problem. An intelligent approach, the so called the *bubble growth* algorithm, tries to solve the problem of detecting this information.

4.2 Bubble Growth Technique

Suppose that a very elastic ball, or a soap bubble, is located at the point χ . Then this ball is inflated inside the domain. Until this ball touches to the wall, its shape is spherical. As it keeps expanding, the ball progressively fills the void, and for a sub-domain in the geometry located between two parallel boundary, this ball takes a shape between a rectangle and an ellipse like it is shown in the Figure 4.3(a).

Although the shape of the bubble is not spherical anymore, this information can still be used. The shape of the bubble changes from a perfect sphere to a perfect rectangle for an infinitely long parallel channel, (or a closed box). Therefore, one can replace this shape with a spheroid, or a spherical metric, so that the anisotropic directions of the bubble can be defined. Of course this procedure introduces an approximation error.

At this point two questions arises: first, how to control this error, second, how can we use this technique to determine the local anisotropic directions. Luckily, the answer of the first question solves the second. In the left part of the Figure 4.3, the bubble growth procedure is shown. The bubble grows to the natural limits of the channel, which is indicated in the figure with green colour. At this point, the shape is more like a rectangle and it contains a certain amount of approximation error, since it deviates from ellipsoidal shape by a certain amount. However, if this bubble keeps growing, it will start changing its shape dramatically, as it is seen in the right-side of the same figure with red line. Hence, the shape deviates very much from the ellipsoidal shape.

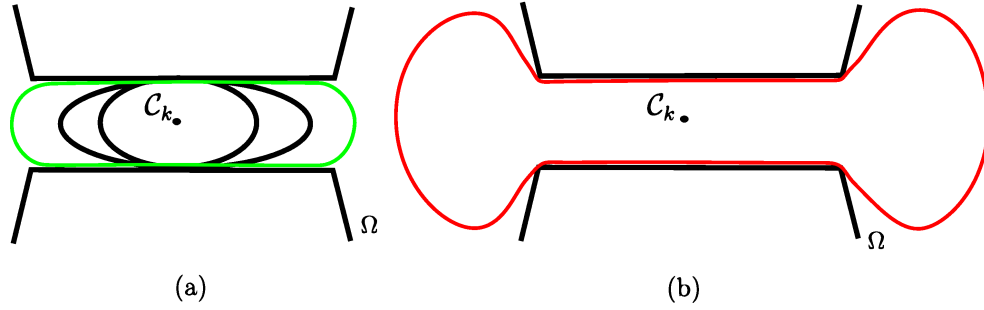


Figure 4.3: The bubble growth algorithm.

The big distortion of the bubble shape indicates that the bubble front exits a sub-domain, where natural layout is the same or similar for every point, and enters another subdomain, probably a curved section, or a bulkier section of the geometry. This is an indispensable information, since one can now distinguish a large body from a small detail, or, can sense the change in curvature. What needs to be done is to stop the bubble growth at the last moment, when the bubble still keeps a good shape. The measure of how *good* the bubble shape is, can be set by making an interpolation from bubble to spheroid, and controlling this interpolation with a good error estimation method. Therefore, a *good-shaped* bubble is distinguished from an *ugly-shaped* bubble, using this error estimation.

Although the development of the basic idea is built around it, one of the advantages of the bubble growth algorithm is that, it is not bound up to the medial. A bubble started to grow from any point inside the geometry can be interpolated to a spheroid; that makes the metric available everywhere in the geometry. This feature, in fact will help to develop the algorithm more flexibly.

In the following sections, the interpolation and error estimation algorithms for bubble-growth algorithm are introduced theoretically. Then, application of this method with the hexahedral meshes will be explained.

4.3 Theoretical Foundation

4.3.1 Notations

Although it can be generalized to any dimensions, since all the practical test cases will be in 3D, the domain Ω will be considered as $\Omega \in \mathbb{R}^3$. Let $\mathcal{N}$ be a finite set of cells from Ω . $\mathcal{T}$ is a mesh topology defined on the set of nodes $\mathcal{N}$ and consists of hexahedral elements. One can consider successive vicinities around an hexahedral element as:

$$\mathcal{N}_0 = \{S\} \quad (4.1)$$

$$\mathcal{N}_k = \mathcal{V}(\mathcal{N}_{k-1}) - \bigcup_{0 \leq l < k} \mathcal{N}_l + \bigcup_{0 \leq l < k} (\mathcal{N}_k \cap \partial\Omega) \quad (4.2)$$

where, $\mathcal{V}(\mathcal{N})$ is the union of neighbouring cells for every cell in $\mathcal{N}$. In brief, $\mathcal{N}_k$ indicates the sets of cells of the k^{th} vicinity around a cell. The centre of the vicinity $\mathcal{N}_k$ is indicated as C_k . The initial vicinity $\mathcal{N}_0$ is nothing but the cell itself.

Let's define $\mathcal{T}_k^{ext}$ as the sets of elements in $\mathcal{T}$ at the exterior of the vicinity k . In other words, $\mathcal{T}_k^{ext}$ are sets of elements which do not contain any element from $\bigcup_{0 \leq l \leq k} \mathcal{N}_l$. Then the front of the bubble of order k is the set of faces on the bubble boundary which is defined as

$$\mathcal{F}_k = \bigcup_{F \in \partial\mathcal{T}_k^{ext}} F \quad (4.3)$$

where, F denotes the cell faces in $\mathcal{T}$. An important property of our definition should be underlined here. According to equations (4.2) and (4.3), the faces located on the object boundary, which is the member of the previous vicinities are also included in the list of the faces. In other words, the surface faces that are covered by the previous bubble, $\mathcal{F}_{k-1} \cap \partial\Omega$ are reused.

For completeness, $\partial\mathcal{T}_k^{ext}$ denotes faces on the elements $\mathcal{T}_k^{ext}$, whose nodes are at the vicinity $\mathcal{N}_k$. In Figure 4.4, $\partial\mathcal{T}_k^{ext}$ is shown by red lines, where $\mathcal{N}_k$ is the nodal points on this lines.

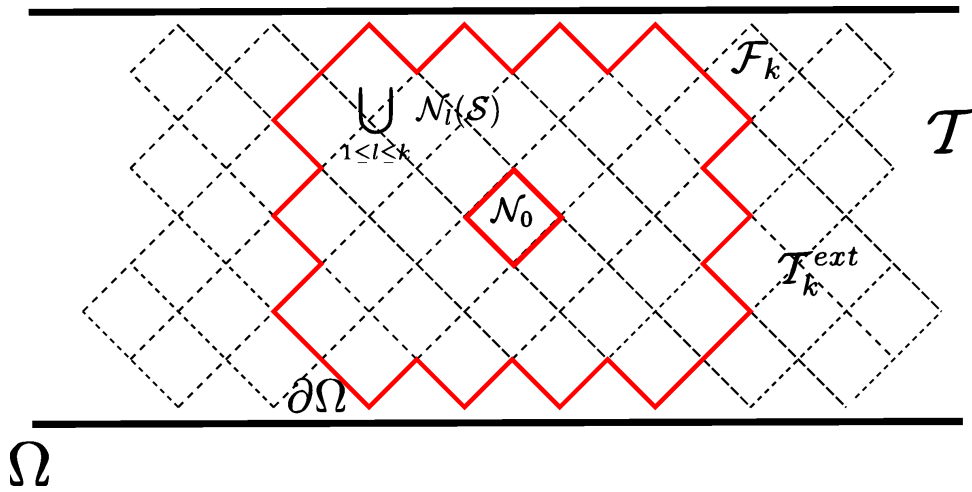


Figure 4.4: Notation for bubble growth

4.3.2 Elliptic Interpolation

Consider an ellipsoid in $\mathbb{R}^d$. From tensor algebra, an elliptical metric that fits this shape can be written as the tensorial products of the vectors from the centre of ellipsoid to the surface of ellipse, as it is seen in figure 4.5.(a) and given in equation (4.4). From the basic notion of metric, this equality can be re-written in two-dimensions as in equation (4.5).

$$\mathcal{M}^{-1} = \sum x \otimes x = \sum (x_i x_j)_{1 \leq i, j \leq d} \quad (4.4)$$

$$= \begin{pmatrix} a^2 & 0 \\ 0 & b^2 \end{pmatrix} \quad (4.5)$$

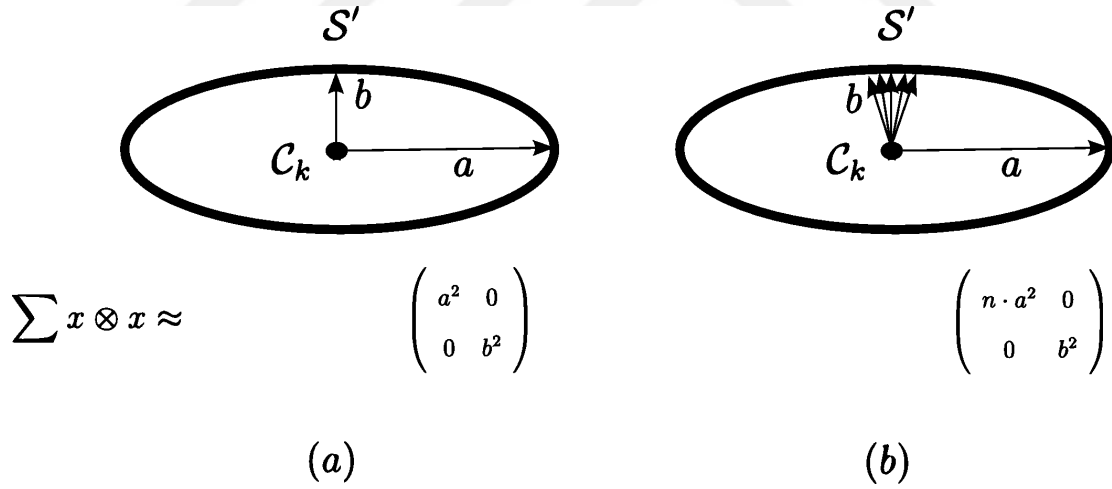


Figure 4.5: The effect of point density on the tensor product

In these equations, x is the vector from the centre of the ellipse to a surface point, a, b are the primary axis lengths of the two-dimensional ellipse, and $\otimes$ denotes the tensorial product.

Let C_k be the barycentre of the bubble and S' is any point on the bubble surface $\mathcal{N}_k$. Therefore, one can re-write the equation (4.4) on the bubble surface in order to find a metric in a similar manner as:

$$\mathcal{M}^{-1} = \sum_{S' \in \mathcal{N}_k} (S' - C_k) \otimes (S' - C_k) \quad (4.6)$$

It should be noted that, the equation (4.4) is only valid if the points around the spheroid are selected uniformly. However, if the points are condensed in any region on the boundary as it is seen in figure 4.5.(b), one of the metric lengths is overestimated. The metric found in this way is given as:

$$\sum x \otimes x \approx \begin{pmatrix} n \cdot a^2 & 0 \\ 0 & b^2 \end{pmatrix} \quad (4.7)$$

This is a very important point, since the background mesh that will be used for extracting natural metric will be an octree hexahedral mesh, containing very different cell sizes in different cell refinement levels. In order to remove this effect, one can integrate the tensor product over the surface of the bubble:

$$\mathcal{M} = \left(\int \bigcup_{F \in T_k(\mathcal{S})} F (x - \mathcal{C}_k) \otimes (x - \mathcal{C}_k) dx \right)^{-1} \quad (4.8)$$

Therefore, in the final algorithm the tensor products of points associated to each face $\mathcal{F}_k$, as it is shown in the Figure4.6, are integrated on these surfaces for a normalization of the effect of the cell size differences.

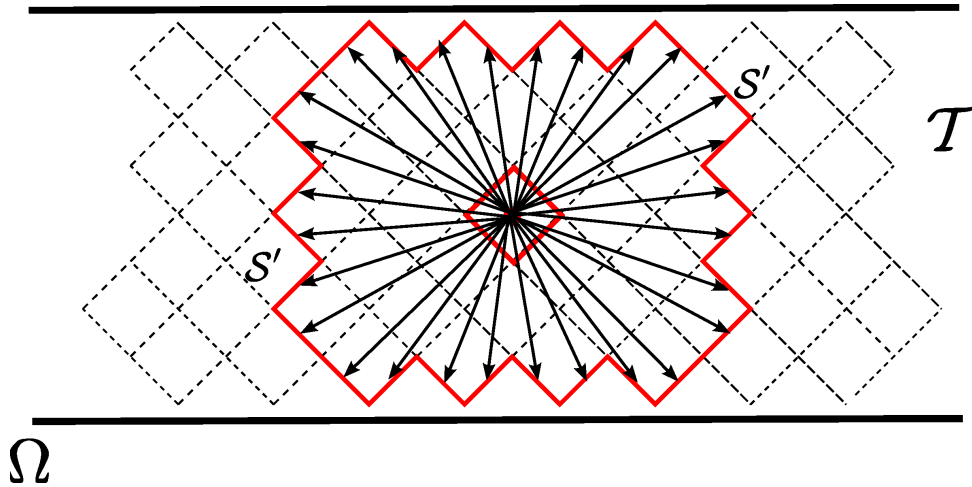


Figure 4.6: Metric calculation on a bubble.

4.3.3 Correction of metric size

The metric found with the procedure described above has a good shape but a bad scale. In order to control the metric extraction process, the size of the metric should be corrected. This can be achieved by scaling the metric with a correction factor c , that minimizes the error of elliptic interpolation. The Riemannian distance from any point $\mathcal{S}'$ to the metric centre is given by

$$d_{\mathcal{M}}^2 = \|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}}^2 \quad (4.9)$$

This value should be 1 if the point is on the ellipsoid. Therefore, the error of the elliptic interpolation for a single point can be defined as:

$$e_{local} = \left| \|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}}^2 - 1 \right| \quad (4.10)$$

So the problem can be defined as a minimization problem of function $f(c)$; to find a c , such that c minimizes the total error produced by all the points. The function $f(c)$ is written as:

$$f(c) = \sum_{\mathcal{S}' \in \mathcal{N}_k} \left(c \|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}(\mathcal{S})}^2 - 1 \right)^2 \quad (4.11)$$

Rewriting,

$$f(c) = \left(c^2 \sum_{\mathcal{S}' \in \mathcal{N}_k} \|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}(\mathcal{S})}^4 - 2c \sum_{\mathcal{S}' \in \mathcal{N}_k} \|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}(\mathcal{S})}^2 + 1 \right) \quad (4.12)$$

Solving for minimizing c ,

$$\frac{df(c)}{dc} = 0 \quad (4.13)$$

$$= \left(2c \sum_{\mathcal{S}' \in \mathcal{N}_k} \|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}(\mathcal{S})}^4 - 2 \sum_{\mathcal{S}' \in \mathcal{N}_k} \|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}(\mathcal{S})}^2 \right) \quad (4.14)$$

results in,

$$c = \frac{\sum_{\mathcal{S}' \in \mathcal{N}_k} \|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}(\mathcal{S})}^2}{\sum_{\mathcal{S}' \in \mathcal{N}_k} \|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}(\mathcal{S})}^4} \quad (4.15)$$

Therefore the following corrected metric has a good shape, and also minimum elliptic interpolation error.

$$\mathcal{M}_{corr} = c * \left[\int_{\bigcup_{F \in \partial T_k}} (S' - \mathcal{C}_k) \otimes (S' - \mathcal{C}_k) \right]^{-1} \quad (4.16)$$

4.4 Numerical Implementation

After the problem is formulated, the next step is to implement a computer algorithm to apply the ideas presented above. Consider there exists a mesh, which is preferably isotropic inside the domain. Select a cell, which is our first bubble, $\mathcal{N}_0$, where the faces of this cell are in $\mathcal{F}_0$. Then the shape of the zeroth bubble is considered as spherical, it is enlarged by adding new neighboring cells from faces $\mathcal{F}_0$ to obtain $\mathcal{N}_1$ and hence, $\mathcal{F}_1$. This procedure continues on $\mathcal{F}_k$ until the bubble is not spherical anymore. Reminding the properties of this bubble,

- The bubble first hits the closest walls, and continues to grow in the direction of the natural directions of the body. For example, in a narrow channel, the bubble will grow through the axis of the channel. Therefore, the shortest anisotropic direction is the direction to the walls, the longest one is in the local direction of the channel. This guarantees the capturing of local anisotropic directions.
- If there exists a sharp turn in the geometry or a strong deviation of the local size, for example a turning channel, or abrupt change of the channel size, shape of the bubble will not fit to an elliptical metric anymore. This guarantees the locality as well as anisotropic size control, which does not exist in medial methods.

The procedure is explained briefly in Figure 4.7. Note that, this figure is simplified by excluding non-conformal cells.

Figure 4.7: Bubble growth in a volume. **Bold** lines indicate the object boundary, **blue** lines are the background mesh where the metric is calculated, **red** lines are consecutive bubbles and **green** ellipse is the natural metric.

4.4.1 Adaptation of the Method to Hex Meshes

The bubble growth algorithm is implemented using the formulation given above. As it is indicated previously, the original work is implemented by Gruau [119, 120] on tetrahedral

meshes. The direct implementations of the techniques which are applied in tetrahedral mesh generation, most of the time, are not available for the hexahedral mesh generation. For example conformal coarsening operations are not possible most of the time in hexahedral cases. Similarly, flipping or dicing operations are extremely limited, since they are not mature enough to be used in real-life applications, yet. These constraints lead us to calculate the natural metric field on a background mesh and use this information afterwards in the control of the mesh generation process.

The metric extraction algorithm is implemented in the framework of the commercial hexahedral mesh generator HEXPRESS. For this purpose a background mesh composed of octree hexahedron are produced with the mesh generator. The generated mesh is adapted by recursive subdivision, however, in order to save the computational resources, the mesh is not snapped. In other words, the final background mesh is not body-conforming. This background mesh should satisfy a couple of features. First of all, there should exist enough cells in the thickness. Otherwise one ends up with regions of domain without a metric. For the sake of good interpolation, it is decided to select almost isotropic cells and isotropic refinements.

Since the control hexahedral meshes is the aim of this study, and since our goal is to develop a tool for more general geometries, instead of Gruau target geometries of moulding problems, a different implementation path is followed. The main differences between two studies can be listed as follows:

- The bubble growth algorithm starting point $\mathcal{C}_k$ is selected as a cell centre in hexahedral algorithm, instead of a node in tetrahedral meshes.
- Instead of the nodes of the front, the centres of the faces on the front are used as the points on the surface, $\mathcal{S}'$, to calculate tensor products. These two steps simplified the algorithm significantly and resulted in a more robust implementation.
- Gruau first introduced a fixed centre of the bubble, which is the node itself, then he improved the algorithm where centre $\mathcal{C}_k$ is calculated for every step as the geometric centre of the bubble. We have observed that, fixing the node brings together prematurely ended bubble growth; on the contrary, freely floating bubble centre may result in a metric, the centre of which is too far away from the cell. Therefore, in this study, a floating centre approach with an exclusive control of location of $\mathcal{C}_k$ is applied by making use of the cell-based implementation of algorithm. For details see Section 4.5.2.
- The error tolerance is chosen differently from the original study (see Section 4.4.2).
- In the original study, the aim is to put enough number of layers of cells inside the thickness for moulding problems. Therefore, Gruau scales the shortest edge of the metric with the number of layers to ensure that there exist enough cells in the thickness. In our study, this is not necessary, since more general geometries are aimed in our study.

4.4.2 Bubble Quality Criterion

One of the major undefined points in the bubble growth algorithm is how to determine if the bubble is in a good shape or not. This decision can easily be made comparing the maximum deviance of Riemannian distance on the bubble from metric radius,

$$e = \max \left(\|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}}^2 - 1 \right) \leq TOL \quad (4.17)$$

where, e is the maximum error and TOL is the error tolerance, to be given as a parameter to the program. This parameter given by Gruau as 76%, which is determined empirically after a series of tests applied on different meshes.

However, it is found that this limit is too restrictive for our purpose. The main reason for this is the staircase-like shape of our front, resulted from the characteristic behavior of hexahedral meshes. To be able to find a good tolerance, a series of experiments are carried out to detect the limit conditions. The most important tests are given below.

The first test is to determine the maximum error found on the surface mesh of a simple bar-geometry in three-dimensions. The bar has dimensions $1.0 * 1.0 * 5.0$. The maximum approximation error for a bubble defined at the surface mesh for different surface mesh subdivisions is given in Table 4.1.

# of Mesh Divisions	maximum error
100*20*20	0.678984
50*10*10	0.62264
25*5*5	0.523374
15*3*3	0.496224
5*1*1	0.302436
3*1*1	0.280856

Table 4.1: The elliptic interpolation error for the surface mesh for the bar geometry

As it seen, the maximum error produced is less than the limit given in the original study. This is due to selection of face centres instead of nodes in our algorithm to calculate the vector $\mathcal{S}' - \mathcal{C}_k$. So that one prevents the higher error produced by the corners of the bar. In other words, our algorithm produces *smoothed* corners. However, as it is seen in the table, this error keeps increasing with smaller cell size, so that for the limit case of division approaching to infinity, one should use the error estimation given in the original study.

One of the major drawbacks of using hexahedral background mesh is the staircase-shaped front. To investigate the staircase effect, two more tests are made. For the first one, the geometry used in the first example is rotated 45° in two axes to produce the

maximum staircase effect. The maximum errors found for different cell sizes are given in Table 4.2.

Cell Size	maximum error
0.25	0.556438
0.167	0.682666
0.125	0.590074
0.1	0.621412

Table 4.2: The elliptic interpolation error for the surface mesh for the rotated bar geometry

The second test addressing the staircase effect is made on a unit cube. The metric is calculated at the moment, when a bubble started from the centre node reaches the boundary. At this moment the bubble is diamond shaped, and contains maximum amount of the staircase cells. The maximum errors for this case is shown are the Table 4.3.

# of Mesh Divisions	maximum error
3*3*3	0.441989
7*7*7	0.72597
21*21*21	0.855279

Table 4.3: The elliptic interpolation error for the surface mesh for the diamond shaped front.

It is seen that, the maximum error for the diamond shaped bubble is 0.855. This number is significantly higher than the one in the original study. Therefore, in this study, the metric quality criteria of 0.88 is used, concerning the staircase effect.

$$e = \max \left(\|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}}^2 - 1 \right) \leq 0.88 \quad (4.18)$$

4.4.3 Algorithm for successive bubbles

The calculation of the enlargement of the bubble is the key issue in the implementation. There are several alternatives of selecting the neighboring cells. Since a cell-centred approach is chosen for implementation, the selection of the neighborhood can consist of face neighbors, edge neighbors and finally corner neighbors. For a simpler algorithm, only face neighbors are used for expansion of the bubble. The basic algorithm is given in Algorithm 4.2.

Algorithm 4.2 Algorithm for growth for the bubble $\mathcal{N}_k$ of order k around cell $\mathcal{S}$

Require: $k, \mathcal{N}_k$

- 1: **for all** $\mathcal{S}_i \in \mathcal{N}_k$ **do** $\triangleright$ for each cell on the front $\mathcal{N}_k$
 - 2: **for all** $F \in \mathcal{S}_i$ **do** $\triangleright$ for each face F of $\mathcal{S}_i$
 - 3: $\mathcal{V}_k(\mathcal{S}) \leftarrow [\mathcal{V}(\mathcal{S}_i)]_F$ $\triangleright$ add the neighboring cell on face F to the neighbor list
 - 4: **end for**
 - 5: **end for**
 - 6: $\mathcal{N}_{k+1} = \mathcal{V}_k(\mathcal{S}) - [\mathcal{V}_k(\mathcal{S}) \cap \mathcal{N}_k]$
-

However, it should be noted that, the algorithm given here is an over-simplified version. Indeed, the real algorithm is much more complex due to the existence of the cells in octree type multiple refinement levels. Enlarging the bubble with the basic algorithm yields a non-uniform growth of the bubble due to different sizes of the cells in different refinement levels. Therefore, the actual algorithm that takes care of this size difference is a recursive one based on Algorithm 4.2. In this algorithm, first the difference between the smallest and the biggest refinement levels r_{min} and r_{max} in the $\mathcal{N}_{k-1}$, is detected. Then the neighborhood search radius, D for cell of refinement level is found as,

$$D = 2^{r-r_{min}} \quad (4.19)$$

then, the neighborhood of degree D for a cell is defined as.

$$\mathcal{V}_{D(\mathcal{S})}(\mathcal{S}) = \mathcal{V} \left(\bigcup \mathcal{V}_{D(\mathcal{V}(\mathcal{S}))}(\mathcal{S}) \right) \quad (4.20)$$

Which guarantees the correct step size in every direction. Therefore, the lines 4-6 of Algorithm 4.2 should be read as a recursive function that replaces $\mathcal{V}_k(\mathcal{S})$ with the recursive function $\mathcal{V}_{D(\mathcal{S})}(\mathcal{S})$.

4.4.4 Algorithm for Metric Extraction for single cell.

The algorithm for metric extraction around a single cell is given in Algorithm 4.3. This algorithm basically consists of three steps. In the first step, the metric is calculated by integrating the tensor products of the vectors from the bubble centre to the centres of the faces of the front (steps 1-9). The next step is to find a correct scaling factor which minimizes the interpolation error, as shown between lines 10-15. Finally the maximum error should be calculated in order to ensure that the bubble has a good shape. This is done by comparing the Riemannian length of every vector from bubble centre to the centres of the mesh faces with unity. This error should be less than the error tolerance given by equation (4.17).

Algorithm 4.3 Metric Calculation algorithm for single cell

```

1:  $\mathcal{M} = [0]$   $\triangleright$  begin with zero metric
2:  $\mathcal{A} = 0$   $\triangleright$  surface area of the bubble
3: for all  $\mathcal{S} \in \mathcal{N}_k$  do  $\triangleright$  Loop over the cell faces on the bubble front
4:    $\vec{\mathcal{S}} = \mathcal{S}' - \mathcal{C}_k$   $\triangleright$  Calculate the vector between Face center to bubble center
5:    $A_{face} = A(\mathcal{S}')$   $\triangleright$  Get the surface area of the face
6:    $\mathcal{A} = \mathcal{A} + A_{face}$ 
7:
8:   
$$\mathcal{M} = \mathcal{M} + A_{face} * ((\mathcal{S}' - \mathcal{C}_k) \otimes (\mathcal{S}' - \mathcal{C}_k))$$
 (4.21)
    $\triangleright$  Add the tensor product of this vector with itself multiplied by the face area to the metric
9: end for
10:  $\mathcal{M} = \left(\frac{1}{\mathcal{A}}\mathcal{M}\right)^{-1}$   $\triangleright$  invert the matrix
    $\triangleright$  Then scale the vector
11:  $\Sigma \|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}}^2 = \Sigma \|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}}^4 = 0$ 
12: for all  $\mathcal{S} \in \mathcal{N}_k$  do
13:
14:   
$$\Sigma \|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}}^2 + = \|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}}^2$$
 (4.22)
   
$$\Sigma \|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}}^4 + = \|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}}^4$$
 (4.23)
    $\triangleright$  Calculate the sum of second and fourth tensor products of the same vectors to find scaling
15: end for
16:  $\mathcal{M} = \frac{\Sigma \|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}}^2}{\Sigma \|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}}^4} \mathcal{M}$   $\triangleright$  scale the metric with this corrector
17:  $err = 0$   $\triangleright$  calculate the error
18: for all  $\mathcal{S} \in \mathcal{N}_k$  do
19:
20:   
$$err = \max\left(err, \left|\|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}}^2 - 1\right|\right)$$
 (4.24)
    $\triangleright$  In the previous step error is minimized. Now find the maximum error of scaled metric
21: end for
22: return  $err$ 

```

4.4.5 The Basic Algorithm For Natural Metric Generation

After defining the pieces of the algorithm, one can calculate the natural metric for every cell in the mesh $\mathcal{T}$ in the domain Ω . The complete algorithm merges the routines described

above. Basically the bubble growth algorithm given in Algorithm 4.2 is followed by metric calculation algorithm as shown in Algorithm 4.3. The interpolation error found in this algorithm is compared with the quality criterion defined in Section 4.4.2. If this error is less than the quality criteria, then the bubble is kept increasing. Else, the last good metric is selected as the natural metric assigned to cell $\mathcal{S}$. The basic routine is given in Algorithm 4.4.

Algorithm 4.4 The basic algorithm for natural metric extraction in mesh $\mathcal{T}$.

Require: $\mathcal{T} \in \Omega$

```

1: for all  $\mathcal{S} \in \mathcal{T}$  do  $\triangleright$  for each cell in the mesh
2:    $\mathcal{N}_0 = \{\mathcal{S}\}$ 
3:    $k = 0$ 
4:   repeat
5:      $k = k + 1$ 
6:     Calculate  $\mathcal{N}_k(\mathcal{S})$   $\triangleright$  enlarge the bubble with Bubble growth algorithm
7:     Calculate  $\mathcal{M}_k(\mathcal{S})$   $\triangleright$  get the metric with metric algorithm lines 1-9
8:      $\mathcal{M} = c * \mathcal{M}$   $\triangleright$  scale the bubble with metric algorithm lines 10-14
9:      $e = \max(\|\mathcal{S}' - \mathcal{C}_k\|_{\mathcal{M}}^2 - 1)$   $\triangleright$  get interpolation error with metric algorithm lines 15-18
10:   until  $err > TOL$   $\triangleright$  stop if the error is bigger than the tolerance
11:    $\mathcal{M}(\mathcal{S}) = \mathcal{M}_{k-1}(\mathcal{S})$ 
12: end for
```

4.5 Improvements on the Algorithm

After implementing the basic algorithm, some inefficiencies and weaker parts of the method became visible. These inefficiencies can be summarized as:

- The metric extraction requires enormous computational time. This is due to high computational cost of integration of the tensor products (and other similar operations such as error calculation) on the bubble surface, as well as matrix operations like matrix inversion.
- It is seen that the metric extraction is not robust enough. The quality criterion given in Section 4.4.2, is found to be too restrictive, such that a successful metric cannot be computed in many cells in the geometry. Relaxing this criterion resulted in erroneous metric sizes and directions. A very important observation is made at this point, that is, the larger error is produced at the cells near the boundary. This is an expected behavior, since a bubble that is started growing closer to the boundary is less likely to

be symmetric, hence produces larger error when compared with a symmetric spheroid from the beginning.

Therefore, to minimize the number of cells for the metric calculation will be wiser than to try to find a metric for every cell. This selection of cells should be done in such a way that, the cells that produce the largest error should be eliminated. Also, some tweaks on the metric extraction should be applied to ensure robustness and to eliminate unnecessary calculations. In this section these improvements are explained.

4.5.1 Skeletonisation for Natural Metric Extraction

In the very beginning of the chapter, while discussing about determining the layout of the geometry, the medial axis transform of the geometry is considered as the first option. The skeleton contains various information including the distance to the closest wall, which is approximately equal to the length of the smallest axis length of the natural metric. Actually, what is searched for is the information in other axes, namely the anisotropic sizes.

Actually, the natural metric that is defined in this chapter is an information defined strictly over the thickness. In order to have a successful metric, the bubble should grow at least to fill two opposing walls. Therefore, the foundation of the method shows strong similarity with the medial axis transform.

Then, the idea of calculating the natural metric only on the medial axis of the shape becomes an option. This is a wise idea since

- Every point which is not on the medial axis can be associated to a unique point on the medial. This is quite an obvious operation. The medial axis inside a geometry is similar to the Voronoï diagram of the surface, which is analogous to the Voronoï graph of a point set. In a Voronoï graph inside a volume, the Voronoï regions are no more closed surfaces (or volumes in three-dimensions), but a line¹ that lies on the axis of the maximal sphere to the closest walls. Therefore, the metric defined at a point inside the geometry is associated to the metric of a point on the medial, closest point to the surface of the geometry of which is equal to the that of the point in the space.
- Using the medial axis, the cells that produce the biggest errors are eliminated and natural metrics defined at these points are replaced with the metrics from the highest quality bubbles from central nodes.
- The cell count is reduced by 3-20 times, that yields an order of speed-up, while still having the metric field at every point.

¹This definition is true for non-cylindrical and non-spherical cases, where the Voronoï region can be shaped not as a line but as a surface. However, the same conclusion can be drawn in this condition.

It is decided to calculate the metric on the *skeleton* of the mesh, which is a simplified medial axis of the geometry. Since the algorithm is built on a background mesh, the analytical description of the skeleton is not necessary, but the cells that are intersecting the skeleton are sufficient. This leads to the definition of the *digital skeleton*, which is the skeleton of the geometry that is made up for central cells of the geometry.

Although it is difficult to build a good medial axis transform algorithm, the digital versions of them are simpler. During this study, two different skeletonisation algorithms are developed. For this part a thinning based skeletonisation procedure is employed. The method is explained in Appendix Chapter A.

The skeleton of a simple geometry is given in Figure 4.8. The thinning type digital medial methods have certain limitations, however, our algorithm is quite succesful in moderately complex geometries.

4.5.2 Other Improvements

A couple of more improvements are implemented in order to ensure the robustness as well as to improve computational efficiency. These steps can be summarized as:

Improvements to speed-up the process

- The metric calculation does not start until the bubble reaches to the boundary. This leads to a good speed-up especially after the skeletonisation, since many bubble growth operations may be necessary to reach the boundary. The metric calculation, which is a very costly operation, is unnecessary at these steps.
- The metric calculations applied on the cells at the bubble surface are tried to be re-used every time step, if the centre of the bubble $\mathcal{C}_k$ does not move in an significant amount. This saves many calculations given in equations (4.8) and (4.15). Unfortunately, this cannot be extended for the evaluation of the error given in (4.18), since the metric size can change significantly, and the cell face with the maximum error can be on the surface mesh. Indeed, one expects the maximum error due to the minimum edge length on the central part of the surface mesh, especially after skeletonisation.

Improvements to ensure the robustness

- The skeletonisation algorithm used is very fast, however, it is not so accurate. The algorithm may mark non-central cells as in skeleton, which in the end results in a cell with bad metric. To prevent this, a constraint on acceptance of the metric is set. At least two consecutive successful bubbles, interpolation error of which is less than the tolerance, are set to be necessary to ensure that the cell is a central cell.

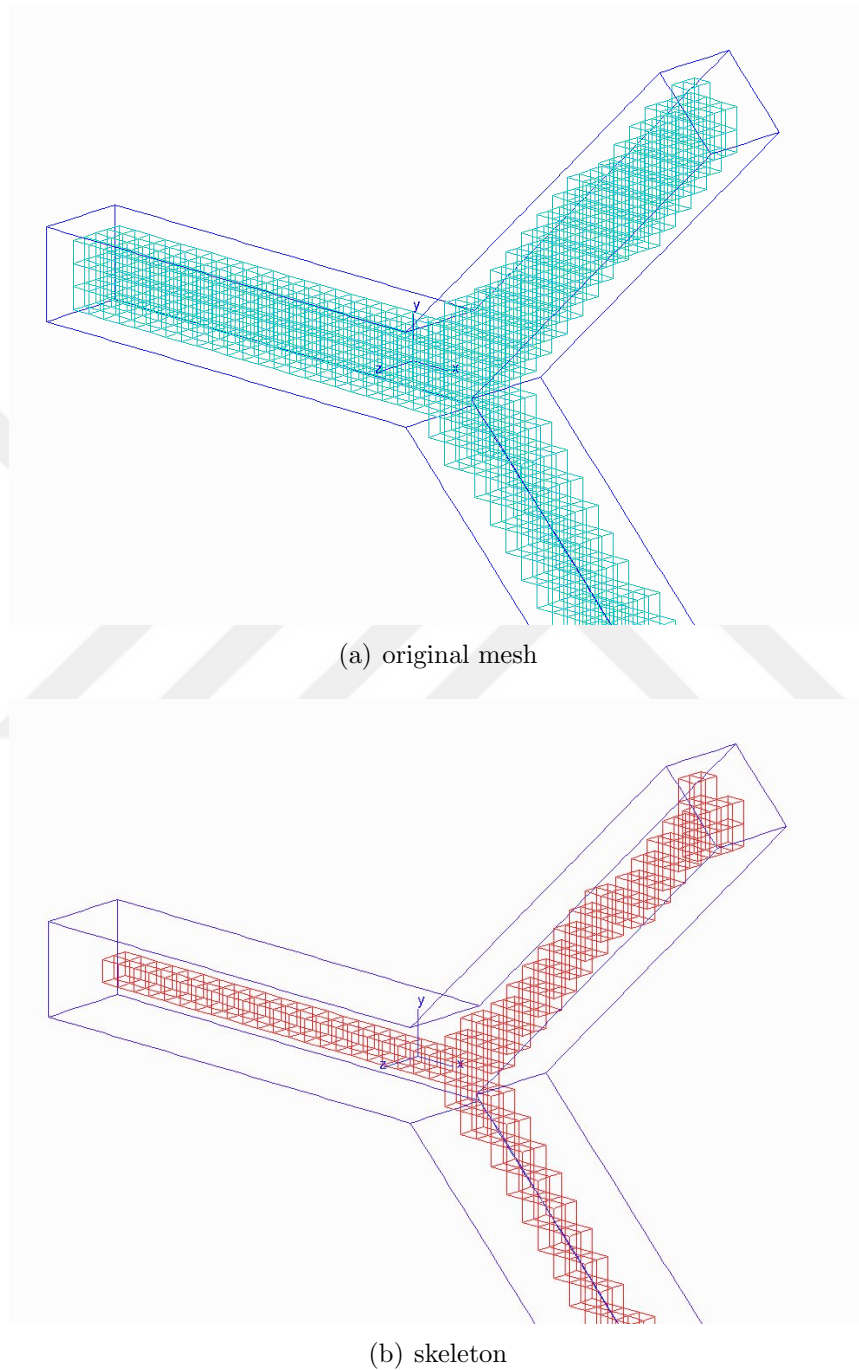


Figure 4.8: Skeletonisation process for natural metric extraction

Else, the cell is accepted as a non-skeleton cell and the metric calculated for it, is not used. This limit can be extended to an higher threshold value; however, one needs much refined background mesh for this purpose, and this requires much more

computational resources.

- An extensive monitoring is set for the control of the centre of the bubble. If the bubble centre $\mathcal{C}_k$ exits the periphery of the initial cell, the calculation of the metric is stopped. This periphery includes all neighboring cells of the initial cell in second degree. The reason for this is simple; if the centre is moving away from the cell, the bubble has probably hit to the wall, and cannot move in that direction. In this case, next to this cell, there should be another skeleton cell, which contains all the bubble cells of the skeleton cell into consideration. So, it is not necessary to calculate this cell, and one can stop the calculation.

Another possibility is that, the cell in the consideration is not a real skeleton cell, but a cell very close to the boundary. In this case the bubble centre immediately exits the cell periphery. Therefore, one does not need to perform all metric calculations to stop the metric calculation.

Actually, this operation is also very effective to fulfil the speed-up task.

- For practical reasons, if the channel is too long, one can stop the bubble growth after a certain number of successful bubbles, without waiting for failure of the elliptical interpolation. Although it seems like an operation to speed-up the process, it is essentially an action to ensure the robustness of the next step, which is the application of the metric in the mesh control process. In the mesh control algorithm highly anisotropic metrics may not be applied as will be explained in the following chapters, and a control over the metric anisotropy should be set.

The first idea is to control the ratio of the minimum and maximum sizes of the metric. However, this involves computation of the eigensystem of the metric for every iteration, which is a very computationally costly operation. Instead, given a cell on the skeleton, one can estimate this parameter from the number of successive bubble growth operations held after the bubble touches the wall. Note that, for a skeleton cell, when the bubble hits the boundary, the metric is (almost) isotropic.

4.5.3 The Final Algorithm for the Bubble Growth

Using the robustness enhancements and the speed-up improvements, the final algorithm becomes as such:

1. Start with skeletonisation to reduce the number of cells. For this purpose, use Algorithm A.2 given in Appendix A
2. Start with a cell $\mathcal{S}$ on the skeleton
3. Apply the Algorithm 4.2 to expand the bubble around this cell, until the bubble hits the boundary.
4. Start storing the faces of the bubble at the surface of the geometry to reuse them.

5. Expand the bubble with Algorithm 4.2.
6. For the new bubble
 - (a) calculate the centre of the bubble
 - (b) calculate the metric $\mathcal{M}$ with Algorithm 4.3, lines 3-9.
 - (c) calculate the scale factor c and scale the vector with Algorithm 4.3, lines 10-14
 - (d) calculate the error with Algorithm 4.3, lines 14-18.
7. Mark the cells at the boundary to avoid the recalculation.
8. If
 - (a) the interpolation error is below tolerance AND the metric extraction did not fail consecutively
 - (b) maximum anisotropy ratio was not met
 - (c) bubble centre is still at the near vicinity of the initial cell,

then, repeat the bubble growth steps.
9. Else if,
 - (a) A metric within the acceptable interpolation error is calculated.
 - (b) bubble centre is still at the near vicinity of the initial cell,

then, save the metric for the cell $\mathcal{S}$
10. Repeat procedure 2-9 for every cell on the skeleton.

4.6 Results

After formulating the problem, developing the algorithm and fine-tuning it for the hexahedral meshes, the final product should be tested on various geometries. An extensive geometry database is created for this purpose. These geometries are selected or created to cover many different geometrical combinations that possibly contain different problems for metric extraction. Unfortunately, it is not possible to review all of them; instead a simple and a complex geometry is presented here. The experience obtained from all those examples is explained briefly in Section 4.7.

The first test case that is presented in this study is a very simple one that contains single bifurcation, as it is shown in Figure 4.9. The skeleton of the mesh is also shown on the

geometry. In the first sub-figure, the natural metrics are displayed as stretched cubes that encapsulate the spheroids associated to the corresponding metrics. As it is seen here, the geometry and the directions are captured very well. Note that, the growth of the metrics is limited to maximum anisotropic ratio of 10 in this example. The union spheroids that are fit into the voids are also shown in the second sub-figure. One can notice that, some skeleton cells have no spheroids, indeed, these are cells located on the boundary of the mesh, left in the skeleton erroneously. There are no metrics associated to those cells, which verifies the selection of the error limit determined in Section 4.4.2. However, all of the volume is covered with cuboids, in other words, one can associate a metric everywhere in the volume. This shows that, the decision of using only the skeleton was right.

The second geometry is a very popular test case in computational geometry area that can be found in public domain web site [123]. The test case is three-dimensionally scanned image of a toy dinosaur. The original data is scanned from a real toy, so that it contains many irregularities on the surface. The geometry is shown in Figure 4.10 (b). Similar to the previous test case, the metrics extracted for this geometry are shown in Figure 4.10 (b) and (c). As it is seen here, although the metric directions are good in general, the details are not captured. This is due to the highly irregular surface shape, which creates a lot of disturbance during interpolations. Locations like the tail and ribs contain a lot of such details. The background mesh contains 19434 cells, which is too coarse for this geometry; however, the metric extraction time on the skeleton of 3354 cells requires 12 minutes on the 2.2 GHz Pentium machine. A really detailed mesh gives slightly better results; however, the computational time on the same computer increases to 11 hours. It is seen here that, the method cannot be applied robustly on an arbitrary geometry.

4.7 Conclusion

Based on the experiences gained on extensive testing of the Natural metric extraction procedure on different geometries, the evaluation of the method in terms of robustness and efficiency is presented here.

The first observation is, the method has an extremely high computational cost. This is not unexpected, since the tetrahedral natural metrics extraction algorithm is reported to be very slow. Knowing this fact, the algorithm developed in this study is optimized for speed from the beginning. The improvements on the algorithm to achieve this goal are described in section 4.5. The results of these improvements are very good. Although it is not possible to compare the two algorithms due to their different background mesh types and different test geometries, it seems that in certain conditions the new algorithm is almost an order faster than the original study.

However, time required for the analysis of a real world geometry is still tremendous. It should be noted here that Gruau used this algorithm for simulation of the moulding processes, for thin geometries. In thinner geometries the bubble growth is less costly than

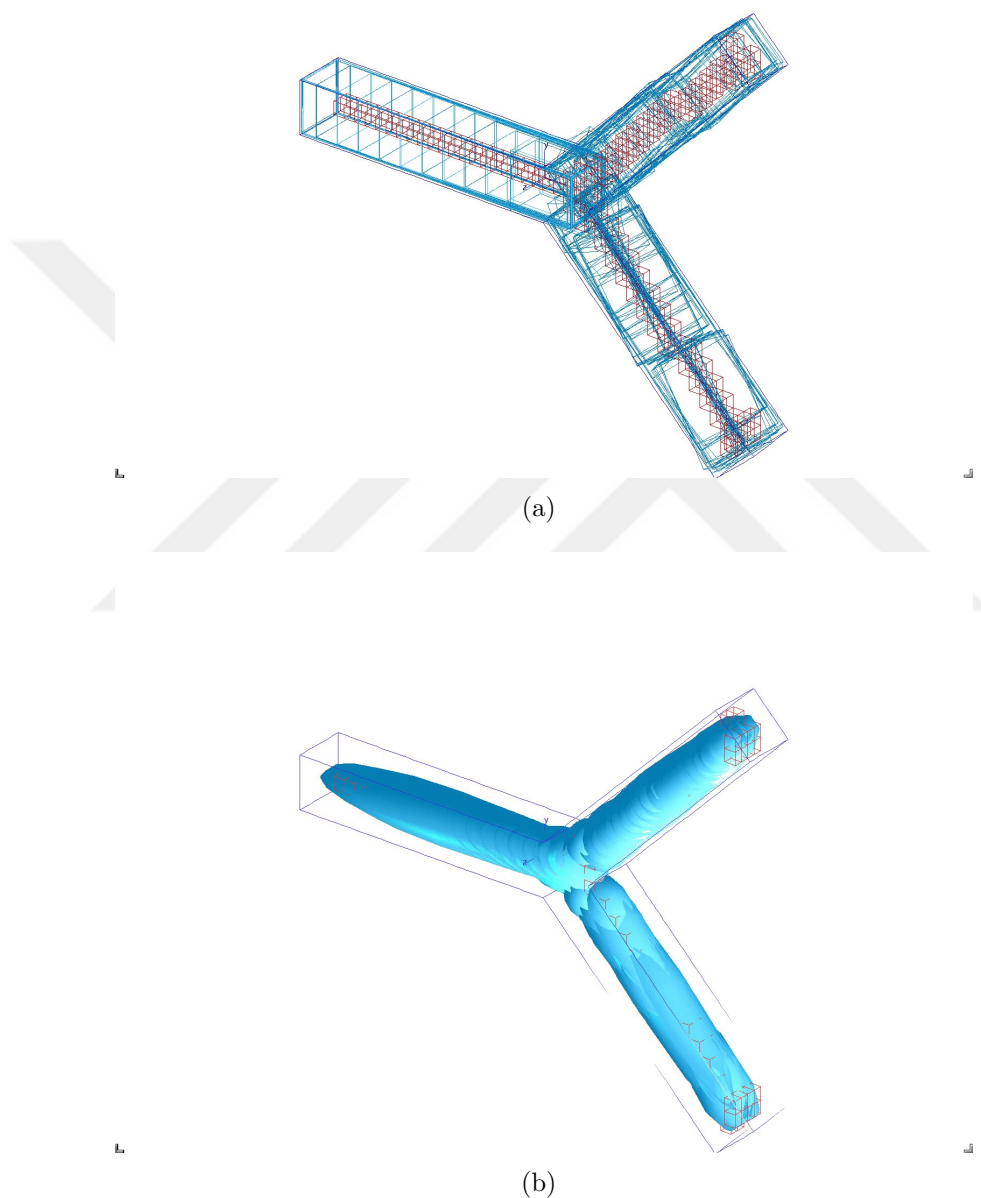


Figure 4.9: Basic test case for Natural metrics extraction

a thicker one, especially when the bubble is let to grow up to a certain anisotropic size. The biggest number of cells that is presented in his thesis is 96033 and he remarked that the 100000 cell meshes are their objective, since the most complex moulding cases are around this size. However, for our point of view, this cell count is considered as small to mid-

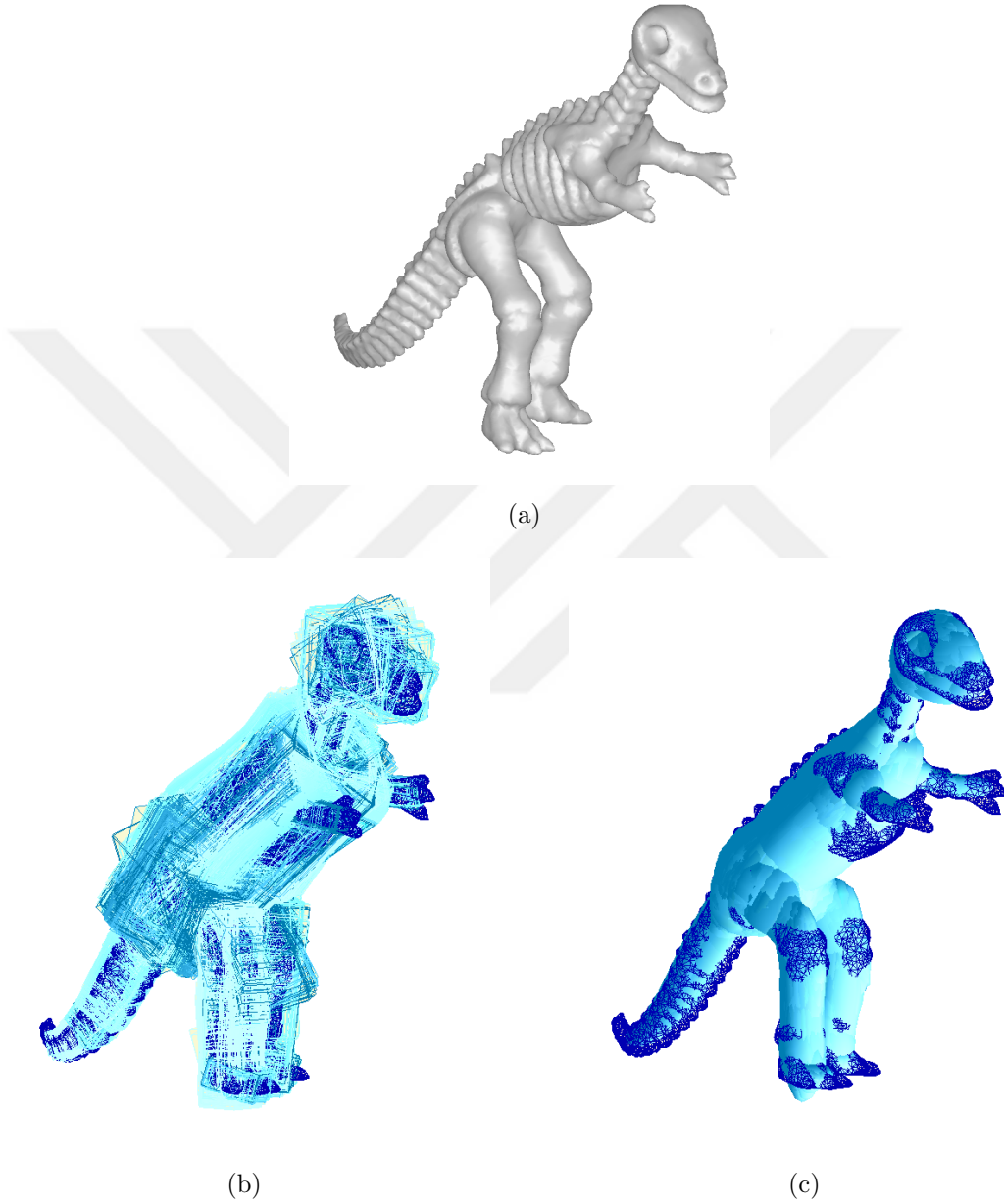


Figure 4.10: Toy dinosaur test case for natural metrics extraction

sized mesh. In computational fluid dynamics problems for example, the cell count easily goes beyond tens of millions of cells. Therefore, the computational cost is an important bottleneck for the scope of this study. The option of parallelization is also considered and planned for future studies. It is a rather trivial task, if the mesh is not partitioned.

For the robustness of the method, one can speculate that the algorithm produces good enough metrics for thin sections. Unfortunately, generalization of the method for arbitrary

geometries is not that successful. One of the main groups of geometries that the method fails is the external aerodynamics test cases. In these kind of geometries, a solid body is located in the middle of a much bigger domain. The calculation domain is so large that, the solid geometry becomes a small detail; such that it does not even disturb the elliptic interpolation. Another example can be made as, if the geometry has channel-like parts, the metrics can be successfully extracted. However, if the channel is conic, then the metric may never be extracted for these locations.

It seems that, the surface complexity is also a very negative factor. For example the toy dinosaurs test case has a lot of impurities on the boundary, which disturb the metric extraction. As it is seen in that example, there are regions without metric, which is not acceptable for our purposes. In order to extract natural metric on these kinds of geometries, some detail suppression operations on *dirty* geometries may be useful.

The bubble growth algorithm is a very smart tool to detect the natural anisotropy of the geometry. However, it is seen that the algorithm is useful only for thinner, more regular and cleaner geometries. As a generalized layout detection tool, it is too costly and unrobust to be used in an all-automatic hexahedral meshing tool, which is expected to mesh every geometry in a reasonable time.

Therefore, in the following chapters, new metrics, which require less computational resources while still providing good information about the geometry will be presented. The natural metrics as the descriptor of the anisotropy applied in the meshing process of thinner objects is given in examples section.

Chapter 5

Distance Map Metric

In the previous chapter, the problem of detection of the natural layout of a geometry is discussed. This information is crucial for an effective mesh generation control procedure. To make the natural layout data that contains local anisotropy information available, a metric called natural metric is introduced. The idea is based on expanding an elastic ball starting from the skeleton of an object, while checking if the ball keeps an almost elliptical shape. Therefore, the elliptic metric that fits to this almost elliptic bubble contains local anisotropy information, which is needed for the mesh generation control process.

Unfortunately, it is seen that, natural metrics algorithm is not efficient to be applied on real world geometries. The reason for this is various. First, the bubble growth algorithm that relies on elliptic metric fitting is extremely slow. Although, many tricks to speed-up this process are implemented and the final algorithm developed for this study is one order faster than that of original study, the computational cost is still far too expensive to be accepted for industrial meshing applications. Therefore, an alternative metric to the natural metrics, which also defines the natural layout of the geometry and which is also computationally more efficient is a must.

The second problem is, natural metrics extraction procedure is not robust to be used for general geometries. The method is applicable for thin geometries with rather parallel surfaces. For more complex geometries, the metric field cannot be calculated at some sections of the geometry. This is unacceptable for the application of point relocation algorithm. Therefore, the new metric defined in this chapter should be general enough to be applied for geometries of all complexity.

In this chapter, a new metric, which is a function of the distance to the nearest boundary of the domain, is introduced. This distance information that is defined on a background mesh is called distance map in computer graphics. Since distance map contains the data of distance to the nearest wall inside a volume, they give an indirect information about the local size. The size information requires process of the distance map data. Later in this chapter, it will be shown that, the anisotropy data, including both size and direction information, can also be extracted from the distance maps with a proper processing.

A distance map also contains all necessary information for extracting the skeleton of the geometry. Indeed, skeletonisation via distance maps is so efficient that, the thinning type skeletonisation algorithm is changed with a distance-map based variant.

In the forthcoming sections, the definition and properties of the distance maps will be given. Then, a new Riemannian metric based on the Hessian of the distance map data will be introduced. This section will be followed by discussion on the method, presentation of test results and a conclusion.

5.1 Distance Maps

A very good representation of the shape can be obtained by using the well-known distance maps. Distance maps are widely used in many areas of computer graphics, image recognition, and others. A distance map is a function defined in each cell of a mesh based on the distance to the closest point on the boundary of an object [124]. In real life applications, the distance map is calculated on a background grid that encompasses the geometry, and this information is used while processing the geometry.

The user is free to choose the method of the distance calculation, so called distance function, on this background grid. The possibilities vary from a digital distance maps, as it is used in the image recognition, to calculating the exact distance to the wall. The selection of this function is related to the requirements of the problem.

For a demonstration of the idea in Figure 5.1, a simple digital distance map using the so-called city-block type distance function is presented. This is a very common representation in digital image processing. In this representation of the distance map, the minimum distance to the wall appears at the intersecting pixels (or cells of background mesh). The distance values at these cells are set to one, while others are marked as unprocessed as it is seen in the sub-figure (a). Then, starting from intersecting cells, the neighbouring unmarked cells are set by increasing the distance value by one at a time. This neighbourhood marking process is applied by propagating layers of cells by increasing the distance value by one for each layer until no unmarked pixels are left. The resulting distance map is shown in Figure 5.1.(b).

Although the distance calculation is inexact integer values, the application is still useful. For example in the bottom image (c) of the Figure 5.1, the distance map calculated around a simple shape, with such a method is shown. In this figure, the thickness of the figure can be identified from the colour values of the centreline, which can be detected easily only by following the colour variation. Also, from the overall colour variation figure, it is easy to guess the orientation of the image.

In order to detect the features of the geometry, the calculation of exact distance to wall function yields better results. However, in this study it is seen that, even this coarse information may be very useful and both methods will be used.

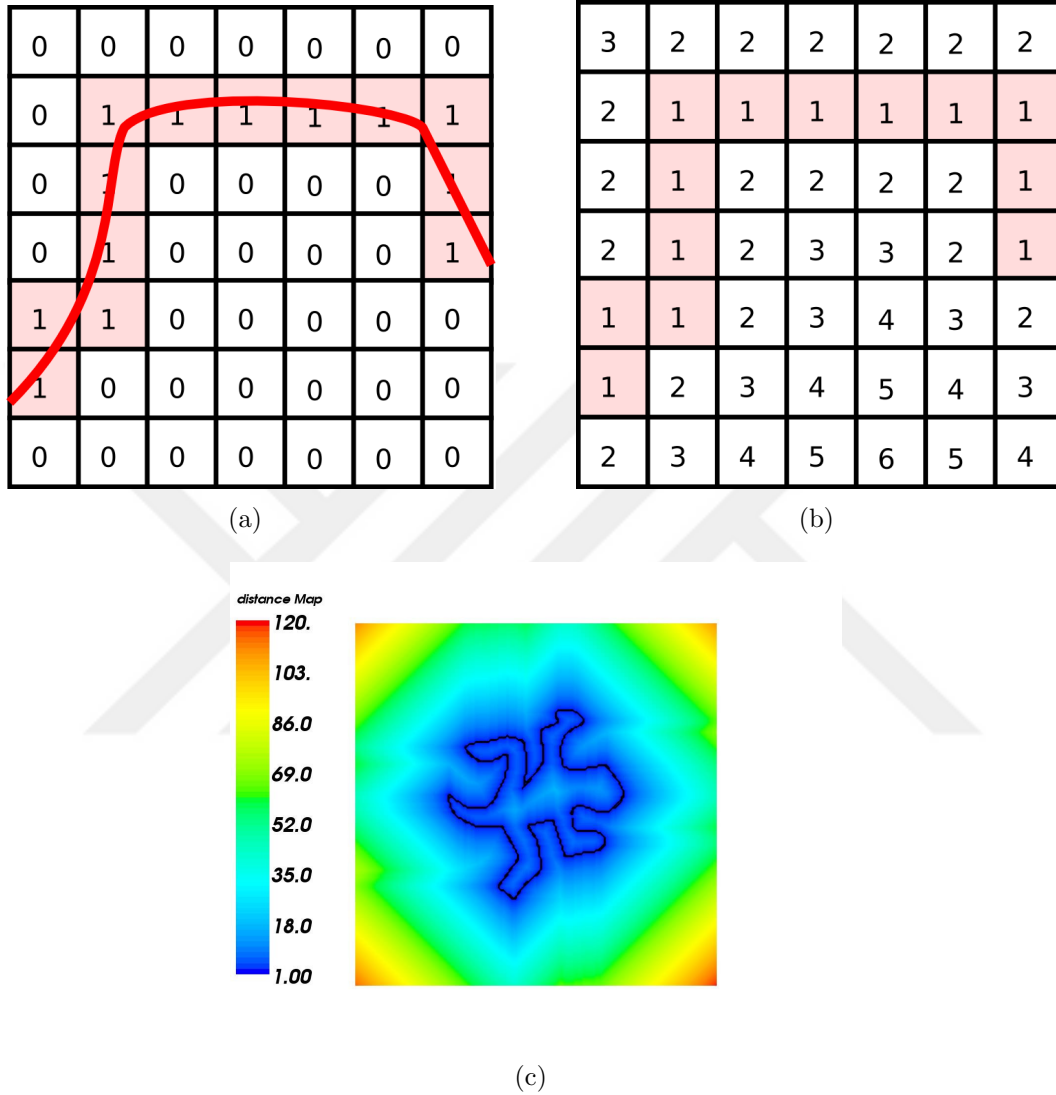


Figure 5.1: Distance map calculated by city block type distance function. On (a), cells intersecting the geometry are marked as distance =1, then on (b), the distance value is evaluated progressively through neighbouring cells. At (c) is the digital distance map around a boundary.

5.1.1 Properties of the distance function

Let's consider a distance map, where on a background mesh the distances to the nearest boundary and calculated. The boundary is shown in Figure 5.2 in red colour and this distance data is plotted as equi-distance surfaces. These surfaces are locally parallel to each other. One can consider these parallel surfaces as a propagating front. The primary

direction of change of distance, or in other words the direction of the propagation of the front is always in the direction to the (local) normal to the surface, provided that the front is propagating with a constant speed. For such a condition, the secondary axes are tangent to the line and orthogonal to the primary axis¹.

Therefore, distance map contains not only a hint about the local size, but also sufficient data about the local directions. These two data are adequate to define a metric, that contains information on local anisotropy. Still, how to extract these two parameters from a scalar data defined at the background grid is not clear. The solution to this problem will be given later in this chapter.

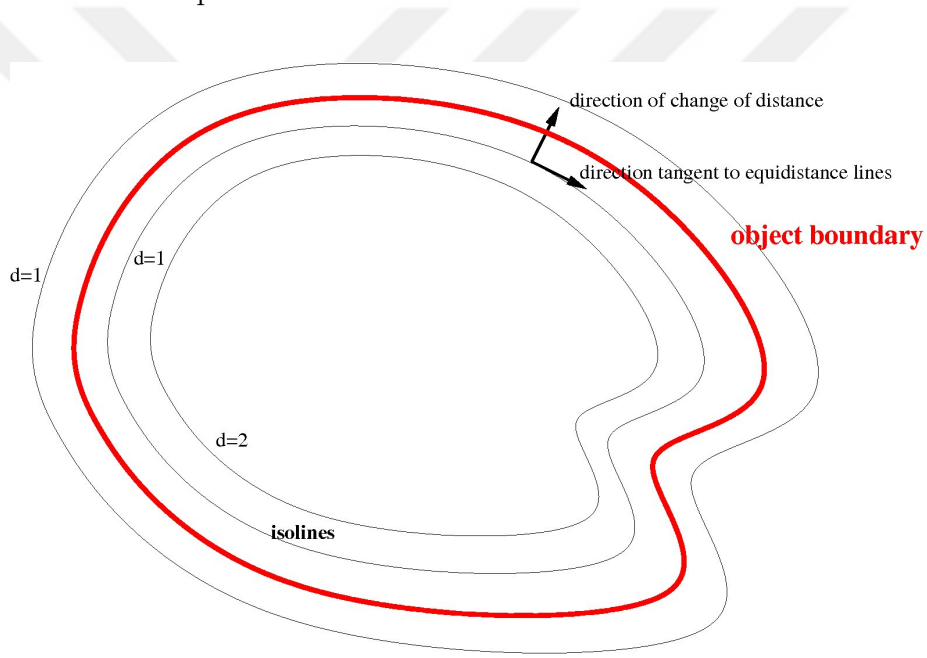


Figure 5.2: The equi-distance lines and metric definition

The iso-distance lines calculated for the distance map of the same figure given in Figure 5.1.(c) is presented in Figure 5.3. As it is seen from this figure, the propagation direction can be easily followed. A very interesting feature of this map is that, if this propagation is followed towards inside of the geometry, it is seen that the propagation front is collapsed by another front moving towards the opposing wall. Provided that propagation speed is the same² at any time of the propagation, let's draw a line connecting point, where the fronts collapse, as it is done in the same figure with red line. Since the fronts are propagating with the same speed, this line passes through the centre line of the shape, which is analogous to the medial axis of the shape.

¹Actually, the distance function should be also continuous, in other terms the front propagation shouldn't collapse. However, this is locally true and discontinuity is an indication of the medial axis.

²Not necessarily constant.

Therefore the distance maps also contain information of the medial axis. The front propagation analogy reminds the skeletonisation by thinning procedure. Indeed, many thinning procedures in the literature rely on a background distance map. However, if an accurate distance function is calculated, then the skeleton can also be calculated by only observing the local variation of the function.

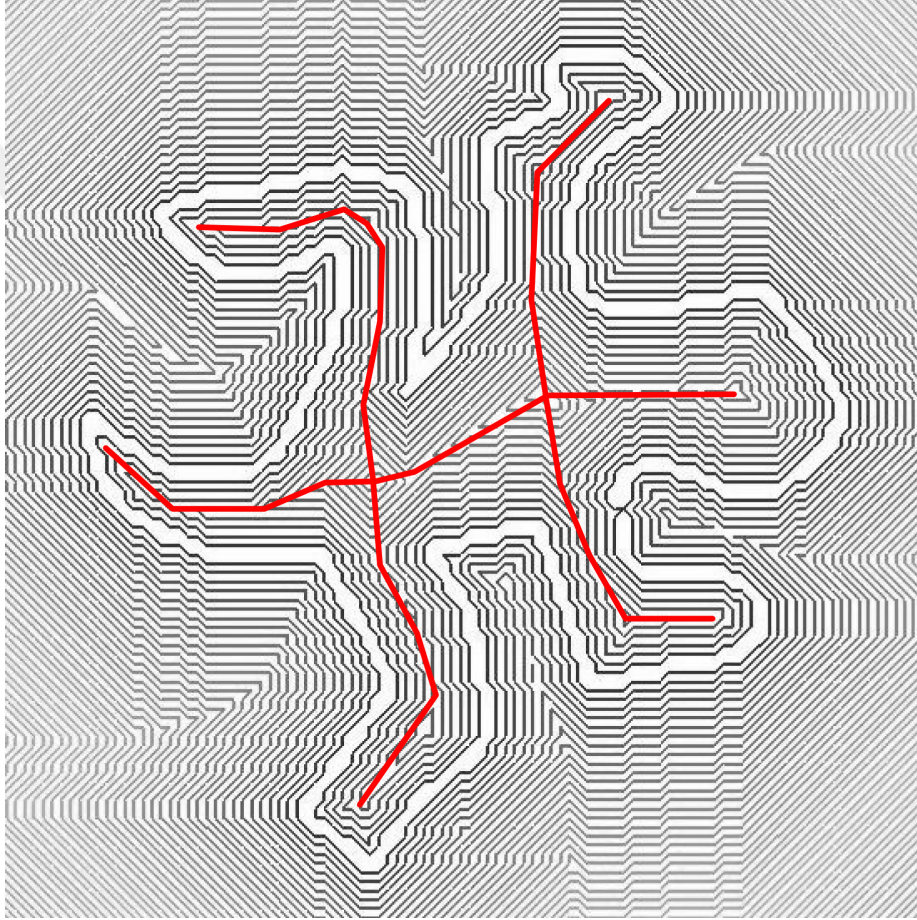


Figure 5.3: Medial axis from distance maps

In the preceding chapters, the properties of the medial axis as well as the metric based mesh control methods were already discussed. In Chapter 4, a metric, which is suitable for mesh control applications is presented. This metric is defined on a background mesh and also utilises the medial axis data. One can conclude that, the size and direction information available in the distance-map data can also be used to define a similar metric. Using the distance map, the medial axis, which is an invaluable tool as the shape descriptor, can also be computed easily.

In the following sections, necessary tools to extract these precious data hidden in the

distance maps will be presented, and using these data, new metrics to be used in the mesh control will be presented.

5.2 Hessian Matrix

The Hessian matrix is named after the German mathematician Ludwig Otto Hesse. The concept is introduced for functional determinants. The Hessian of a scalar function of a vector is the matrix of partial second derivatives. The Hessian matrix of a function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is:

$$\mathcal{H}(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1} & \frac{\partial^2 f}{\partial x_d \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_d^2} \end{bmatrix} \quad (5.1)$$

In other words, Hessian is the Jacobian matrix of the derivatives $\partial f / \partial x_1, \partial f / \partial x_2, \dots, \partial f / \partial x_d$ of the function $f(x_1, x_2, \dots, x_d)$. Note that, when f has continuous second derivatives, the Hessian is symmetric.

5.2.1 Relations Between Hessian Matrix and Local Extrema

Hessian³ is a very useful tool in differential analysis. It is mainly used to analyse the variation of functions, or scattered data. For example, for one-dimensional case, the Hessian reduces to the well-known second derivative test, which is given as follows:

Let the neighbourhood of x be in the domain for f , and let f have continuous partial derivatives of first and second order. Let $f'(x) = 0$. If $\mathcal{H}(x) = f''(x) > 0$, then x is a strict local minimum. Else, x is a strict local maximum.

This procedure can be extended to higher dimensions. Let x be a vector, and let $\mathcal{H}(x)$ be the Hessian for f at a point x . Let the neighbourhood of x be in the domain for f , and let f have continuous partial derivatives of first and second order and $\nabla f = \vec{0}$. Therefore following conditions are given:

- Extending the analogy for the second derivative test, local extrema can be detected. If $\mathcal{H}(x)$ is positive-definite, then x is a strict local minimum for f . Similarly, if $\mathcal{H}(x)$ is negative-definite, then x is a strict local maximum for f .

³In literature, the term Hessian is both used for the Hessian matrix and the determinant of the Hessian matrix. In this thesis, Hessian stands for the original matrix form.

- The reverse analogy is also valid. If x is a local minimum, then $\mathcal{H}(x)$ is positive semidefinite and if x is a local maximum, then $\mathcal{H}(x)$ is negative semidefinite.
- As a last property, if $\mathcal{H}(x)$ is indefinite, then x is a non-degenerate saddle point.

Therefore, the variation of any continuous function can be determined by analysing the Hessian of that function.

5.2.2 From Hessian Matrix to Riemannian Metric

Suppose we have a continuous scalar function f as it is defined before. Then Hessian of the function given in equation 5.1 is redefined as:

$$\mathcal{H}(x) = R\Lambda_{\mathcal{H}}R^T \quad (5.2)$$

where $\Lambda_{\mathcal{H}}$ is the diagonal matrix of eigenvalues of $\mathcal{H}$ and R is the matrix of eigenvectors. Since $\mathcal{H}$ is a symmetric matrix, its eigenvalues, Λ are real, and its eigenvectors are orthogonal as it is in Riemannian metric. Therefore, one can observe the similarity between Hessian of a continuous function and the Riemannian metric. The basic difference is that, the Riemannian metric is positive definite, whereas the Hessian is not necessarily positive definite. Therefore, the eigenvalues can be negative. The metric edge sizes are given in the previous chapter as:

$$\lambda_i = \frac{1}{h_i^2} \quad (5.3)$$

Accordingly, negative eigenvalues, or complex anisotropic size information, implies a self-folding geometry, a non-manifold. Obviously, this kind of mesh manipulating tool cannot be used. This problem can be easily solved by using the absolute value of the eigenvalue matrix Λ .

$$\mathcal{M} = R|\Lambda_{\mathcal{H}}|R^T \quad (5.4)$$

5.3 Distance Map Metric

In the previous section, the Hessian concept is introduced. A Hessian matrix of a function consists of Jacobian of the second derivatives. It serves as multi-dimensional second derivative test, and it is mainly used for analysis of the variation of a function. It is also seen that there is a strong similarity between the Hessian of a continuous function and a metric. If the eigenvalues of the Hessian matrix are guaranteed to be positive, then the resulting matrix can be used as Riemannian metric.

This is an interesting feature of the Hessian matrices. The eigensystem of the Hessian matrix, therefore, shows nothing but primary directions and the rate of change in those directions of this function. In other words, the largest change is in the direction of the eigenvector of the largest eigenvalue.

Of course, this information can only be used in meshing purposes, if these directions are meaningful for the geometry. In section 5.1.1, properties of the distance function are discussed. Let's recall the isolines of the distance map given in Figure 5.2. Then following properties of the distance maps can be distinguished:

1. The distance map is always parallel to the surface for a continuous curve. Therefore, the rate of change is the minimum in the direction of tangent to the surface.
2. The maximum rate of change is in the direction normal to the surface.
3. The function is continuous except on the medial axis.

It is discussed before, these directions as well as the medial axis itself would be very precious information for mesh control, and a method that will be utilised to extract them is needed. Apparently, Hessian of the distance function can provide this information.

There is only one problem, that the distance to the nearest wall function is linear, and its second derivative is meaningless. However, as it is indicated before, one is free to choose the distance function. Therefore, the modified distance function δ^2 is introduced, which is a higher order variant of the distance function that produces non-zero second derivatives. In this study, several of them are used. But as a basic variant square of the exact distance to the wall value is selected.

Therefore using the Hessian of the distance function, the primary direction can be identified. The primary direction is similar to that of the Natural metric, which is actually direction to the closest medial axis. The other two axes are directions of the local curvature change and they are very useful too. Unfortunately, neither the eigenvalues, nor ratios of them have a sensible meaning for the mesh size control.

Therefore, the anisotropic sizes of distance map metric should be controlled explicitly, while anisotropic directions can be obtained from the Hessian. Let's define the Hessian of the distance map as such:

$$\mathcal{H}(\delta^2) = R_{\delta^2} \Lambda R_{\delta^2}^T \quad (5.5)$$

where, R_{δ^2} is the rotation matrix for the Hessian of distance function. Therefore the new metric, named Distance Map metric is defined such as:

$$\mathcal{D} = R [\Lambda(\delta^2)] R^T \quad (5.6)$$

where, $\Lambda(\delta^2)$ is diagonal matrix of the size function $\{1/h_1(\delta^2), \dots, 1/h_d(\delta^2)\}$, which is determined explicitly. This size information is semi-empirical and can be found from the distance map itself. In this study, mainly the data on the medial axis is used. Such size functions will be given in Chapter 6.

5.4 Application

In order to implement the ideas given above, distance maps should be calculated robustly for the geometry, and then Hessian and the skeleton should be computed. It is seen that, these calculations take a great amount of computer resources, therefore, most efficient computational methods are aimed at. In order to simplify the calculations, as well as to save the computer memory requirements, an octree mesh on Cartesian axes is utilised. This background mesh is significantly lighter than the original data structure of HEXPRESS, and since it is full Cartesian, some geometric calculations like neighbourhood search or closest surface point search are much faster.

The details of the implementation are given below.

5.4.1 Computation of the Distance map

As it is discussed earlier, the distance map is calculated on a background grid. For this purpose an initial Cartesian grid is refined with isotropic refinement operations as it is explained in section 2.3.4, until details of the geometry is captured with these refinements. Mesh cells that are intersecting with the boundary are detected at this step. Then, starting from these intersecting cells, the distance to the wall from each cell centre is calculated by recursive layers. In order to save from the very expensive distance calculations, first triangle lists that are intersecting to the mesh cells are merged as triangle groups and each of them is associated to the cell that its triangles are intersecting with. Then the distance function for these intersecting cells is calculated. After that, the neighbouring cells are marked. For each cell in this layer, the triangle groups associated to the surrounding cells are marked as candidate triangles. The minimum distance is calculated between these triangles and the cell centre. The triangle group that contains the closest point to the cell centre is associated to the cell. This procedure is repeated by recursive layers of cells, until distance function for every cell in the background grid is computed. This procedure is given briefly in Algorithm 5.2.

The result of the distance map calculation on the toy dinosaur test case is given in figure 5.4. In this figure, recursive thinning of the shape can be observed easily. The distance function seen here is the square of the distance to the closest wall. The results are very good, and the calculation of the distance map in a mesh of $2 * 10^5$ cells requires less than one minute on a 1 GHz Pentium class computer. It should be noted that the model has

Algorithm 5.2 Distance map calculations**Require:** $\mathcal{T} \in \Omega$ **Require:** $\mathcal{T}_b \in \mathcal{T}$, cells intersecting the boundary

```

1: for all  $\mathcal{S}_i^b \in \mathcal{T}^b$  do  $\triangleright$  for each cell intersecting the mesh boundary
2:    $\mathbf{T}_i = \mathcal{S}_i^b \cap \partial\Omega$   $\triangleright$  mark the triangles intersecting to the cell
3:    $\delta_i^2 = \delta(\mathbf{T}, \mathcal{S}^b)$   $\triangleright$  calculate the distance
4: end for
5: repeat
6:    $\mathcal{T}_b = \mathcal{N}(\mathcal{T}_b) - \mathcal{T}_b$   $\triangleright$  detect the neighbouring cells
7:   for all  $\mathcal{S}_j \in \mathcal{T}^b$  do  $\triangleright$  for each cell in the neighbouring layer
8:      $\mathcal{S}_N = \mathcal{N}(\mathcal{S}_j) \iff \mathbf{T}(\mathcal{S}_N) \neq \emptyset$   $\triangleright$  find neighbouring cells, that have triangle-list
9:      $\delta_j = \min(\delta(\mathbf{T}(\mathcal{S}_N), \mathcal{S}_j))$   $\triangleright$  find the minimum distance function
10:     $\mathbf{T}_j \leftarrow \delta_j$   $\triangleright$  assign the triangle set containing the smallest distance to the cell
11:   end for
12:    $\mathcal{T} = \mathcal{T} - \mathcal{T}_b$   $\triangleright$  mark the cells as processed
13: until  $\mathcal{T} \in \emptyset$   $\triangleright$  No cells to compute

```

only 27000 triangles and one should expect longer computational times for more detailed models.

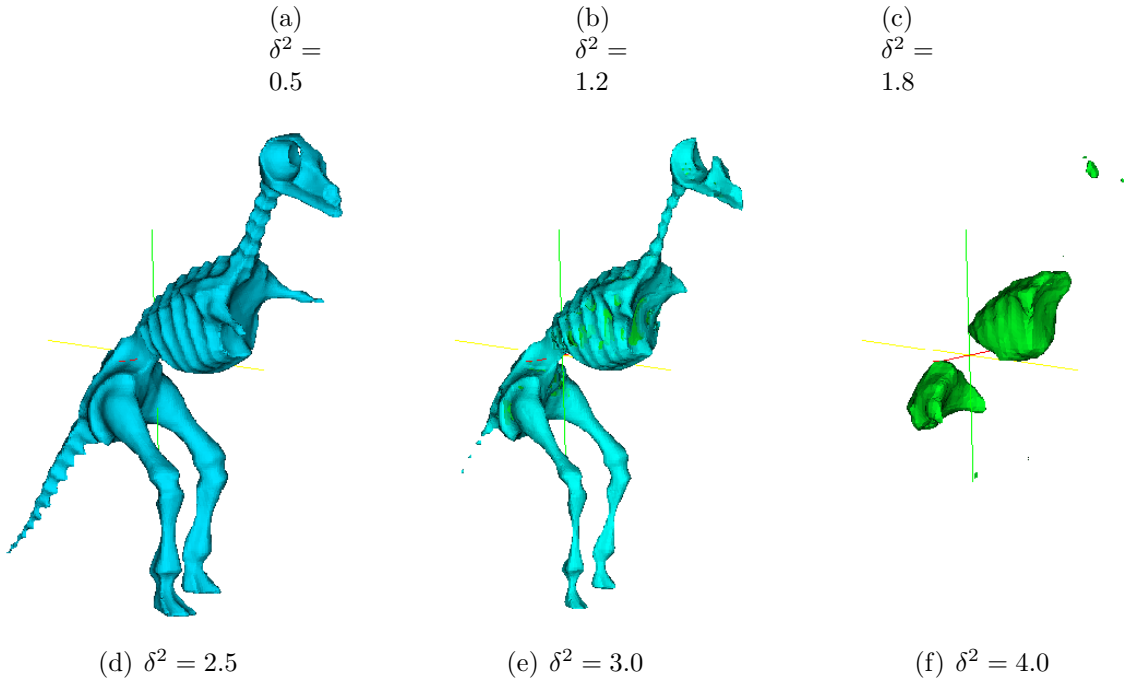


Figure 5.4: Equi-distance surfaces of δ^2 from the distance map around the toy dinosaur geometry

The city-block type distance maps are also calculated in the framework of this study. The algorithm is derived from the skeletonisation algorithm used in Chapter 4 and given in Appendix A. This method is much faster than the exact distance calculations and requires a couple of seconds for the same mesh and computer configurations.

5.4.2 Computation of the Hessian

Regardless of the size function selection for anisotropic sizes, square of the exact distance to the wall is used for calculating the anisotropic directions. This function is cheaper to calculate, and can be made available on the distance map using the Algorithm 5.2. The distance function is second order and therefore its Hessian can be computed.

The numerical scheme used to compute the Hessian is based on standard second order quadrature rule for second derivatives as it is given in the books of Hirsch [125, 126]. The stencil is determined from the first order neighbourhood of the cell whose Hessian of the distance function is calculated on.

After computing all the derivatives, the Hessian matrix is formed and eigenvalues and eigenvectors of this matrix are computed. As it is stated before, one does not need the raw Hessian matrix. The Hessian is calculated for extracting the anisotropic directions; therefore only eigenvectors are needed to be kept. However, one should distinguish which anisotropic directions are associated to normal and tangential directions. The direction on the largest eigenvalue is the direction normal to the surface, since the largest rate of change of the distance function is observed in this direction. Therefore, the eigenvectors are sorted by the size of the corresponding eigenvalue. Therefore, in the updated rotation matrix the biggest eigenvector occupies the first column. This procedure is given in Algorithm 5.3.

Algorithm 5.3 Hessian of the distance function

Require: $\mathcal{T} \in \Omega$

Require: $\delta^2 \in \mathcal{T}$ cells intersecting the boundary

- 1: **for all** $\mathcal{S} \in \mathcal{T}^b$ **do** $\triangleright$ *for each cell in the mesh*
 - 2: $\delta_{stencil}^2, \Delta x_i$ $\triangleright$ *find the stencil*
 - 3: $\partial \delta^2 / \partial x_i \partial x_j \leftarrow \delta_{stencil}^2, \Delta x_i$ $\triangleright$ *calculate derivatives*
 - 4: $\mathcal{H} \begin{bmatrix} \ddots & \dots \\ \dots & \delta_{i,j}^2 & \vdots \\ & \dots & \ddots \end{bmatrix}$ $\triangleright$ *form Hessian*
 - 5: $R, \Lambda \leftarrow \mathcal{H}$ $\triangleright$ *calculate eigensystem*
 - 6: $r_i, r_2, \dots, r_d \leftarrow \lambda_1 > \lambda_2 > \dots > \lambda_d$ $\triangleright$ *sort eigenvectors*
 - 7: **end for**
-

For demonstration purposes, the biggest eigenvectors of the Hessian of distance map are

computed on a background mesh around a pipe-like geometry and it is shown in the Figure 5.5. As it is seen from this figure, the normal directions are captured very well. Other two eigenvectors are tangents to the surface, and the results are satisfactory. The computation cost of the Hessian computation is quite high, For a million cell-mesh, the process takes 288 seconds on a 1 GHz Pentium class computer. The reason for this high computational cost is the computation of derivatives and eigenvalues. However, this duration is reasonable, since the direction of anisotropy is computed once and stored to disk. The computed anisotropic directions therefore are reused later for different meshing purposes.

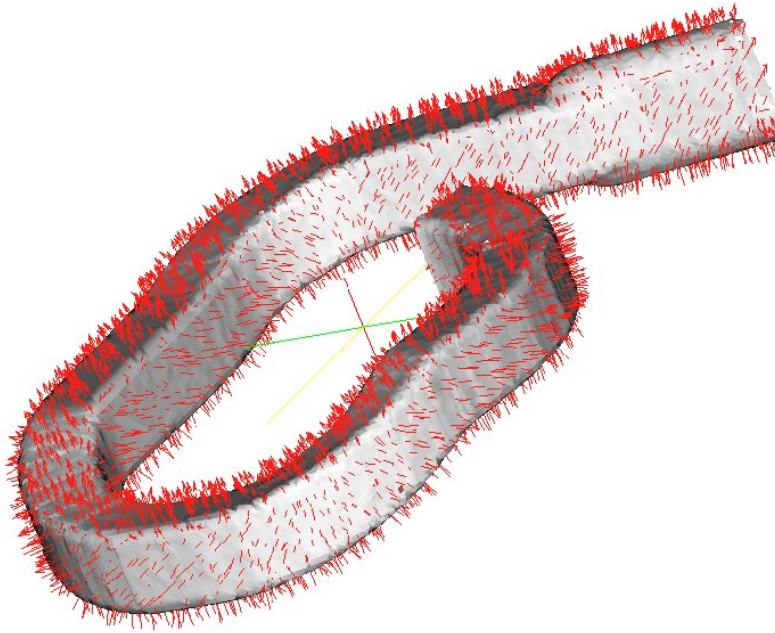


Figure 5.5: Eigenvectors of the largest eigenvalues from the Hessian of the distance map

5.4.3 Skeletonisation via distance maps

It is stated that, the size information will be taken from the skeleton of the geometry. A skeletonisation algorithm is developed for natural metric extraction given in previous

chapter and its algorithm is given in Appendix A. However, with this algorithm the aim is to decrease the number of cells to enhance the computational performance. This algorithm is not sufficient for the purpose of computing distance map metrics and should be replaced with a more accurate one.

It will be wiser if the already calculated distance map data is reused. From the figure 5.4, it is seen that thinning-based skeletonisation algorithm is an option, since the distance maps also imply propagating front. However, thinning based algorithms are known to be eroding the surface excessively and they should be constrained in order to get a high quality skeleton. Constrained algorithms, in turn, are complex and expensive.

Another option is processing the Hessian data to detect the abrupt change of local extrema. But, this method would be quite slow and inefficient.

Therefore, a more geometric method which is using the distance map data and easy to compute, is proposed. The new algorithm is based on checking whether the location of a cell would be at the centre of the locus of the maximal sphere. This algorithm is an expensive one and it is not used much in computer graphics. However, with the availability of the distance calculations that are already made at previous steps of the implementation, the new algorithm becomes very fast. The details of the new skeletonisation algorithm are given in Appendix B. As it is seen here, the medial is very high quality and the computational cost is significantly low.

5.5 Conclusion

The basic difficulty on efficient hexahedral meshing is to find a high quality and simple enough descriptor of the properties of the shape. As it is concerned before, in real-world applications, simulations are made on very complex shapes with significant local orientation and thickness changes. Thinner regions require extra refinement cycles, in order to fit the cell size to the feature size of these fine regions.

In order to solve this inefficiency, a series of studies employing different possibilities of shape descriptors are being examined. The previous chapter introduced such a descriptor, based on metric information extracted from geometry by expanding elastic balls inside the domain. This chapter covers another attempt to find such a metric that is based on the distance maps, which concentrates on the capturing feature size.

In this chapter, distance map concept is introduced. A distance map is the function of the distance to the closest wall defined on a background grid. It is shown that, a distance map contains many useful features, which can be exploited in mesh generation control.

The necessary tools to extract this useful data from the distance maps are also developed. It is shown that, using the Hessian of the distance function, the directions to the medial axis as well as the normal directions indicating the local parallel axes can be determined. Therefore, these directions, which are the eigenvectors of the Hessian, R_{δ^2} can be used to

define such a metric, that can be utilised in the anisotropic size control mechanism given in Chapter 3. The new metric is given as:

$$\mathcal{D} = R [\Lambda(\delta^2)] R^T \quad (5.7)$$

It should be noted that, the anisotropic size information is not available. Size information can be selected by user. The data on the distance map, as well as the skeleton are two good candidates for this purpose. Selected size functions for this study have not been given in this chapter, yet. Such functions will be presented in the forthcoming chapters dedicated to application of the mesh control algorithms.

Chapter 6

Mesh generation control

Necessary tools and theory developed for mesh generation control are presented in preceding chapters. In Chapter 3, a point relocation algorithm named spring analogy, which replaces mesh with a network of springs in tension is introduced as the basis of the mesh control algorithm. It is also shown that, the mesh can be controlled by adjusting stiffness values of the mesh springs. Mesh anisotropy can be imposed using Riemannian metrics as well, which is defined as a positive-definite symmetric matrix. Such metrics are presented in Chapters 4 and 5. The natural metric, which is given in the former of these two chapters, is based on expanding an elastic ball starting from medial axis of the geometry, and interpolating a similar shaped ellipsoid to the shape of the inflating ball. Although high quality metrics can be extracted for simpler geometries using this method, it is seen that robustness and computational requirements of the algorithm are not compatible to industrial standards.

In the previous chapter, the distance map metrics are presented. These metrics are based on numerical operations on a distance-function defined on a background grid that is called distance map. It is shown that largest eigenvector of the Hessian of the distance function is always in the direction to the closest medial axis. The other two eigenvectors defines an orthogonal plane to the largest one. Therefore, the matrix of the eigenvectors can be a good mesh control metric to be exploited in mesh control algorithm. Unfortunately, unlike the natural metrics, eigenvalues of the Hessian are not meaningful for mesh generation control, and cannot be used as a control mechanism of the anisotropic sizes.

In this chapter, tools that are developed as pieces of the mesh control methodology and that are presented in the previous chapters will be assembled into a complete mesh generation control system. To achieve this goal, the anisotropic size definition for metrics will be refined, and the correlation between spring analogy and the metrics will be introduced. This chapter is dedicated to the application and verification of the concepts developed in this thesis. The mesh control algorithm is applied in the mesh generator HEXPRESS and some examples for both thin geometries and the domain around the geometries under the influence of external aerodynamics will be presented. The resulted meshes will be compared with the grids generated by unmodified algorithm.

6.1 Application of the mesh generation control

6.1.1 Selection of metrics and stiffness functions

The only necessary step to exploit the metrics defined in Chapters 4 and 5 in the mesh control is to define a stiffness function that can be utilised with the spring analogy. This can easily be achieved by applying those metrics as a Riemannian distance measure and regularising the mesh in Riemannian space by modifying the stiffness function as given in equation (3.6).

It should be noted that, the metric should be continuous to be used in spring analogy. This may not be directly achieved due to the existence of discontinuity of the metric eigenvectors around the medial; however, our initial studies suggest that, at least the anisotropic size function should be continuous. Therefore, the metrics given in the previous chapters should be modified as such.

In this thesis, three stiffness functions are presented. The first two are metric based stiffness functions; whereas the third is a scalar function.

Distance Map Metric Based Stiffness

As it is stated earlier, eigenvalues of the Hessian of the distance function is not usable for mesh control; therefore the anisotropic sizes should be determined. In the previous chapters, it is concluded that the medial axis contains the local thickness information. The value of distance function at the medial, which is equal to the radius of the maximal tangent sphere inside the domain and associated to the medial, is an indicator of the local thickness. This size information can be exploited to modify the mesh. In the selected scheme, the maximum anisotropic size at any point in the domain is selected as the distance function at the closest medial. This selection is necessary to control global cell size distribution. The value of the distance function at the medial surface is smaller at the detailed sections of the geometry, hence, the sizes of the cells at those locations are compacted.

In order to impose anisotropic sizes, the size at the direction of the largest eigenvector should be compacted. This can be achieved by imposing the value of the distance function defined at the point as the edge size at the direction of the largest eigenvector. It should be noted that, provided that the point is inside the domain, the value of the distance function at a point is always smaller than or equal to distance function value of the point on the medial axis, where the point is associated to. However, it may not be enough to use this value to achieve a good anisotropy. The anisotropy is further imposed by modifying the value of the smallest eigenvector. This is achieved by multiplying the anisotropic size by a parameter called *factor of maximum anisotropy*. This factor implies the anisotropic size of the cell that is defined at medial and it is applied only on the cells that are located inside of the domain. For the cells outside of the domain, the cell sizes should be kept larger. Hence, the size at these directions are kept unmodified.

Therefore, for this study the distance map based metric is given as,

$$\mathcal{D} = \mathcal{M}_{\delta^2} = R_{\delta^2} \Lambda_{\delta^2} R_{\delta^2}^T \quad (6.1)$$

where, the eigenvalues, in other words the anisotropic sizes are given for a matrix of sorted eigenvectors as:

$$\lambda_{\delta^2} = \begin{cases} \left[\frac{1}{A^2} \cdot \frac{1}{\delta_D^2}, \frac{1}{\delta_{Medial}^2}, \frac{1}{\delta_{Medial}^2} \right] & \rightarrow \mathcal{D} \in \Omega \\ \left[\frac{1}{\delta_D^2}, \frac{1}{\delta_{Medial}^2}, \frac{1}{\delta_{Medial}^2} \right] & \rightarrow \mathcal{D} \notin \Omega \end{cases} \quad (6.2)$$

The δ_{Medial}^2 is defined as the distance map value at the closest medial point to the point into consideration. A^2 is the anisotropy factor to impose the anisotropy on the mesh. δ_D^2 is the value of the distance function at the point that the metric is defined.

Therefore, the stiffness function is given using equations (3.14) and (3.15) as,

$$k_I(x_J) = \frac{\int_0^1 \sqrt{(x_I - x_J)^T \mathcal{D} (x_I - x_J)}}{\|x_I - x_J\|} \quad (6.3)$$

Natural Metric Based Stiffness

As it is stated at Section 4.5, natural metrics are computed only at the medial. The rest of the mesh cells have no metrics associated to them and these metrics should be computed in order to perform mesh generation control process. The two largest anisotropic sizes that constitute the medial surface plane is used as unmodified in the metric that are used for computation of the spring stiffness. Similar to the distance map stiffness, the anisotropic size in the direction towards the wall is modified by the anisotropy factor to ensure more compact cell sizes in thickness. The metric size is also smoothed using the distance map function to ensure continuous metric size distribution.

Therefore, the natural metric used for the mesh control process is given as:

$$\mathcal{M}_N = R_N \Lambda_N R_N^T \quad (6.4)$$

where,

$$\lambda_N = \begin{cases} \left[\frac{1}{A^2} \frac{\delta_{Medial}^2}{\delta_D^2} \cdot \lambda_1^N, \lambda_2^N, \lambda_3^N \right] & \rightarrow \mathcal{M}_N \in \Omega \\ \left[\lambda_1^N, \lambda_2^N, \lambda_3^N \right] & \rightarrow \mathcal{M}_N \notin \Omega \end{cases} \quad (6.5)$$

Similarly, the spring stiffness for the natural metric based relocation algorithm is given as:

$$k_I(x_J) = \frac{\int_0^1 \sqrt{(x_I - x_J)^T \mathcal{M}_{\mathcal{N}}(x_I - x_J)} \, dx}{\|x_I - x_J\|} \quad (6.6)$$

Distance Map Based Scalar Stiffness

Another idea is to use the distance function value directly as the descriptor of the size. Replacing the size function of the cell with an isotropic one using the distance map data at the point of consideration, the length matrix given in equation (6.2) can be rewritten as.

$$\lambda_{\delta^2} = \left[\frac{1}{\delta_{\mathcal{D}}^2}, \frac{1}{\delta_{\mathcal{D}}^2}, \frac{1}{\delta_{\mathcal{D}}^2} \right] \quad (6.7)$$

Applying these eigenvalues into equation (6.1), an isotropic metric is obtained. However, this metric does not show any control mechanisms in terms of the anisotropic directions. Reminding the analogy between the metric and the ellipsoid, the isotropic metric can be represented as a sphere, where no primary axis can be identified. Therefore, instead of using a metric stiffness, one can use the scalar value directly to control the cells sizes. This can be done by modifying the stiffness function such as:

$$k_I(x_J) = \frac{1}{\sqrt{\delta_{\mathcal{D}}^2}} \frac{1}{\|x_I - x_J\|} \quad (6.8)$$

where, $\sqrt{\delta_{\mathcal{D}}^2}$ is the average distance function of the cell edge. This stiffness function does not require expensive matrix multiplications that metric based stiffness functions require. Therefore, the mesh control algorithm based on this metric runs significantly faster than the metric based method. The computer memory requirements is also significantly lower for the method based on this metric, since, only distance map function array is kept in the computer memory instead of an array of 3×3 matrices.

6.1.2 Selection of Distance Map Function

The value of the distance to the wall function approaches to zero close to the wall. This creates a significant numerical problem, since all the cells are attracted to the walls, and near the walls, very compact cells are achieved. Therefore, the distance functions should be constrained after computation. Another control that is necessary to be imposed on the distance function is the control of the global size distribution. The ratio of maximum and minimum cell sizes in the mesh should be constrained to impose a better control on the mesh quality. Therefore, maximum global anisotropy ratio, A_{max}^2 , which is defined as the ratio of the maximum and the minimum cell sizes desired to be in the mesh, is introduced

to limit the skewness of the mesh. Therefore, the distance function is modified such as its value changes between 1 to A_{max}^2 .

In this thesis, two distance map functions are exploited. First one is a digital city-block type distance function that is used in conjunction with the scalar distance map stiffness function. The definition of such distance maps is given in Section 5.1. To compute the stiffness on this map, distance function is integrated on a digital line, pixels of which is calculated with a three-dimensional octree version of Bresenham's line drawing algorithm. Bresenham's algorithm is a well-known algorithm that is used to draw pixel based line in computer graphics. The algorithm is very efficient compared to geometric methods.

The second distance map used in this research is based on exact distance to the nearest wall function. This function is used with mesh control algorithm employing both metric based stiffness functions. The distance function is calculated using the algorithm given in Section 5.4.1 and Algorithm 5.2. It should be noted that, the Hessian computation uses the square of the exact distance to wall data and the distance function is modified to change between 1 to A_{max}^2 after the computation of the Hessian.

6.1.3 Numerical scheme

The method is implemented in the refinement operations in the overlay grid based mesh generator HEXPRESS. First, a background mesh is created using octree Cartesian meshes using the refinement adaptations. The cells are marked as intersecting to the boundary, inside of the boundary or outside of the boundary. Then using this information, the necessary parameters, such as distance function, Hessian of the distance function and the Natural metrics are computed on this mesh. Also created is the skeleton of the geometry that will be exploited extensively.

After these preliminary steps, the actual controlled mesh generation process starts. First an initial mesh, which is encompassed by the background grid, is inserted into the domain. A spring system is created for this initial mesh, with the desired metrics. The spring system is solved and the mesh vertices are relocated accordingly. Then the usual adaptation refinement operation is applied on the mesh into consideration. The adapted mesh is again converted to network of springs and this system is solved for the refined mesh. This process is repeated until desired cell sizes are achieved, and the details of the geometry are captured.

After geometric adaptation step is finished, the cells intersecting with the boundary and the cells outside of the computational domain are trimmed away from the mesh. In order to achieve a better boundary capturing, the mesh should be smoothed before boundary snapping step. This is done by applying spring analogy based regularisation, which is defined earlier in Chapter 3. For this purpose, the trimmed mesh is converted to a network of springs with the stiffness function given as:

$$k_I(x_J) = \frac{1}{\|x_I - x_J\|} \quad (6.9)$$

After that, the overlay grid generation algorithm is run normally, in other words, the surface boundary capturing is followed by an optimisation process and optional steps of boundary layer insertion and parallel partitioning. The algorithm that encompasses the mesh control methodology is given in Algorithm 6.2.

Algorithm 6.2 Final algorithm for mesh generation control

Require: $\mathcal{T}_B \subset \Omega$, $\triangleright$ background mesh
Require: $\mathcal{T} \subset \Omega$, $\triangleright$ initial mesh to generate
Require: $\mathcal{M}, \delta^2, S \in \mathcal{T}_b \triangleright$ compute the skeleton, distance map and metric on background mesh
1: **for all** Mesh refinement levels **do**
2: $\mathcal{S} \leftarrow \mathcal{T}$ $\triangleright$ convert from mesh to spring system
3: $\Sigma F_s = 0$ $\triangleright$ solve spring system
4: $\mathcal{T} \leftarrow \mathcal{S}$ $\triangleright$ convert spring node coordinates to the mesh nodes
5: Apply refinement adaptation on $\mathcal{T}$
6: **end for**
7: Remove the intersecting and exterior cells from the mesh $\mathcal{T}$ such that $\mathcal{T} \supset \Omega$
8: smooth the mesh

6.2 Meshing Applications

In this section, the meshing examples using the control metrics are presented. In the first part, the mesh control on the compact geometries, as appears in internal fluid dynamics and solid mechanics problems, are presented. These geometries contain detailed sections.

The external aerodynamics geometries contain small, detailed geometry of the aircraft surrounded by a huge computational domain around this geometry. Therefore, the aircraft geometry appears as a small detail in the geometry. Control of such mesh with the same techniques used for compact geometries seems not possible. The algorithm should be modified slightly to capture the details of the aircraft geometry.

6.2.1 Mesh Generation Control for Compact Geometries

The first test case presented in this section is a moulding geometry taken from the HEX-PRESS testing geometries database. The geometry consists of several parallel surfaces

with highly varied distances between each other with varying curvatures. A sensible natural metrics can not be computed for this test case, since the parallel surfaces are not flat enough to let bubble grow to an anisotropic shape before the interpolation fails. Therefore, the stiffness functions used for this test case are distance metric based and scalar distance map based.

In order to be able to move the vertices successfully, the dimensions of the initial mesh are determined by expanding the bounding box of the geometry by a factor of two. This expansion factor is used for all cases with compact geometries. In Figure 6.1, the mesh being adapted is presented. It is seen here that the nodes are moving freely and the boundary mesh is stretched by keeping a good degree of orthogonality. This verifies the applicability of spring boundary conditions defined in Section 3.1.1 for metric based methods.

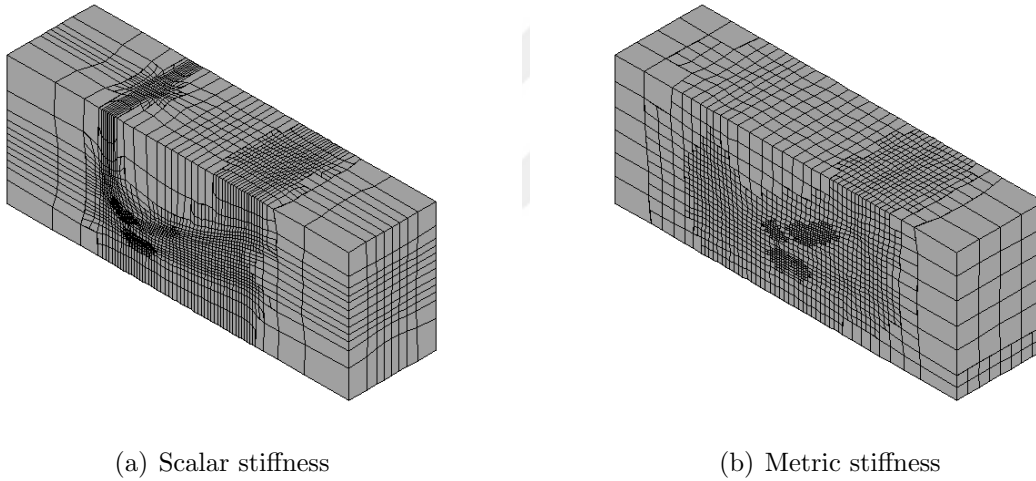


Figure 6.1: Comparison of the deformed meshes on boundary of the adapting mesh.

The results of the test runs are given in Figure 6.2. The original mesh that is produced without applying the mesh control is shown in subfigure 6.2.(a). Since there is a thinner section at the top rounded part, the minimum number of refinement adaptation necessary to capture this thickness is six. The resulting mesh contains 57513 cells.

The next subfigure, 6.2.(b), shows the mesh obtained by controlling the mesh generation process using the scalar distance-map base stiffness, calculated on a digital distance map constrained by maximum global aspect ratio of 5. It is seen that, the number of necessary adaptations can be reduced to five and consequently the final cell count in the mesh is reduced to 31970.

In the last subfigure, 6.2.(c), the mesh produced by the metric based stiffness function can be seen. The number of necessary adaptations is again reduced to five and cell count

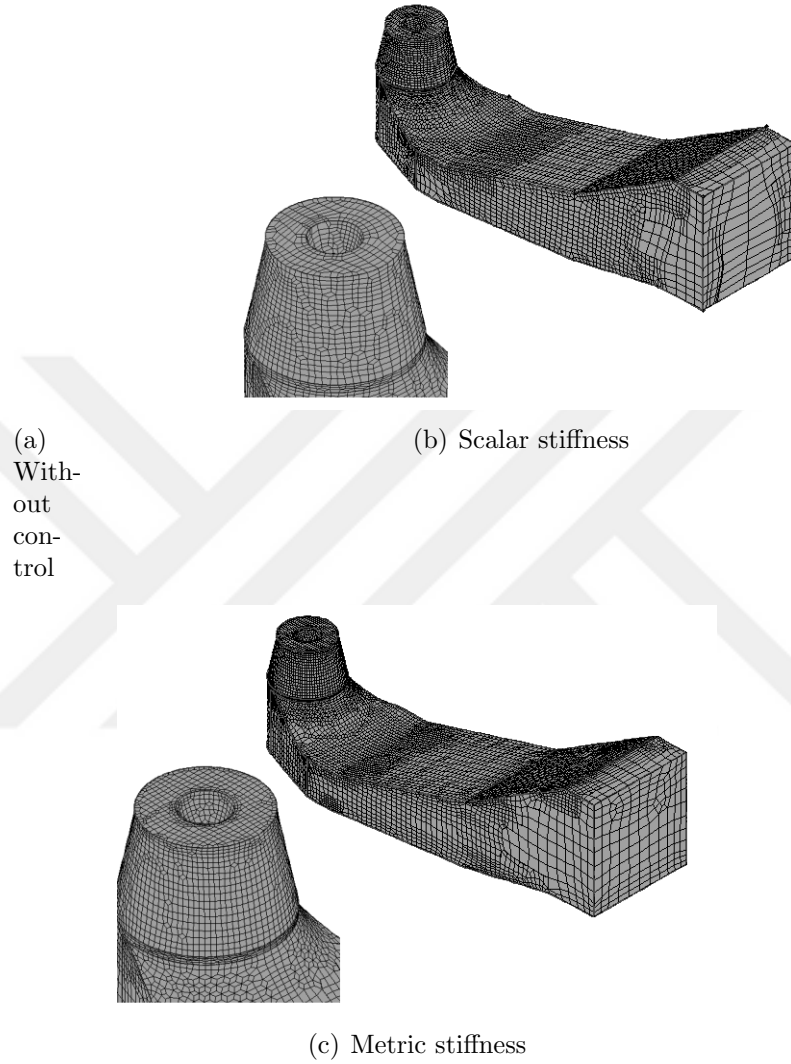


Figure 6.2: Comparison of the final meshes on moulding geometry

is reduced to the 30211 using this method.

Comparing the results visually as shown in Figure 6.2, it is seen that the metric based algorithm has results in the best shaped surface mesh, also better than the mesh produced by unmodified algorithm. This is expected; since the metric based method supposedly produce cells following the change of surface by a metric based on the Hessian of distance map function. This is best visible at the rounded conic surface. The scalar stiffness function based method shows a better tendency to capture the thin surface patches appearing in the domain topology. These kinds of surface patches are inserted by the user to be able to apply different boundary conditions. Such a surface appears at the connection between the half cone and the main body. This patch is captured best by the scalar spring stiffness.

This is also expected, since the isotropic compaction produces very small cell around this zone, which produces a lot of cells around this area that ease surface and edge snapping processes.

The comparison for the cell size distribution with two approaches can be seen in Figure 6.3. It is seen that, the new algorithms are quite efficient in capturing the thin details with lower cell counts. By point relocation using the distance map, smaller cells are located at the thinner sections and the cell sizes are scaled to larger ones at wider regions. It is also observed that, the metric based algorithm produces the most anisotropic sizes, while the scalar based control method has better distribution of orthogonality. The reduction of the cell count is 44.5 percent for the scalar case and 47.5 for the metric case.

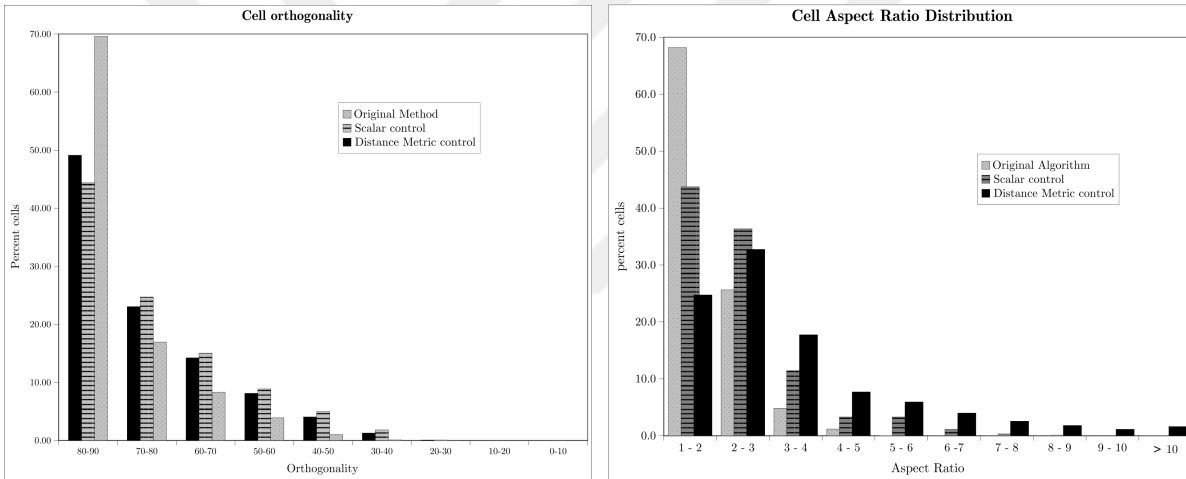


Figure 6.3: Orthogonality and the aspect ratio distribution for test cases applied on moulding geometry.

The second example consists of a volute geometry. The volute has a thin outlet, requiring many refinement iterations to capture. Mesh control technique employing all three stiffness functions defined in Section 6.1.1 are applied on the mesh generation process. It is seen that, the grids produced by natural metrics based algorithm and distance map metric based algorithm show strong similarities for a given anisotropy factor. The meshes are so similar that, it is very difficult to visually distinguish the two meshes. The mesh statistics also show resemblance. This is an indication that, the difference of the metric sizes on the directions of the largest and the smallest eigenvectors are of secondary importance for moderately thin geometries, compared to the effect of anisotropy factor. Therefore, in this thesis, the mesh produced with the natural metrics is not presented in the results.

The three meshes produced with aforementioned algorithms are shown in Figure 6.4. The uncontrolled mesh contains a total cell count of 79637 and is shown in subfigure (a). The detailed view presented in Figure 6.5.(a) shows the excessive number of cells at the thin exit. The mesh produced by the scalar metric on the same geometry is shown in

Figures 6.4.(b) and 6.5.(b) in global and detail view respectively. The cell count reduces to 66886 cells, which indicates a net reduction of 20 percent. The mesh at the exit section has a lower quality for this method. There is a significant improvement on the final cell count of meshes produced by metric based control methodology, as it is seen in Figure 6.4.(c). The final cell count is 41821 cells, which is almost half of the cell count of the original method. Although the cell count is reduced, the exit section is captured perfectly with similar number of layers of anisotropic cells as seen in the detailed view 6.5.(c).

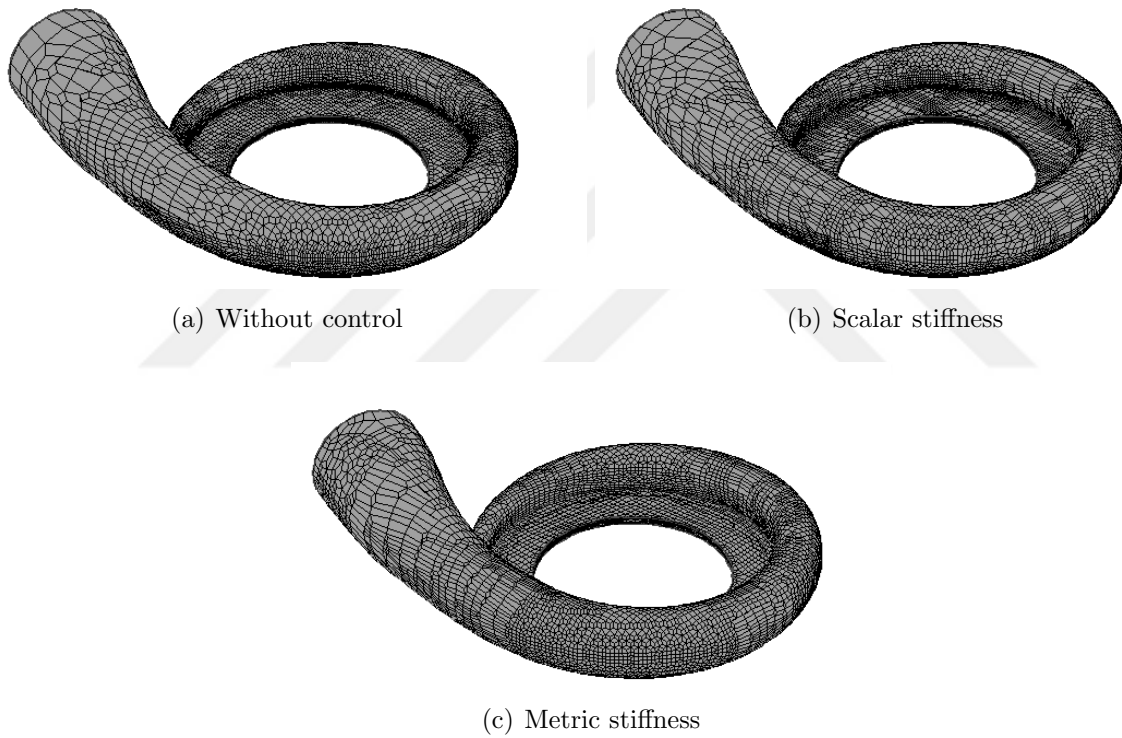


Figure 6.4: Comparison of the final meshes on volute geometry

Figure 6.6 shows the comparison of the cell quality between the meshes obtained by original method and with the controlled method. It is seen here that the cell orthogonality reduces; however, this difference is observed only at the most orthogonal cells. For the rest of the cells, mesh orthogonality distribution with the control methodologies shows a good resemblance with the uncontrolled mesh. The next graph shows the comparison of the results of the original and controlled meshes in terms of cell aspect ratio. It is seen here that, the cell aspect ration distribution is similar, while the metric based method produces a little bit more anisotropic meshes. It should be noted that, anisotropic mesh refinement operation is applied extensively during mesh generation. This explains the highly anisotropic size distribution observed in all cases.

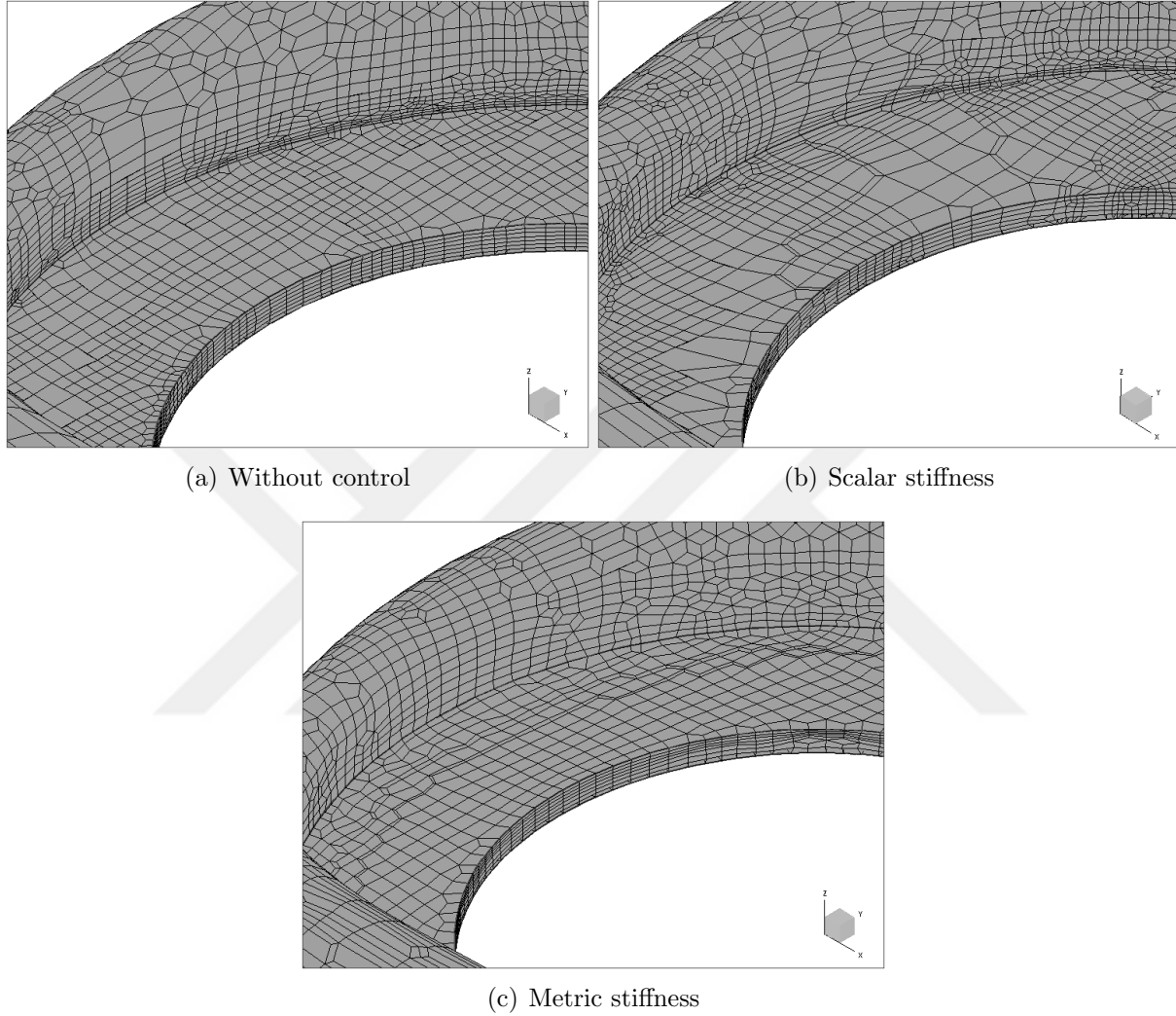


Figure 6.5: Detail view of the final meshes at the diffuser section on the volute geometry

6.2.2 Mesh generation control for external aerodynamics domains

DLR-F4 wing-body configuration is chosen for testing the performance of the mesh control algorithm in external flow configurations. This configuration is a standard test case for aircraft drag prediction. The geometry is defined in the AGARD advisory report number 303 [127]. The test data is available for validation purposes.

External aerodynamic geometries have different characteristics than the compact geometries. In order to be able to apply the mesh control method, some different steps should be taken. It can be safely said that, the quality of the cells at far-field boundaries of the domain is less important than cells located on the aircraft boundary, where the largest

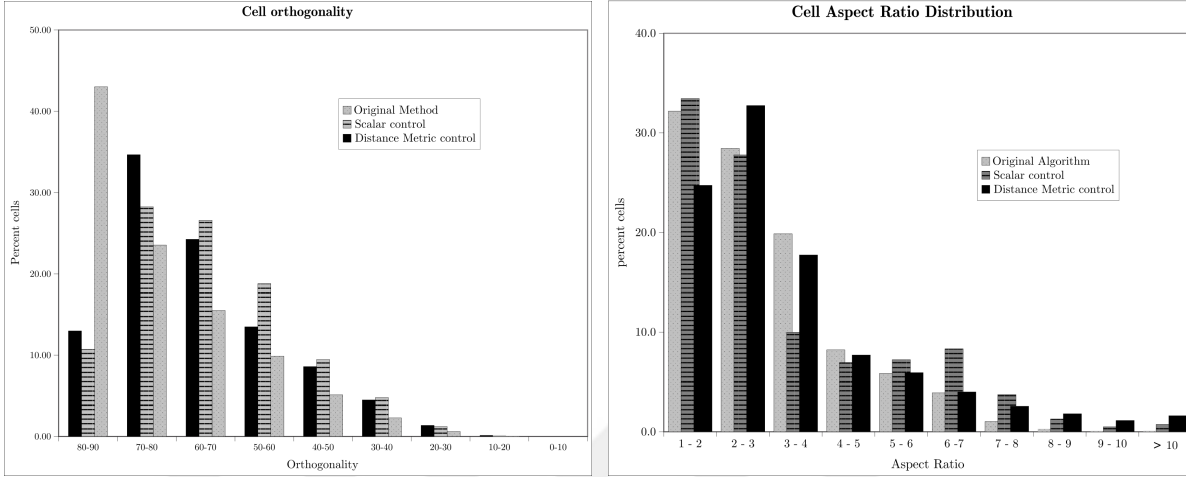


Figure 6.6: Orthogonality and the aspect ratio distribution for test cases applied on volute geometry.

changes of the flow characteristics occur. Also external boundaries are often very simple shapes, and capturing these boundaries does not create any problem for mesh generation. On the other hand, in order to capture the aircraft boundary, many refinement iterations should be applied.

Therefore, one can conclude that, all the effort on mesh control should be concentrated on capturing the solid boundaries. External boundaries can be easily captured with the existing algorithm. Therefore, a reverse approach is taken to control the mesh generation process on external configurations.

The external boundaries are treated as non-existent and only solid boundaries are used for mesh control process. In other words, the operations are applied around the solid boundaries. The solution domain around solid body is large enough to let the springs move around the solid body freely. As a result, initial mesh is not expanded as it is done in the compact cases, but is chosen such that it just encompasses the geometry.

Since the external boundaries are omitted, the distance function is calculated using only the solid boundary. Consequently, the skeleton of the mesh is not computed inside the flow domain, but is calculated at the outside of the domain. In other words, the skeleton of the aircraft is extracted. The resulting skeleton, which is computed by the algorithm presented in Appendix B, is shown in Figure 6.7. It is seen here that the aircraft geometry is perfectly captured.

Summarising the procedure explained above, the aircraft geometry is actually treated as a solid body. This means that, the meshing process is controlled as if the interior of the geometry is being meshed. Then, the mesh generation algorithm is controlled using the distance metric based algorithm with a ratio of maximum anisotropy of 10. The results are promising. The geometry could be meshed with only 62382 cells instead of 102445 cells produced by the uncontrolled algorithm. The final mesh can be seen in Figure 6.8. The net

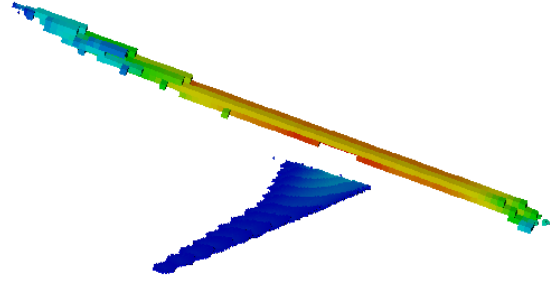


Figure 6.7: Skeleton computed for DLR-F4 test case

gain comes from capturing of the wing tip, as it is shown in Figure 6.9. It is seen here that, since the mesh cells are compacted anisotropically in the wing thickness, less refinement iterations are required, and hence, the final number of cells are reduced significantly.

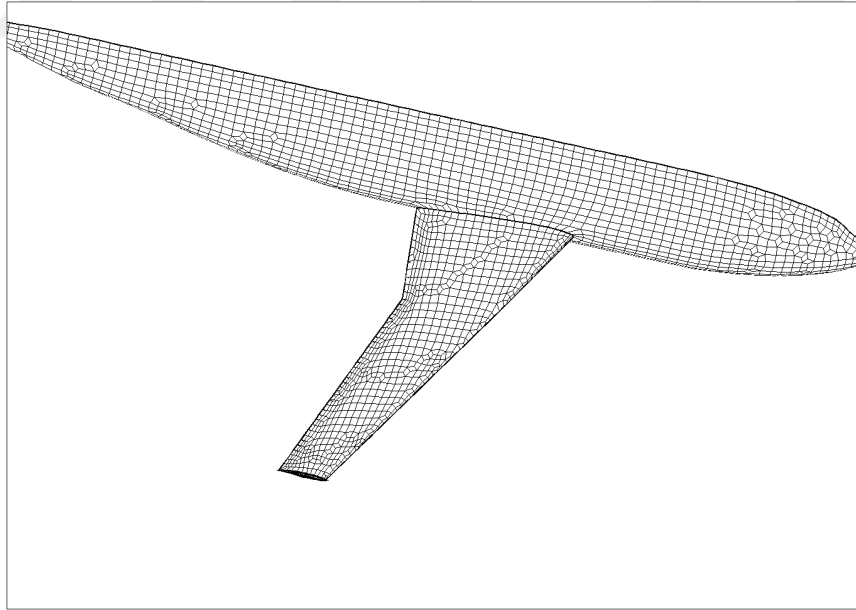


Figure 6.8: Coarse grid generated around the DLR-F4 geometry.

The comparison of the cell orthogonality and the cell aspect ratio are given in Figure 6.10. It is seen that unlike in the case of compact geometries, the mesh control process has no effect on the final quality of the mesh. This can be explained by the existence of a large area to apply optimisation procedure. Therefore, the skew and non-orthogonal cells are easily optimised. This effect can be observed in Figure 6.11.

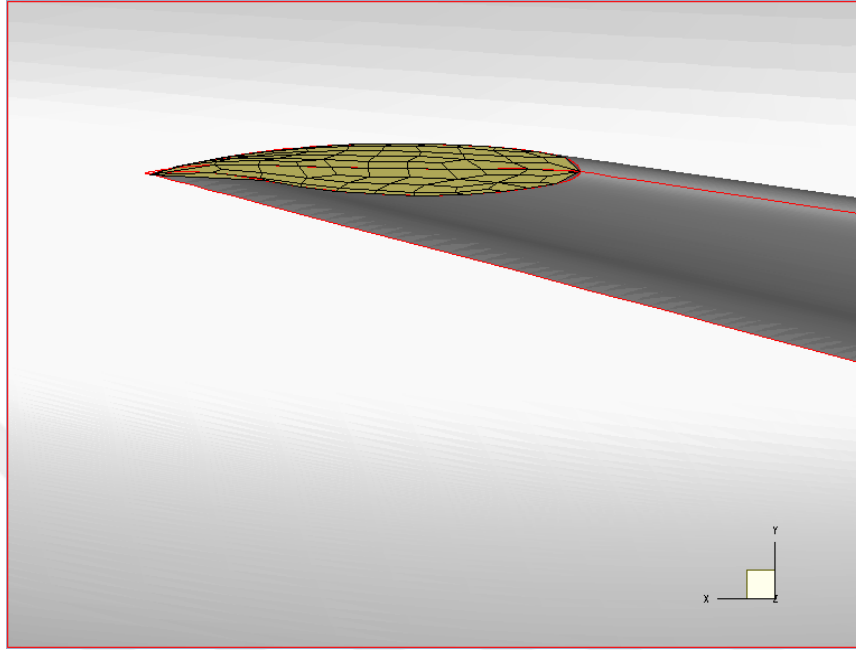


Figure 6.9: Grid at the wing tip of the DLR-F4 configuration.

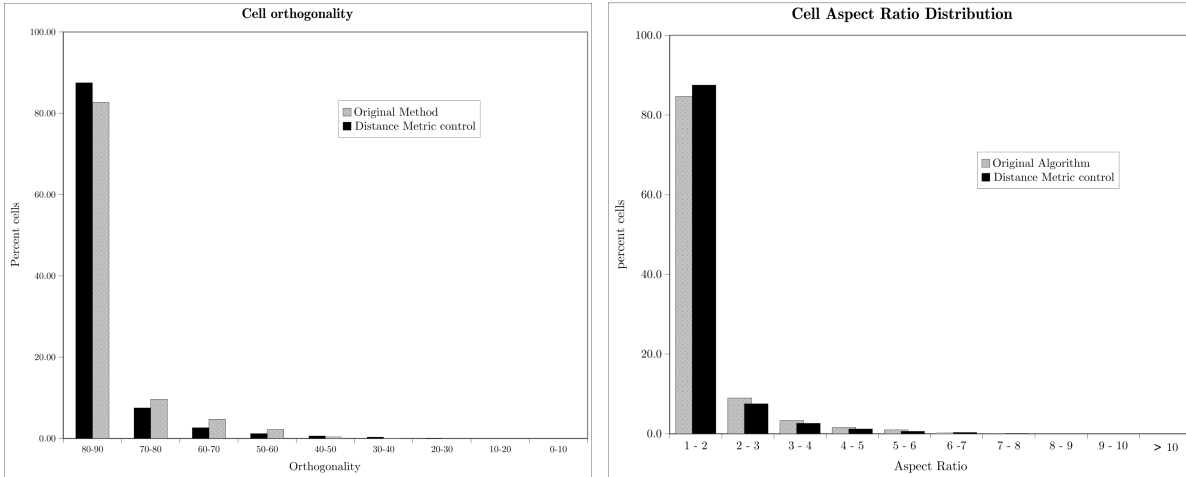


Figure 6.10: Orthogonality and the aspect ratio distribution for test cases applied on DLR-F4 geometry.

6.3 Conclusion

In this chapter, the mesh control methodologies explained in the previous chapters are applied on the mesh generation control. For this purpose, continuous metric fields that are adapted from metrics defined in Chapters 4 and 5 are exploited. These metrics are used as a source of stiffness function for spring-analogy based mesh control algorithm as

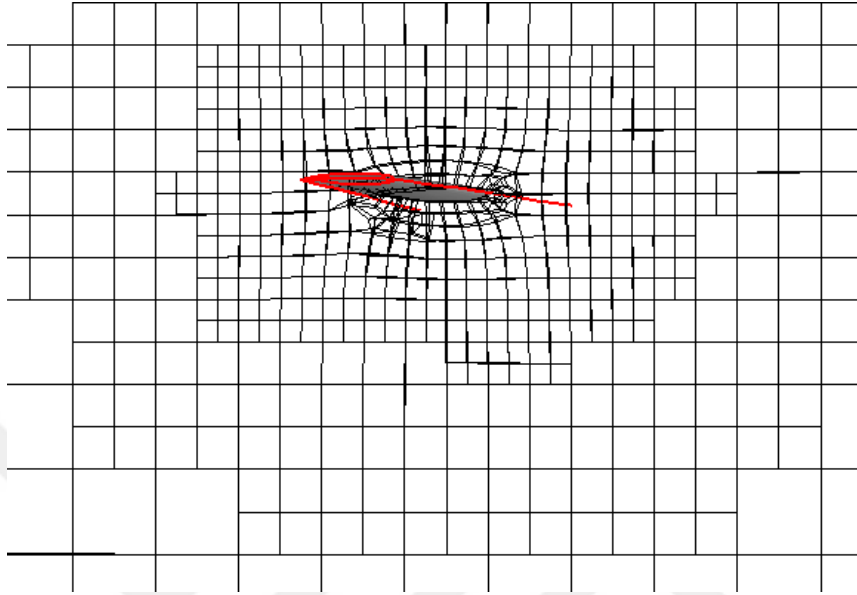


Figure 6.11: Cross-section of the optimised mesh at the wing section of DLR-F4 geometry.

it is explained in Chapter 3. The methods are tested for compact geometries and on an external aerodynamics geometry.

During these tests, it is observed that the distance map based methods are efficient tools for mesh generation control. Two versions of such methods are presented and the meshes, that are generated using these methods, are compared with the meshes produced by the common method. In the first version of these two methods, the distance map information is exploited as a cell size manipulator only. The second method exploits the information of anisotropic directions also.

The first group of tests are applied on more compact geometries, that are seen in internal fluid dynamics or solid mechanics applications. After the tests, following conclusions can be made:

- The meshes obtained with the scalar stiffness function produces the lowest quality mesh. However, the method shows a great potential on capturing small details on the surface meshes. This is a common situation, where a topological edge should be captured separately to be able to apply different boundary conditions. Therefore, the potential of compacting the cells using this stiffness function can be exploited for such meshes. This is addressed as a future study.
- The meshes obtained by distance map metric based stiffness function, gives excellent results for thin and parallel sections. Although the mesh quality is lower than the meshes with uncontrolled generation, they are still in a acceptable quality. How-

ever, the usability of meshes generated with this approach should be verified. The verification is made in Chapter 7 on a compressor rotor blade geometry.

- In both cases, the controlled algorithms produce meshes with half of the cell count of the meshes generated by the original algorithm. Therefore, the goal of increasing efficiency of the overlay grid based hexahedral mesh generation is achieved.

In the second group of the geometries, external aerodynamics configurations are considered. It is seen that, the treatment of such geometries are different from the treatment of compact cases. The skeleton of the geometry is computed at the outside of the flow domain and inside of the solid body. The external boundaries are not considered during the mesh control process. The following conclusions can be made based on these results.

- The controlled mesh has the same quality metrics as the uncontrolled one. This is a result of the huge external volume, that the optimisation can be performed very successfully.
- The final controlled mesh has 40 percent less cells than the original algorithm. This shows that the local anisotropy captured well. The effect of the reduced cell count is discussed in Section 7.2 after the numerical simulation is performed on this geometry.

Unfortunately, the natural metrics based control algorithm can be applied only on the volute test case. For the other cases, the metric extraction does not yield a good definition of the features of the geometry. It is seen that, the results obtained with this metric is very similar to the results obtained by distance-map based method. The final meshes are even not easily distinguished from each other. Natural metric extraction in the geometry process takes enormous computational time compared to the distance map metric. Therefore, the use of the natural metrics for such purposes is not recommended.

Chapter 7

Numerical simulations

In Chapters 3-5, the theory of the mesh control system is developed. The proposed system is implemented and the result of controlled mesh generation are presented in Chapter 6. It is seen that, the controlled meshes have better anisotropy, less cells and reasonable cell quality distribution.

However, one cannot conclude about quality of the meshes without performing numerical simulation on the mesh. In this chapter, the meshes produced with the mentioned control algorithm are used in an internal and an external aerodynamics simulation. Numerical results are compared with the solutions on structured grids or uncontrolled grids depending on availability.

7.1 Turbomachinery application

In order to test the performance of the new mesh in internal flow, the NASA rotor-37 test case is chosen. This test case is included in a wide research program on developing design parameters of high-pressure compressor inlet stage of aircraft turbine engine. The data for the tests can be found in AGARD report 355 [128]. Rotor-37 is an isolated transonic axial compressor rotor with 36 blades as shown in figure 7.1.

7.1.1 Test Conditions

A series of experimental measurements are conducted on the rotor with a rotational speed of 17188.7 RPM. The choke flow is determined by NASA experiments at $m_{choke} = 20.93 \text{ kg/s}$. A near stall flow rate is determined experimentally at $m = 0.925 * m_{choke} = 19.36 \text{ kg/s}$. The overall performance of the rotor is determined between the range of flow rates $0.98m_{choke}$ and $0.925m_{choke}$.

The inlet boundary conditions are derived from the experimental data, where radial profiles of absolute total pressure and total temperature profiles were measured 4.19cm

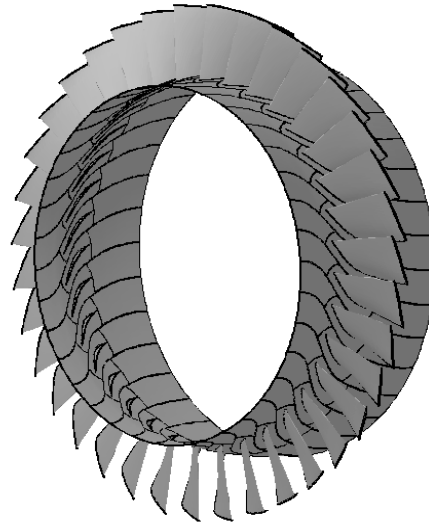


Figure 7.1: Rotor-37 geometry

upstream of leading edge. For this station 3% turbulent intensity is also measured. In the numerical simulation the velocity is assumed axial. The walls are assumed to be adiabatic.

Compressible air is used as the fluid model. Properties of the fluid are given as:

- Specific heat $C_p = 1004.5 J/kg K$
- $\gamma = 1.4$
- Prandtl number: 0.708
- Kinematic viscosity is found by Sutherland Law

7.1.2 Computational grids and solution procedure

An structured grid and a unstructured hexahedral grid produced by the distance-map metric based algorithm is used for computations. The structured grid contains 624000 grid cells. With the metric based mesh generation control, a 454000 cells mesh is produced. This number increases to 1.154 million cells after a high quality boundary layer cells insertion. This is a strict necessity for a successful simulation.

Unstructured mesh generation via a grid based algorithm is quite complex task due to very thin gap between the turbine blade and the compressor casing. The number of necessary refinement iterations to capture this thin gap is very high, that results in a big final cell count. Indeed, a reference hexahedral mesh for comparison reasons cannot be computed

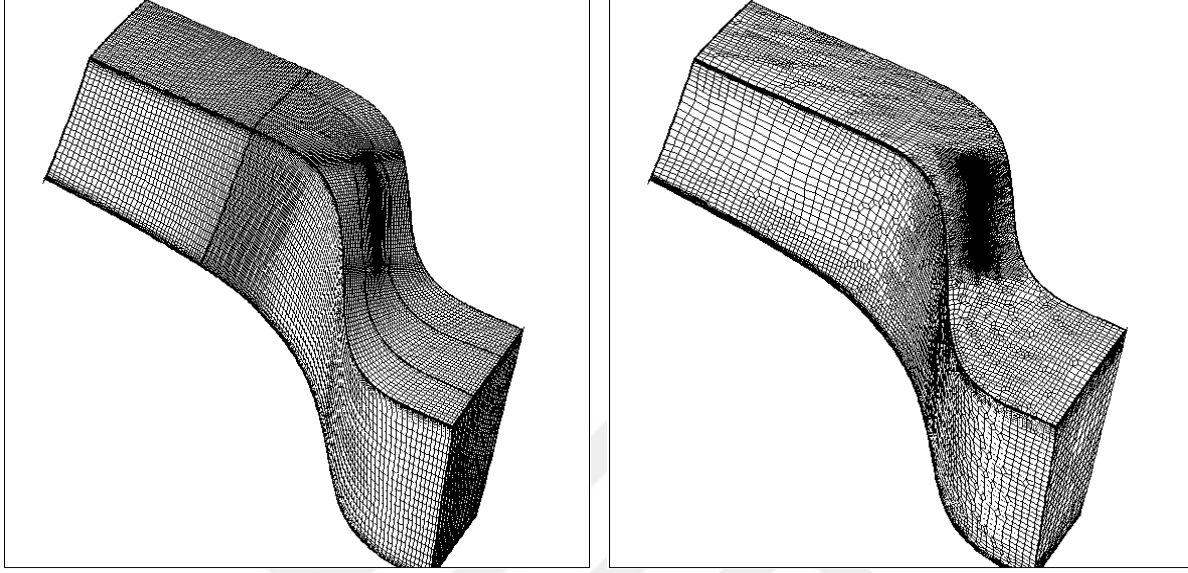


Figure 7.2: Structured and the unstructured meshes for Rotor-37 test case.

with the existing computer facilities due to the high computer memory requirements. The meshes are shown in Figure 7.2.

Unstructured hexahedral flow solver, *HEXSTREAM* of Numeca International is used for flow simulations. For the turbulence model, Spalart-Allmaras one-equation model [129] is used. Since the aim is the demonstration of the performance of the controlled hexahedral mesh, the run is applied on the near stall condition. The calculations are run in parallel with four processors.

7.1.3 Results

The mass flow found for the calculations are 18.94kg/s for the multi-block structured mesh and 18.79kg/s for the unstructured one. Experimental data contains average total pressure ration, P_r and adiabatic efficiency, η , which are defined as:

$$P_r = \frac{p_{t1}}{p_{t2}} \quad (7.1)$$

$$\eta = \frac{\left(\frac{p_{t1}}{p_{t2}}\right)^{\frac{\gamma-1}{\gamma}} - 1}{\frac{T_{t1}}{T_{t2}} - 1} \quad (7.2)$$

where, subscripts 1 and 2 stands for inlet and outlet conditions. The comparison of the results for these data are given in Figures 7.3 and 7.4. Here, it is seen that, the results

obtained on structured and the unstructured meshes complies well with the experimental data, considering the extra complexity created by thin gap between blade and the casing wall. Both codes underestimate the mass flow. Similarly efficiency is found lower than the experimental data as 0.821 and 0.802 for the structured and unstructured meshes respectively.

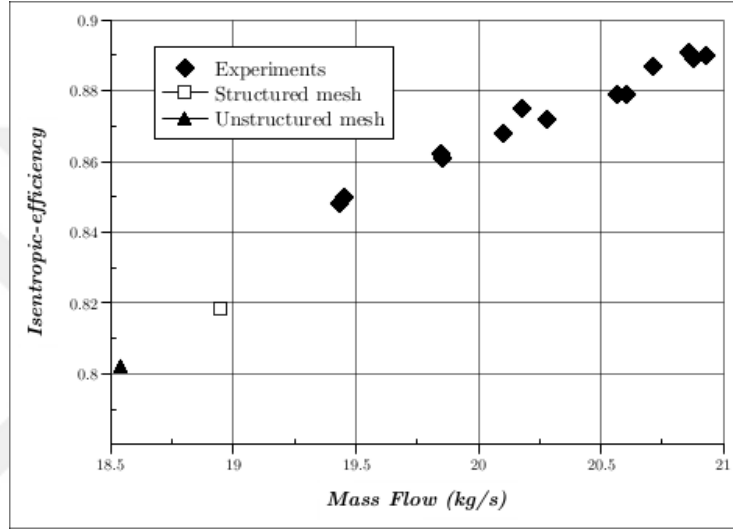


Figure 7.3: Rotor-37 test case, comparison of anisotropic efficiency

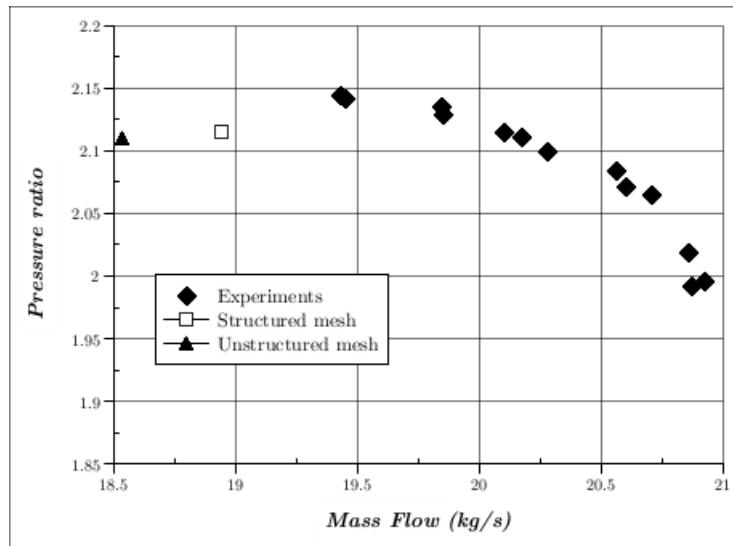


Figure 7.4: Rotor-37 test case, comparison of pressure ratio

In figure 7.5 relative Mach number plots for the midspan for both meshes are shown for comparison. It is also seen here that, on the structured grid shocks are captured sharper. This is normal considering that the unstructured grid is coarse in some critical section. The relative Mach number plot of the solution on unstructured grid is shown in figure 7.6. Here, it is seen that, the grid at the inlet is not fine enough to capture the detail of the flow. However, the solution on the unstructured grid complies well with that of the structured one.

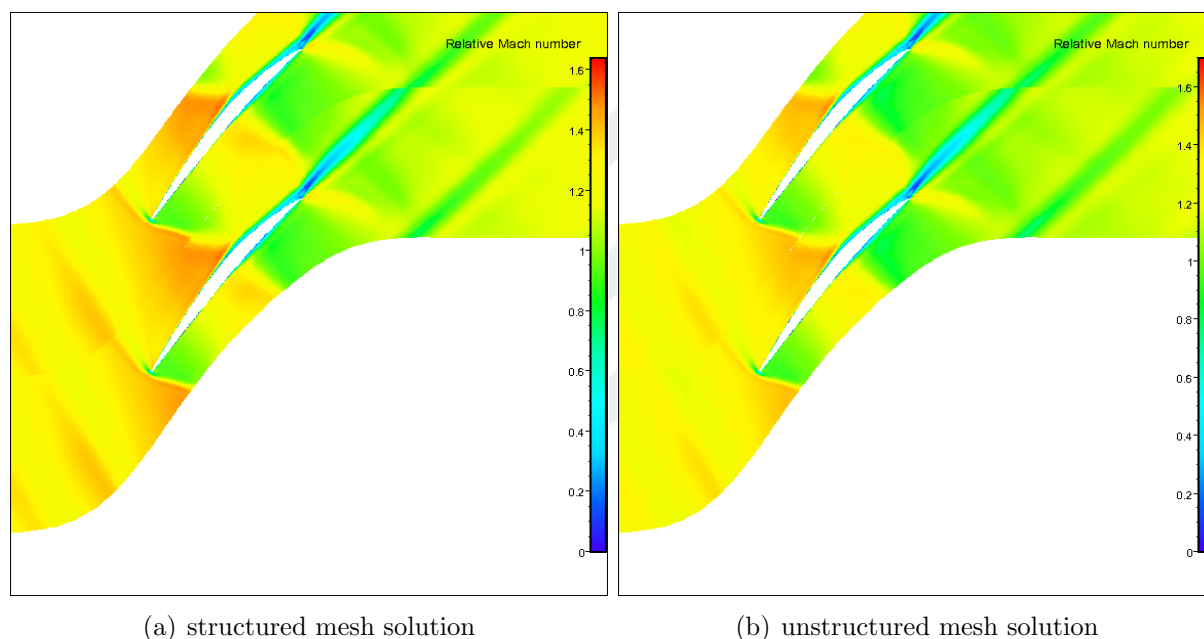


Figure 7.5: Relative Mach number plots for Rotor-37 case for structured and unstructured meshes at the midspan.

7.2 External aerodynamics

In order to test the performance of the mesh control process, DLR-F4 test case is chosen. Meshing process for this geometry is already given in the previous chapter.

DLR-F4 wing-body configuration is prepared for the validation of computational fluid dynamics codes on drag prediction around a subsonic type aircraft. Experimental measurements are carried out in three different European wind tunnels in order to compare

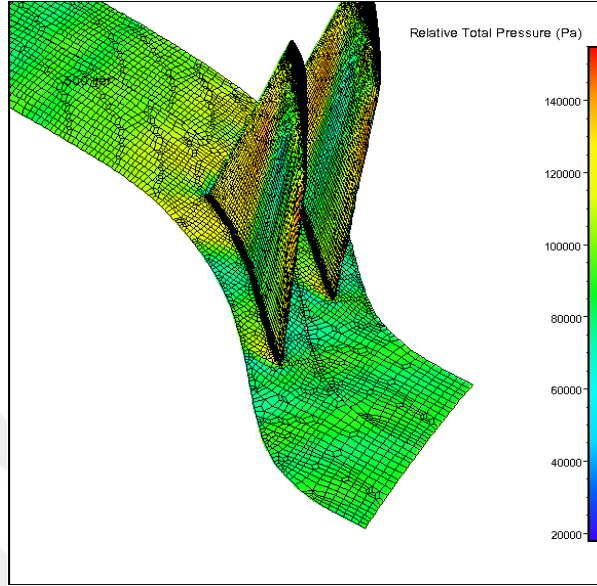


Figure 7.6: Relative total pressure plots for unstructured Rotor-37 mesh.

the results of the configuration in different wind tunnels. The geometry, test conditions and the test results are published in the AGARD report no 335 [127].

In a workshop, organised by AIAA Applied Aerodynamics Technical Committee, flow around this configuration is computed by participants of the workshop. The results for flow configurations selected for this study are compiled into a final report [130].

7.2.1 Numerical simulation

Repeating all the tests that are presented in the reference studies [127, 130] is beyond the purpose of this study. One of the flow conditions, which is given in the Table 7.1 is selected for the numerical simulation.

Re	3×10^6
M_∞	0.75
p_∞	100909.2 Pa
T_∞	293 K
α	0.93°

Table 7.1: Flow conditions for DLR-F4 test case for the numerical test.

The aim in this tests is not to find accurate results for flow around the geometry,

but to demonstrate the concept of controlled mesh generation in external aerodynamics configurations. Therefore, the meshes that is given in the section 6.2.2 are selected for simulation for comparison reasons. Insertion of boundary layer cells is applied on the meshes in order to be able to perform the simulations. After the boundary layer insertion, cell count for the mesh generated by the controlled process is increased to 112000 cells, whereas for the uncontrolled mesh, this number increases to 430248. The boundary layer characteristics are the same for both meshes, the difference comes from the minimal number cells on the surface mesh. Since the boundary layer cells are inserted only on the surface mesh, the difference between number of boundary layer cells are huge.

Of course, the coarse surface mesh for controlled-mesh case is not suitable for an accurate numerical solution. Therefore, on the coarse controlled mesh, first inviscid flow solution is computed. Then, an adaptation is applied on the mesh using both velocity gradient and the pressure gradient. The refined mesh has 376098 cells.

Then viscous flow solution is obtained on both the uncontrolled mesh and adapted mesh, using the Spalart-Allmaras one equation turbulence model [129] by the flow solver HEXSTREAM. The pressure plot and the meshes for controlled (a), controlled-adapted, (b) and the uncontrolled, (c) mesh is shown in Figure 7.7.

Comparison of the results of refined and uncontrolled mesh with the experimental results is given in Figure 7.8. In this figure pressure distribution is plotted at various wing sections. For the same configuration and the same flow solver, Patel [131] suggested to use meshes with the cell count larger than 2 millions for an accurate solution. Therefore, these results are as expected for much coarser meshes.

In the same figure, it is seen that, although the controlled mesh has smaller cell counts, the agreement of the solution with experimental data is better. The reason for the higher performance observed in the adapted mesh is the refinement operations applied, such as the new cells are inserted only at the necessary locations in the flow domain, where the high gradients of the flow variables should be captured. The concept is shown in Figures 7.9 and 7.10. It is clearly seen in these figures that, the fine cells are located only at the necessary locations, where the flow changes its characteristics at most. The locations where the cells, the sizes of whom affect the solution less are left coarse. Therefore, numerical results with better accuracy are achieved with less number of cells.

7.3 Conclusion

In this chapter, numerical simulations performed on the controlled grids produced by mesh control methodology developed in the previous chapters are presented and discussed. Numerical results obtained on these meshes are compared with the results obtained on conventional grids.

In the first part, a turbomachinery application on rotor geometry is performed on a hexahedral grid obtained with the controlled grid generation algorithm. Selected test case

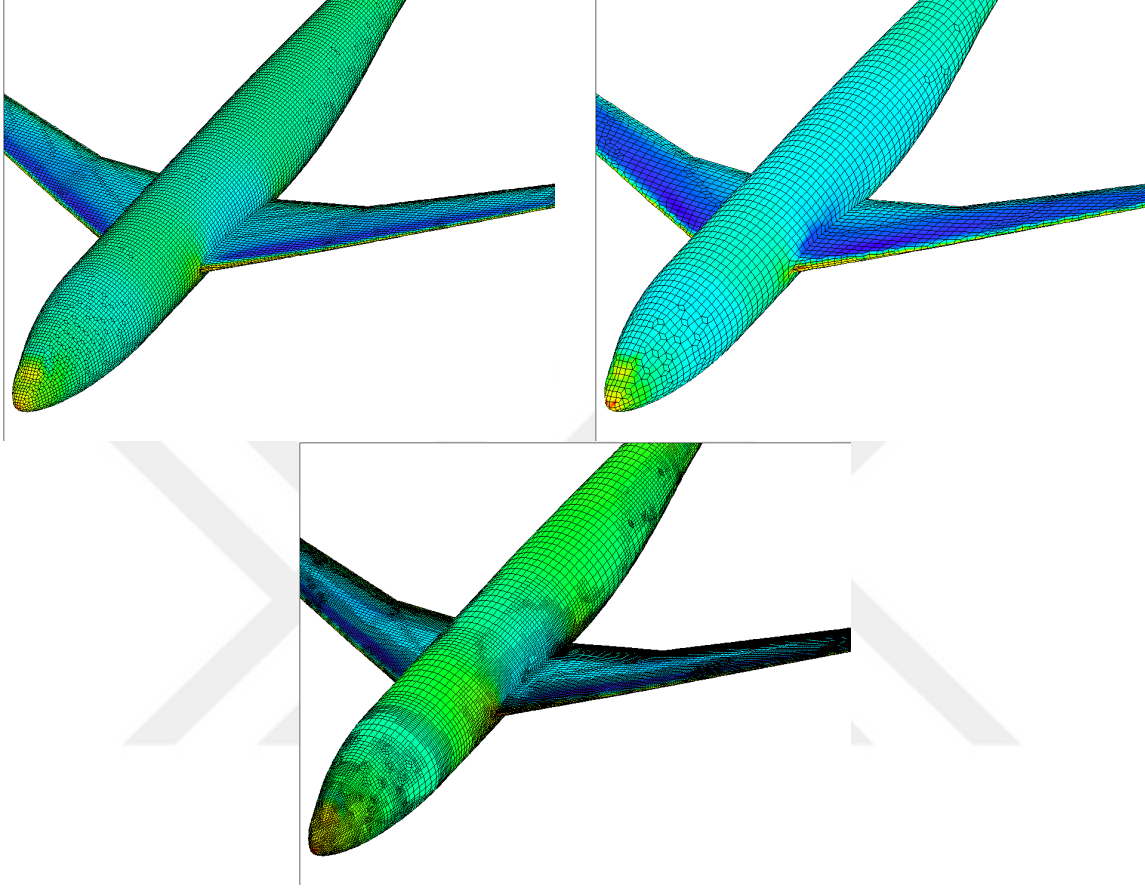


Figure 7.7: Pressure distribution on DLR-F4 geometry for different mesh densities.

is quite difficult to mesh with a reasonably low mesh count, due to the existence of a thin gap between the blade and the wall.

The results match well with the numerical solution applied on a structured grid. The difference between the solution of the structured and unstructured grids comes from the coarse grid at the inlet section in unstructured grid. This shows that, the grid should also be controlled for computational requirements. For example, the grid anisotropy and size should be controlled for the inlet section of the grid. This problem is addressed for future studies.

Although the solution on the unstructured grid is not as accurate as the solution achieved on structured one, the goal is achieved. Without controlling the mesh generation process, the number of cells to capture the geometry becomes too high to perform a simulation on the grid. On the other hand, the structured grid requires a lot of human effort. Therefore, our algorithm makes the automatic mesh generation available.

In the following section, an external flow simulation is performed on the mesh generated with the new algorithm. The external flow is solved around this geometry and then one

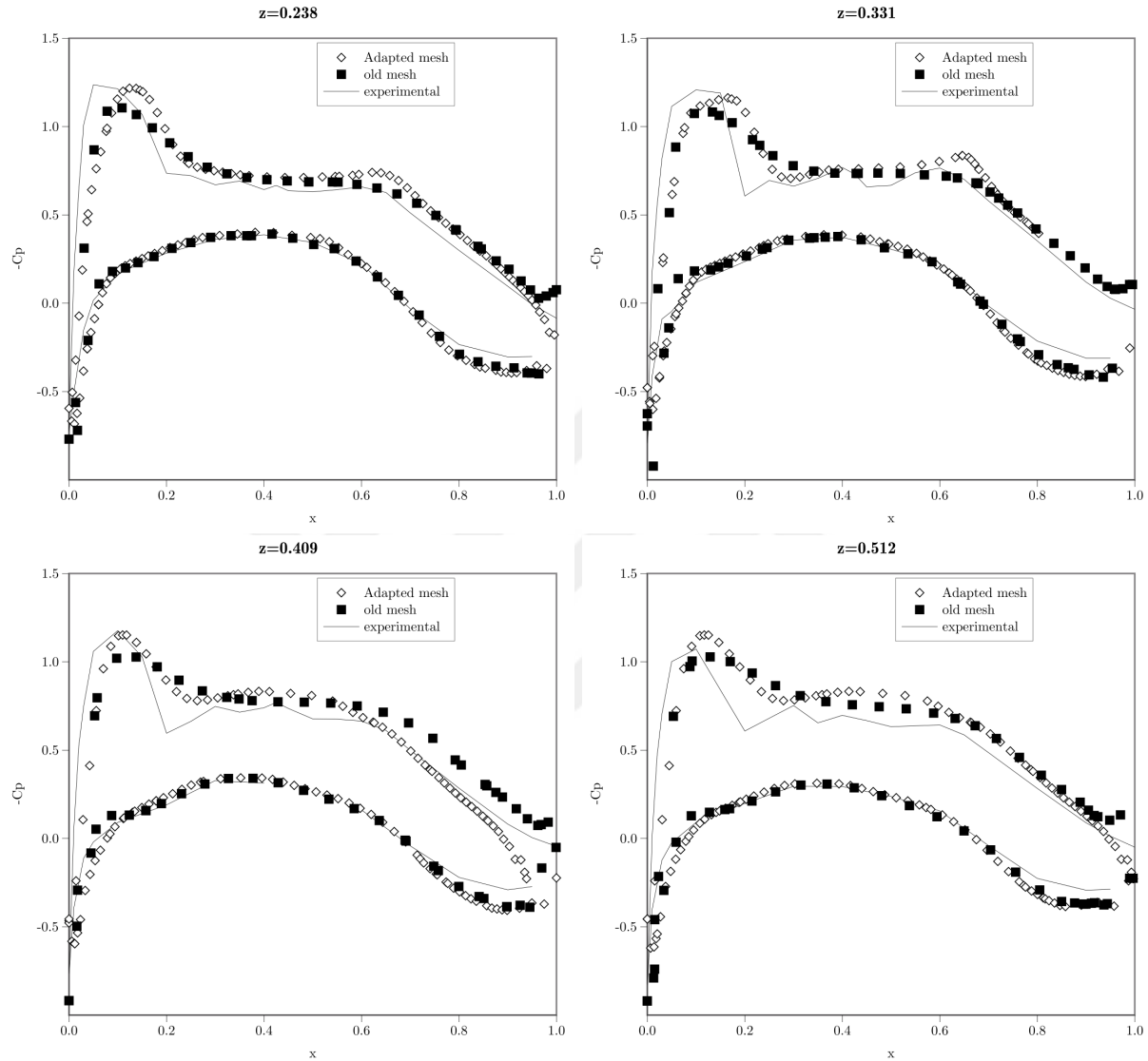


Figure 7.8: C_p distribution in various wing span sections for DLR-F4 geometry

adaptation refinement is applied to the mesh at the locations with high gradients of the flow parameters.

The solution of the finer grid is compared with the solution on a reference coarse grid. It is seen that, the adapted grid gives better results, although the adapted grid contains less cells than the reference grid.

Therefore, the mesh control allows the user to select a mesh basis with minimum number of cells for maximum solution quality. An industrial user can select the level of mesh detail to meet the simulation requirements with a minimum number of cells. If the mesh is found

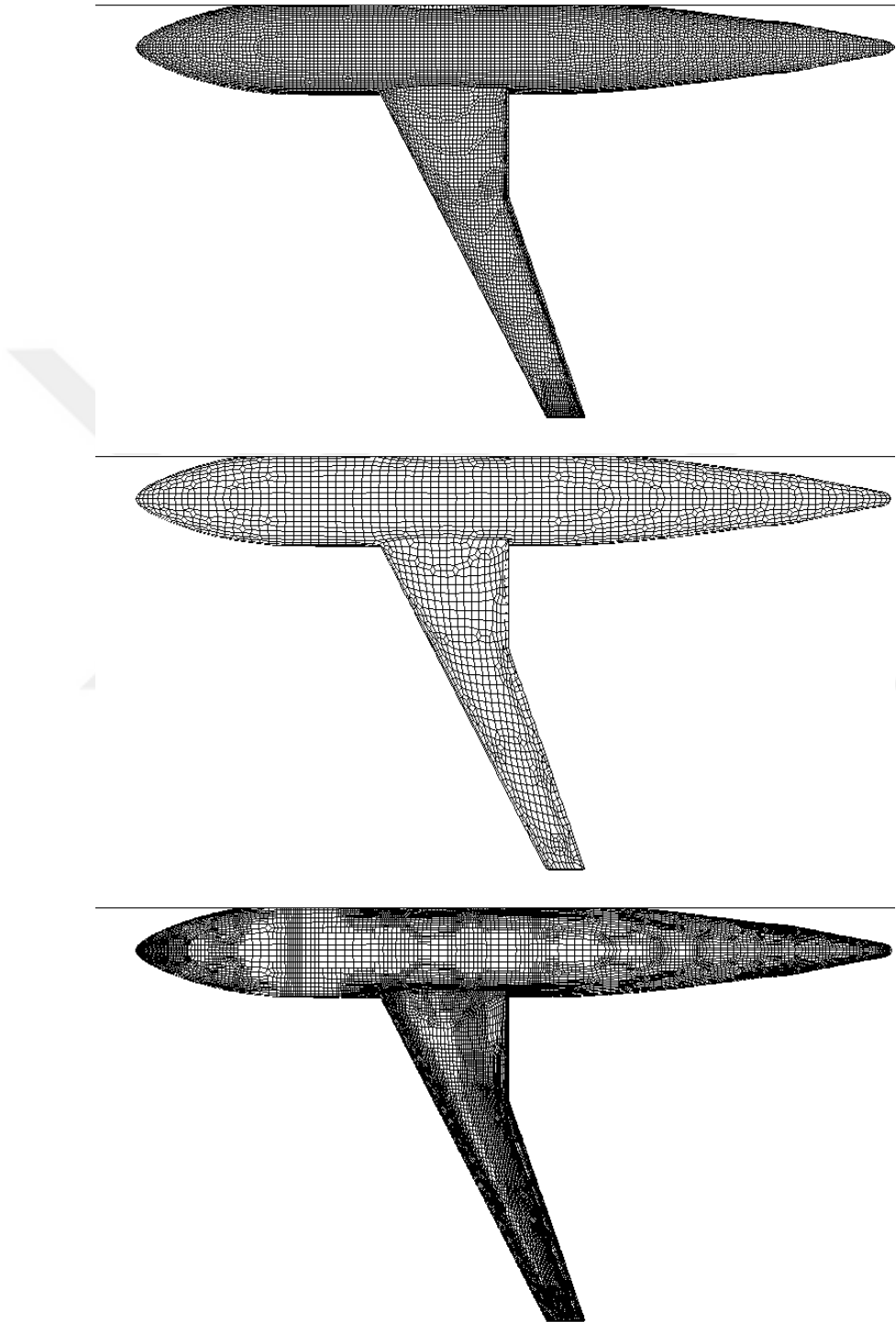


Figure 7.9: Bottom view of DLR-F4 geometry for original, controlled and refined grids

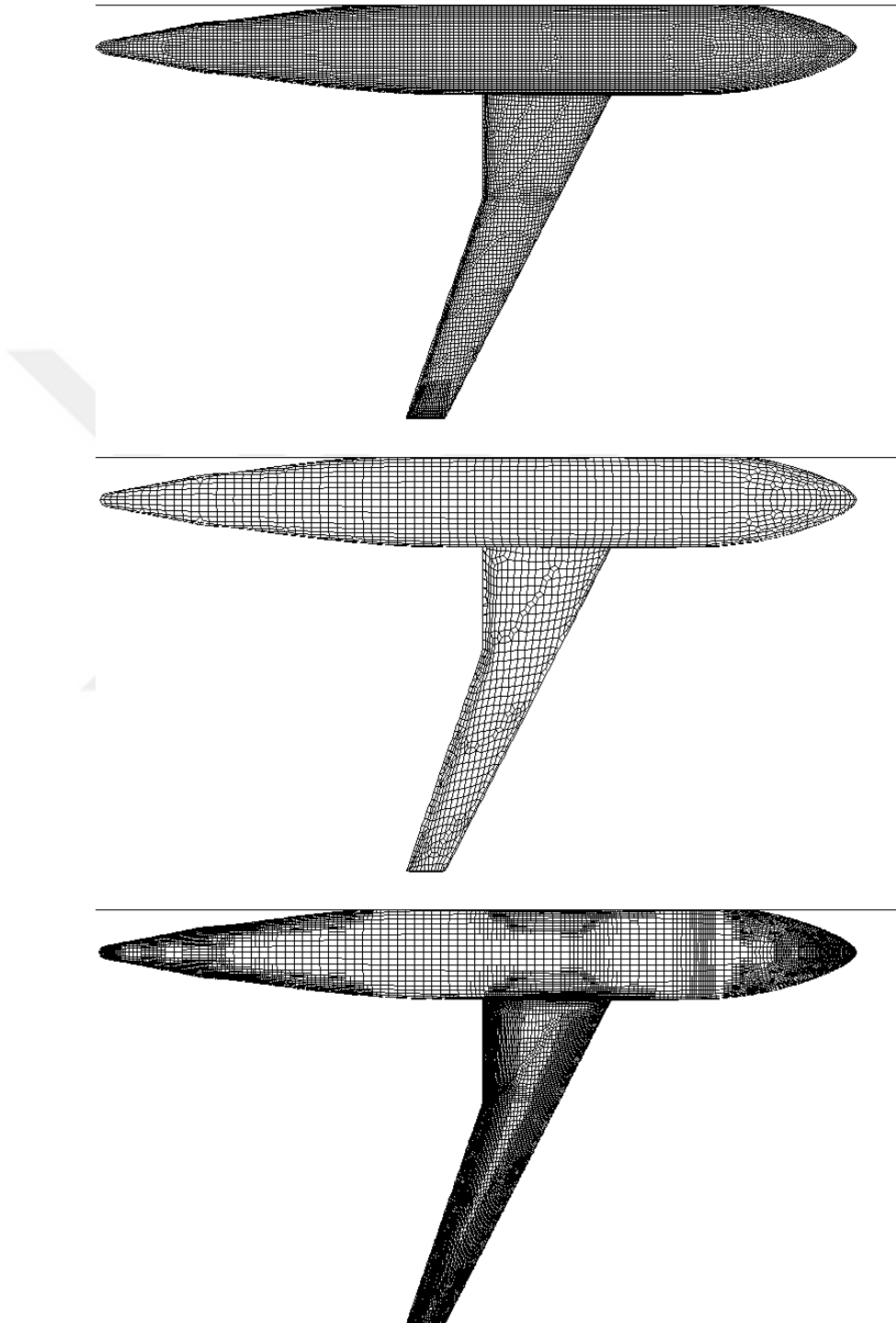


Figure 7.10: Top view view of DLR-F4 geometry for original, controlled and refined grids

to be too coarse, the mesh cell count can be controlled with adaptations, without requiring remeshing. Therefore, the new approach presented here, reduces the necessary time for meshing and remeshing, which is one of the goals of this thesis.



Chapter 8

Mesh control in simulation

In the previous chapters, a mesh generation control technique is presented. With this technique, the mesh vertices are relocated to new positions with the help of anisotropic metrics, which are defined to fulfil optimal mesh generation parameters. The method is based on point relocation algorithm, so called spring-analogy, where mesh edges are converted to a network of springs. The stiffness of the springs in this system is determined by a given metric field, then the spring is set to move freely to the equilibrium position under the tensile forces applied on the vertices by the springs.

Although it is used only in the anisotropic mesh generation control, the mesh control mechanism is very flexible and can be applied to any process involving grid control. There is no obstacle to the application of the method on mesh post-processing, in other words simulation-time mesh control. The only necessary operation is to determine suitable parameters for properties of springs to be used in mesh relocation algorithm. These properties include a stiffness function and the base segment length. Selection of these parameters is application specific and should be done accordingly.

Consequently, in this chapter a preliminary study on simulation-time mesh control is presented in order to demonstrate the expansion possibilities of the algorithm. The moving boundary problem is selected as the model problem. Necessary control parameters for spring system are presented and the new method is applied to two-dimensional geometries. To verify the quality of the mesh, numerical simulations are performed.

8.1 The segment springs approach and moving boundary problem

The moving boundary problems are popular especially in simulations of fluid dynamics problems as well as fluid-structure interaction problems. In these problems the boundary of the geometry evolves from its original shape. The change of the boundary shape requires

re-organisation of the mesh, since the cell vertices at the boundary change their locations with the moving surface. Accordingly, the quality of the cells at these locations becomes very low and the solution is to update the locations of all the vertices inside the domain to sustain mesh quality.

Originally spring-analogy is developed to control such systems [99]. The spring system moves the nodes in an equilibrium position, which is shown to be an optimal position in terms of edge size distribution in Chapter 3. However, the mesh smoothing via spring analogy changes the features of the original mesh. For example, the mesh clustering in certain locations do not appear anymore, since the spring system tends to regularise edge sizes. Similarly, the optimisations applied on the grid such as orthogonality may disappear from the relocated mesh.

Therefore, the spring analogy should be redesigned in such a way that, with a given deformation, the change of cell quality metric should be minimum for every cell. Accordingly, for a stationary mesh, the relocation of mesh vertices should be zero. This can be achieved with the segment springs approach, which is introduced at Section 3.1. In segment-spring variant of the method, the springs are not assumed to be always under tension; instead an equilibrium length of each spring, where spring forces are zero, is considered. If a spring is longer than this equilibrium length, then tensile forces, otherwise compressive forces apply. If one selects the equilibrium length of the spring as the initial length of the spring, l_{IJ}^0 , which is the length of the edge before the boundary movement applied, then equation (3.5) becomes:

$$x_J = x_J + \omega \frac{\sum_{I=1}^n (x_I - x_J - l_{IJ}^0) k_I(x_J)}{\sum_{I=1}^n k_I(x_J)} \quad (8.1)$$

One can observe that, for a stationary mesh, the net movement of the node by equation (8.1) is zero, since $l_{IJ}^0 = x_I^0 - x_J^0$. Therefore, for a given movement, the deformation of the cells is a minimum, where preservation of the mesh features are guaranteed. The same stiffness function with the node-spring approach can be utilised to control the vertex relocation such as:

$$k_I(x_J) = \frac{1}{\|x_I - x_J\|} \quad (8.2)$$

It should be noted that the implementation is applied on octree meshes, which requires the introduction of the weight factor that analogous to equation (3.7) is given as:

$$W_I = 2^{2l_I} \quad (8.3)$$

Inserting this weight to update equation (8.1) one gets the final equation for equilibrium position for single node as:

$$x_J = x_J + \omega \frac{\sum_{I=1}^n W_I(x_I - x_J - l_{IJ})k_I(x_J)}{\sum_{I=1}^n W_I k_I(x_J)} \quad (8.4)$$

This implies a system of equations, which can be solved by updating Algorithm 3.2 accordingly.

The boundary conditions can be applied as it is described in section 3.1.1. The solution of the boundary spring system using segment springs is not obligatory and can also be achieved by applying by node-spring approach. However, in order to respect the initial optimised sizes of the edges, it is recommended to use segment-spring analogy also for the boundary spring system. Therefore, the clustering of the cells at the boundary is not changed. The boundary segment-spring system can be implemented by assigning the initial segment length of boundary connecting node to that of mirror spring.

However, the boundary of the system is the most important part of the numerical solution. Therefore, the optimised distribution of the cells should be preserved as much as possible. Therefore in this study, the approach of fixing a certain number of layers of cells around the boundaries is selected, in order to further ensure the quality of solution around boundary.

8.2 Application of the method

8.2.1 Mesh movement

In order to achieve the goal of simulation-time grid control, the segment-spring analogy described above is implemented in the flow solver HEXSTREAM, which is an highly capable unstructured hexahedral flow solver of Numeca International. HEXSTREAM can work cooperatively with the mesh generator HEXPRESS, the mesh generator used as a basis in this study. The implementation is not limited to two-dimensional meshes, while in this thesis only such grids are considered.

The test case chosen is the standard NACA0012 airfoil geometry inside a flow domain. The initial mesh around the airfoil consists of 3992 cells. The airfoil is rotated around quarter chord by 50° in two steps. The rotation then, handled with the segment-spring analogy. The simulation required only 12 seconds on a 1GHz Pentium class computer. Spring system is reached an accuracy of 10^{-5} in 112 and 108 iterations in each rotation step.

It is stated that the boundary conditions are extremely important for the numerical simulation accuracy. Therefore, the cells around this region should keep their optimised shapes, while the cells away from the boundary compensate the effect of the boundary movement. This can be easily achieved by making the springs close to the boundary *stiffer* than the others. For this purpose, a certain number of layers of cells adjacent to boundary

are selected to be stiffer than the others, and this extra stiffness is imposed by modifying the weight factor given in equation 8.3 for these cell-edges such as:

$$W_I = F^2 2^{2l_I} \quad (8.5)$$

where, F^2 is a sufficiently large number to ensure that the selected cells do not change their length significantly.

For the given example, the number of layer of *stiff* cells near the boundary is selected as 10 while F^2 is set to 10^5 .

Figure 8.1.(a) shows close-up view of the mesh around the rotated airfoil. It is seen here that, although a very big boundary movement is applied, the mesh cells moved to optimal positions. The shape and the orthogonality of the mesh are preserved. Cells from multiple refinement levels are handled correctly.

In Figure 8.1.(b), cells at the located at the very vicinity of the boundary is shown. It is seen here that, cells around the airfoil preserves their shapes and orthogonality as of they are rotated with the airfoil.

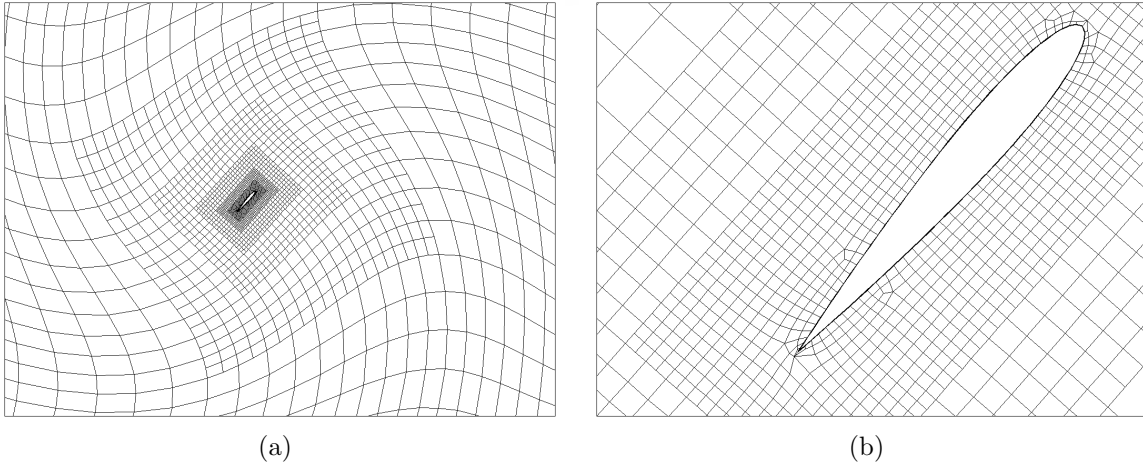


Figure 8.1: Deformed mesh via segment-spring analogy.

8.2.2 Numerical simulation

For the application of the moving mesh method, transonic flow around an oscillating NACA0012 airfoil is simulated. The test case is also studied experimentally and data available from AGARD report number 702 [132]. The problem is a standard test case for moving boundary problems and numerical solutions are available in literature [131, 133, 134].

The problem consist of a NACA0012 airfoil, with an average incidence angle, α_0 and oscillating around the oscillation axis located at quarter chord with a constant frequency, such that the incidence angle as function of time is given as:

$$\alpha(t) = \alpha_0 + \Delta\alpha \sin(\omega t) \quad (8.6)$$

The reduced frequency of the airfoil, f_k , which is a relation between the time required to move the wing in half of the chord length with free stream velocity to oscillation frequency, is given as:

$$f_k = \frac{\omega L}{2u_\infty} \quad (8.7)$$

The simulation conditions are given in Table 8.1.

M_∞	0.775
p_∞	100909.2Pa
T_∞	293K
$\alpha(t)$	$0.016^\circ + 2.51^\circ \sin(\omega t)$
f_k	0.0814

Table 8.1: Flow conditions for NACA0012

Inviscid flow is solved around the airfoil with the given conditions using the flow solver HEXSTREAM. The simulation is run for $\Delta t = 0.000748s$ for 360 time steps. Between each time step, mesh relocation algorithm is applied by fixing the first two layers of cells adjacent to solid and external boundaries as is is described before. The resultant mesh and the static pressure in the domain for rotation in first quadrant are plotted in the Figure 8.2.

The evolution of the lift coefficient with the incidence angle is given in Figure 8.3. In this figure, the experimental data is also plotted. The small difference between experimental data and the numerical solution has also been observed in previous studies [131, 133, 134].

8.3 Conclusion

In this chapter, the mesh control algorithm presented in previous chapters is extended for the simulation-time mesh control. Grids with moving boundaries are selected as model problem. The point relocation algorithm that is used as a basis of the mesh control algorithm is modified to achieve this goal. The segment springs method is selected to handle

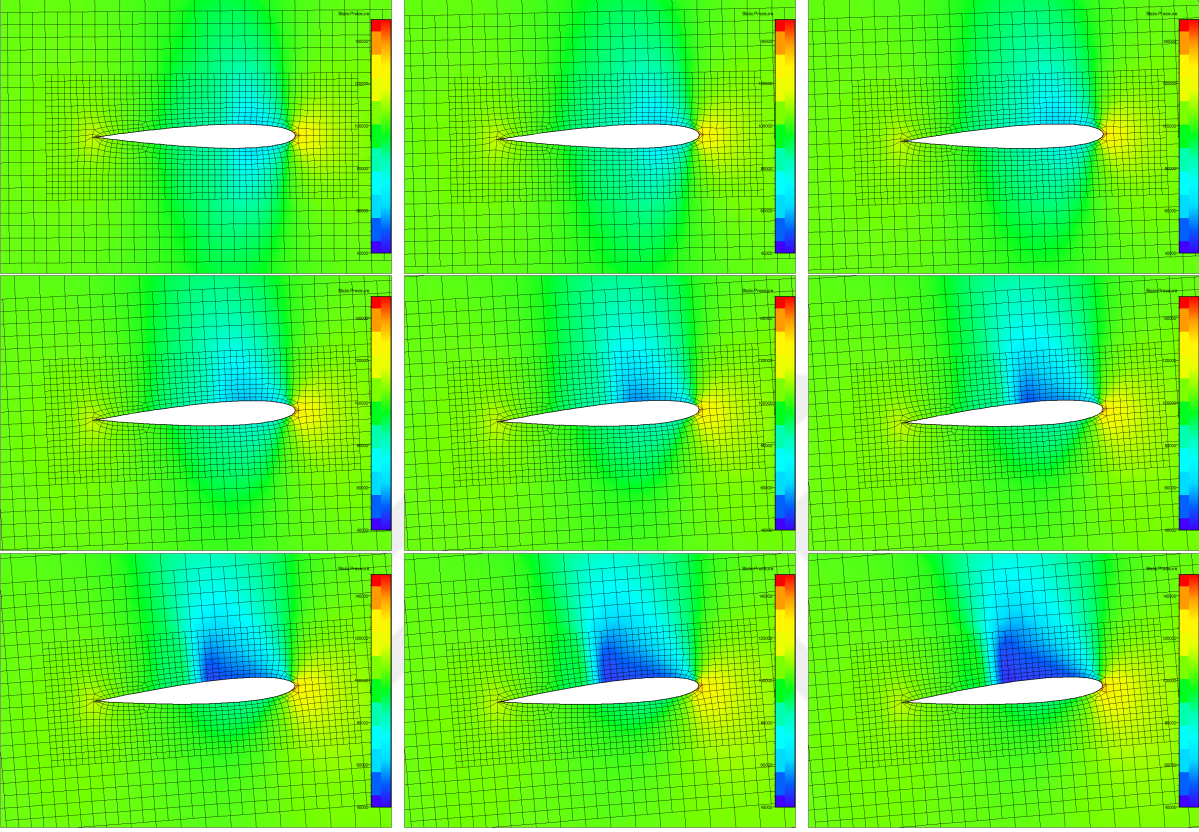


Figure 8.2: Pressure plot over oscillating NACA-0012 airfoil at different time steps.

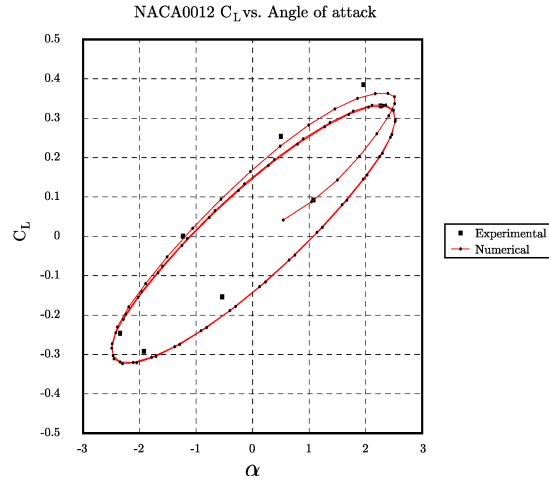


Figure 8.3: Lift coefficient C_L versus angle of attack plot for oscillating NACA0012 wing

mesh deformation, in order to keep the features of mesh assigned to it at the grid generation step.

It is stated that the most important section of the mesh in terms of simulation accuracy is the boundary. Therefore, in this region of the geometry, the quality of the initial mesh should be preserved further, in order to ensure the quality of numerical simulation. Therefore, a certain layer of cells near boundary moved with the boundary itself to preserve the optimised shape at this region.

The numerical simulation performed on such a mesh shows that the method is very successful. The cells near the boundary kept their shape, and the cells at the middle are moved to good positions. Numerical simulation on an unsteady moving boundary problem is demonstrated on oscillating NACA0012 airfoil, where the mesh movement is handled by the grid control algorithm. The solution complies with the experimental solution.

The possibilities for using mesh control algorithm in simulation time are not limited to moving boundary problems. Mesh adaptation with spring analogy is also a very popular topic. The method relies on the replacing mesh vertices with the prescribed analogy using a metric field which is computed using the Hessian of the flow parameters. Then, this data is used as an error estimator and the mesh vertices are relocated such that the error due to spatial representation of the mesh is minimised [105]. This topic is assigned as future work.

Chapter 9

Conclusions and Recommendations

In the scope of this thesis, control methodologies are implemented on an overlay grid based automatic all-hexahedral mesh generator. The objective of the research is to allow user control the features of hexahedral grid in the mesh generation step providing the automation of the mesh generation process. Therefore, the mesh generation process greatly is simplified for the user and more optimal meshes are generated for numerical simulation purposes. Hexahedral grids are reported to be superior to the tetrahedral grids in terms of simulation accuracy by several researchers. The required cell count to achieve similar solution accuracy is four to ten times less for hexahedral meshes than the tetrahedral ones.

However, all-automatic mesh generation techniques have a tendency of producing excessive number of cells in the volume, which prevents the industrial user to exploit this appealing feature of the hexahedral meshes. Therefore, the main objective is established as to develop control methodologies on generation process of all-hexahedra meshes. As industrial design and development cycles are aimed by this thesis, the generality of the geometries that can be meshed by the algorithm should be provided.

The problem is more challenging than it seems. The generation of hexahedral meshes are much more difficult than the tetrahedral meshes. Consequently, main research effort on the hexahedral mesh generation is concentrated on the automatic generation of valid meshes, rather than controlling the generation process. Unfortunately, after more than a decade of research on the hexahedral mesh generation, none of the methods comply with industrial requirements. Most of the methods lack geometric generality. All-automatic methods are also very rare. Apparently, the difficulty of producing all-hexahedral meshes led researchers to develop hybrid meshes.

One of the few hexahedral mesh generation algorithms that met great industrial acceptance is the overlay grid based methods. This method relies on successive refinements of the cells of an initial mesh that encapsulates the geometry until enough cells are located into the geometry. Then cells intersecting with the boundary are trimmed and the topology is captured by insertion of buffer cells. The algorithm provides very robust mesh generation options, and that is the reason such a mesh generation algorithm is used as basis

of controlled mesh generation process during this thesis. The research is conducted on mesh generator HEXPRESS with a close collaboration of Numeca International, a Belgian company with a great expertise on simulation of fluid mechanics problems.

Overlay grid based algorithm are often criticised due to the fact that, the algorithm produces a lot of mesh cells in order to capture smaller details of the domain. The problems sometimes make the task of generating grids with reasonable number of mesh cell impossible. Although it is a very robust algorithm, the method lacks the sense of the layout and features of the geometry. This inefficiency is tried to be compensated by extra refinement operations.

Therefore, the mesh generation process controlled such as the generation of the mesh follows the layout of the geometry and therefore, excessive operations to capture the geometric features during mesh generation are avoided.

Very few of the control methodologies developed for the tetrahedral meshes can be transferred to the hexahedral meshes. The mesh control algorithm should avoid topological operations applied on the mesh cell connectivity. One of such methods, which is named as the spring analogy, is employed in this thesis. The method relies on replacing the mesh edges with a network of springs. Then the nodes of this spring system are let to move under the tensile forces applied by the springs. By controlling the the features of the springs, mesh features, such as cell anisotropy or cell size distribution can be easily controlled. During this thesis, the spring analogy is embraced to control the mesh generation and mesh regularisation and it is seen that, the algorithm is very flexible and robust.

In order to control the mesh generation, the process of detecting the layout of the geometry should be done in such a way that, this information can be exploited in mesh relocation algorithm. This can be achieved by introducing an anisotropic metric that contains such data. Therefore, the data of local anisotropy should be collected from the domain in terms of a field of such metrics defined throughout the domain. In this research, two of such metrics are developed.

The first metric, which is called the natural metrics, is inspired from a former study conducted in L'Ecole de mines de Paris by Gruau, to impose the anisotropy in tetrahedral meshes for problems involving moulding geometries. The metric extraction method relies on expanding an elastic ball everywhere inside the domain. The shape of the bubble is interpolated to an elliptic metric during this process, until a similarity between the bubble shape and the elliptic metric is achieved. The method is applied on a background mesh.

Unfortunately, it is seen that, the metric extraction method is not suitable for the purpose of developing mesh control methodologies for general geometries. It is seen that, metric extraction algorithm is not robust enough to compute the metric field throughout the geometry. In addition, the computational cost of the method is too high to meet industrial limits. It is also shown that the metric presented in the next experimental metric yields similar results with a much less computational cost.

In the next iteration of the search for a metric that collects information about the geometric features of the solution domain, distance maps are utilised. A distance map

is defined on a background grid, where the data of distance to the closest wall from the cell centre is stored. It is seen that, distance maps combined with the medial axis of the geometry, contains all the necessary information to fulfil the task of detecting features of the geometry.

A mesh control algorithm is developed using these concepts. The validation of the algorithm is made by generating the meshes with the new approach in several geometric configurations. Two of these configurations are compact geometries with fine details. The mesh control metrics are applied by modifying the stiffnesses of the springs locally. It is seen that, the net gain of number of cells is no less than 40 percent for such geometries compared to the unmodified algorithm. The cell quality metrics indicate that the mesh has a lower quality than the original mesh. Therefore, the applicability of the meshes generated with the mesh control algorithm is verified in an internal flow flow problem.

For this purpose, a popular turbomachinery solver validation test case provided by NASA, named Rotor-37 is selected. This geometry consist of a compressor turbine stage and contains a very thin gap between the blade and the compressor wall. The geometry is meshed using controlled mesh generation algorithm, with a reasonable number of cells. This was not possible without any control of the mesh, since capturing of thin gap produces too many cells to be handled by ordinary workstations. The results of numerical simulations run on this case are compared with a solution obtained on a structured grid. It can be concluded that, despite the fact that some critical sections of the mesh had been kept too coarse for a good numerical solution, the simulation on the controlled hexahedral mesh is comparable to the solution achieved on the structured grid. Therefore, using the new algorithm, the simulation of the numerical problem is made available on hexahedral grids, which is one of the main goals of this research.

The second group of controlled meshing verification is applied on an external aerodynamics domain. The DLR-F4 wing-body configuration validation test case is used for purpose. It is seen that, the control process of the external flow domains is quite different than that of compact geometries. The mesh control algorithm is tuned for the external flow cases and utilised in meshing the geometry. It is seen that, same order of gain in cell count is achieved using the method. Moreover the surface mesh is much coarser than that of produced by the original algorithm.

The flow simulation applied on an external aerodynamics test case clearly shows the value of this study. In this test case, the flow conditions as indicated in the corresponding AGARD publication are simulated numerically on the reference mesh and the controlled mesh. After the insertion of boundary layer cells of required size, the cell count in the reference cell becomes four times more than that of controlled mesh. This is due to the fact that, with the mesh control algorithm, the boundary of the aircraft could be captured by larger cells. The simulation accuracy is not expected to be the same between the reference mesh and the controlled mesh, since the cell sizes on the boundary is quite different. Then an adaptation is applied on the controlled mesh by the variation of the flow variables in the solution. The simulation is progressed using the new mesh.

It is seen that, the results achieved on the controlled-adapted mesh is more accurate than the non controlled one. The interesting point is that, this accuracy is obtained with a less number of cells in the adapted. This is due to the fact that, the mesh cells are only refined where the flow variables changes the most, instead of using a fine mesh. This completes the concept of mesh control introduced in this study. The engineer is allowed to control the mesh density by the requirements of the simulation, but not the shortcomings of the mesh generator. Starting from the least possible cell count, the simulations can be performed in the most efficient way.

This thesis gives an attempt to control the hexahedral mesh generation using metric fields. It is not possible to claim that, all possibilities for geometry-defining metrics are analysed. On the contrary, the new metrics can be developed and replaced with the current ones, all of which have shortcomings. Such metrics are being proposed continuously in computer graphics and image recognition field without an intention of mesh generations. For the future studies, an extensive survey on this field is recommended.

The mesh control methodologies developed in this study are not limited with the mesh generation. The techniques can also be employed in controlling the meshes during the simulation time. The applications include moving boundary problems, moving meshes and the mesh adaptation by mesh clustering. In this thesis, moving boundary problems are handled using segment-spring analogy. It is seen that, using this method large boundary movements can be handled while preserving the mesh quality. Quality of the mesh is verified by numerical simulation of flow around oscillating NACA0012 airfoil. It is observed that, numerical solution is in compliance with experimental data. The verification of the method in three-dimensional mesh movement as assigned as future work. The topics of mesh movement and the mesh adaptation with mesh clustering are addressed for future studies.

Appendix A

Skeletonisation by Thinning

The concept of skeleton of a geometry is first introduced by Blum [49], as a shape descriptor. In this classical work, Blum described the thinning process by grass-fire analogy. According to this, the field that is bounded by the geometry is set on fire, which propagates from boundary to inside. The medial axis, or skeleton of the object is located at points that fire fronts are met, as the object gets burned and thinned.

Thinning the object, as it is described in the grass-fire analogy, is one of the most common methods of extracting the skeleton of the geometry. The idea is based on moving the object boundary inwards layer by layer, detecting collapsing fronts. The digital version of thinning operation is also possible, which can be described as a morphological operation that is used to remove selected foreground pixels from the geometry and leave collapsing cells in the geometry as skeleton cells.

This eroding process is used extensively in computational geometry. For extensive review, papers of Davies and Lam [135–137] can be consulted. The quadtree and octree versions of the algorithm are also possible.

In the algorithm developed for this study, for a single cycle of thinning operation, different number of erosion operations should be applied on cells of different refinement levels on the front. This is a necessary operation, since the front may contain very different cell sizes on different refinement levels of the octree mesh.

The thinning algorithms tend to produce excessive erosion, and should be constrained. To detect these constraints, boundary cells of three levels of *26-adjacency* neighbourhood are analysed to determine the direction of front propagation. The collapsing fronts are selected such that two fronts propagating in determined directions meet at a minimum angle of $\alpha_{min} > \pi - \pi/12$.

For the sake of simplicity, the algorithm for Cartesian meshes is given in Algorithm A.2. The octree version contains recursive thinning operations for cells of different refinement levels to ensure the proper size of the trimmed layer. The skeleton calculated around a geometry with this method is shown in Figure A.1

Algorithm A.2 Skeletonisation by thinning

```

1: repeat
2:   for all  $\mathcal{S}_b \in \mathcal{T}_b$  do  $\triangleright$  for each cell at the boundary
3:      $\mathcal{N}^b = \mathcal{S}_b$ 
4:     for all  $I < 3$  do  $\triangleright$  make three iterations
5:        $\mathcal{N}^b = \mathcal{N}^b \cup \mathcal{N}(\mathcal{N}^b)$   $\triangleright$  find successive neighbours of three levels at the boundary
6:     end for
7:      $C^b = \mathcal{N}^b \cap \partial\mathcal{T}$   $\triangleright$  faces on the surface for three levels of neighbourhood
8:      $\mathbf{V}_b = f(C^b)$   $\triangleright$  the propagation direction is the function of face normals
9:   end for
10:   $\mathcal{N}^{B-1} = \bigcup \mathcal{N}(\mathcal{S}_b) - \bigcup \mathcal{S}_b$   $\triangleright$  calculate the cells of the next layer.
11:  for all  $\mathcal{S}_{b-1} \in \mathcal{N}^{B-1}$  do  $\triangleright$  inner layer cells
12:     $\mathcal{N}^{b-1} = \mathcal{N}(\mathcal{S}_{b-1}) \cap \mathcal{T}_b$ 
13:    for all pairs of  $\mathbf{V}_i, \mathbf{V}_j \in \mathcal{S}_b$  do
14:      if  $\arccos(\mathbf{V}_i \cdot \mathbf{V}_j) > TOL$  then  $\triangleright$  if fronts met with an angle greater than
         $TOL$ 
15:        mark as skeleton
16:      end if
17:    end for
18:  end for
19:   $\mathcal{T} = \mathcal{T} - \mathcal{T}_b$   $\triangleright$  thin the objects by removing the outmost layer
20: until  $\mathcal{T} \in \emptyset$   $\triangleright$  No cells to thin

```

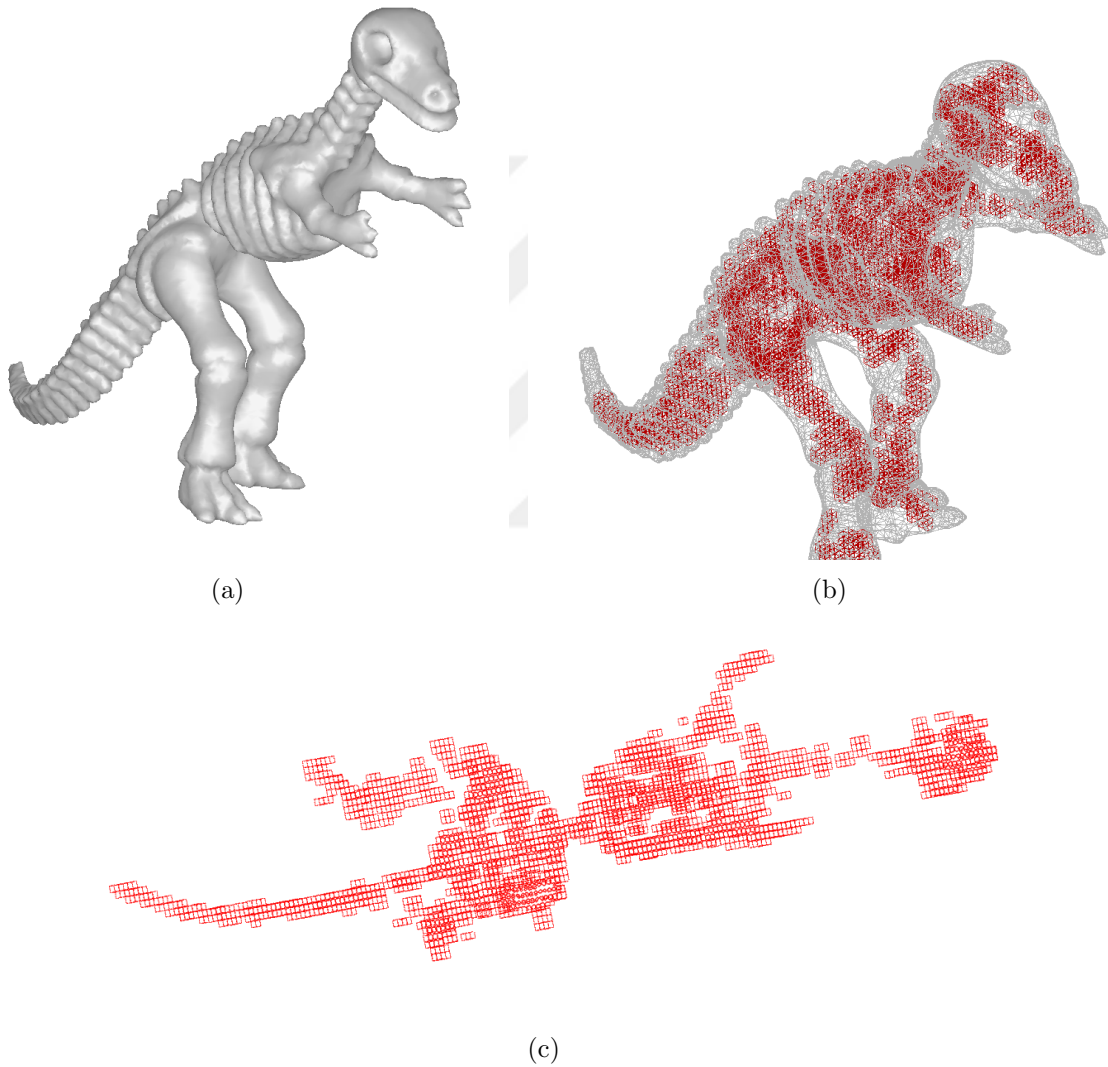


Figure A.1: Skeletonisation via thinning on toy Dinosaur geometry. The original mesh geometry contains 19454 cells. In the skeleton this number is reduced to 3354 cells, with a net reduction of 5.80 times.

Appendix B

Skeletonisation by Distance Transform

The second skeletonisation algorithm uses the information from distance map developed in Section 5.2 to extract the medial. This algorithm is relatively simple compared with thinning-based algorithm, yet, it is more time consuming. However, most of the computational effort is spent on distance map calculations, which is already evaluated for the metric extraction itself. The separate piece of the algorithm that marks the skeleton cells is very efficient compared to distance calculations.

In order to keep the algorithm as an extension of distance map computations, lists of surface triangles that are intersecting with boundary cells are stored at the distance map calculation step. Also reused information is, the mapping for each cell to a list of triangles saved before that contains closest boundary point to the cell centre. Then for every cell, a complete list of triangles that are stored for every neighbour cell around the target cell are marked. For every pair of vector from target cell centre to the closest point on every triangle,

- The angle between them,
- The ratio of their lengths

are calculated and compared with threshold values. The angle tolerance is set as 165° , which ideally should be 180° , and the minimum ratio of distance to the two closest wall is set to be less than 1.4, which should ideally be 1. These tolerances are necessary due to the *digital* nature of the skeleton. This procedure is explained in Algorithm B.2.

Results are extremely good. In figure B.1, toy dinosaur test case is given. Different from the previous method with direct thinning, a denser background mesh of 151042 cells is used in this calculation. Resulting skeleton has only 9880 cells, which is 18 times less than the mesh itself. More important than this, thinning algorithm could reduce this number

Algorithm B.2 Skeletonisation with distance map.

Require: $\mathcal{T} \in \Omega$

Require: $\mathbf{T}(\mathcal{S}) \in \partial\Omega$ $\triangleright$ triangles of the boundary that intersect with cell $\mathcal{S}$
 $\triangleright$ (or $\mathbf{T}(\mathcal{S}) = \mathbf{T}(\mathcal{S}_\Omega)$, $\mathcal{S}_\Omega$ is boundary intersecting cell,
 $\triangleright$ where, $\mathbf{T}(\mathcal{S}_\Omega)$ contains the closest triangle $\mathbf{T}^-$ to the centre of cell $\mathcal{S}$

```

1: for all  $\mathcal{S} \in \mathcal{T}$  do  $\triangleright$  for each cell in the mesh
2:    $\mathcal{N} = \mathcal{V}\{\mathcal{S}\} \cup \mathcal{S}$ 
3:    $\mathbf{T}_\mathcal{N} = \bigcup_{\mathcal{S}_i \in \mathcal{N}} \mathbf{T}(\mathcal{S}_i)$ 
4:   for all  $T_{i, 1 \leq i \leq N} \in \mathbf{T}_\mathcal{N}$  do  $\triangleright$  for each triangle  $T_i$  in  $N$  triangles in  $\mathbf{T}_\mathcal{N}$ 
5:      $\mathbf{V}_i(\mathcal{C}'(\mathcal{S}), \mathbf{T}_i)$   $\triangleright$  vector that connects the cell centre to the closest point on  $T_i$ 
6:     for all  $T_{j, i < j \leq N} \in \mathbf{T}_\mathcal{N}$  do  $\triangleright$  for each triangle  $T_j$ 
7:        $\mathbf{V}_j(\mathcal{C}'(\mathcal{S}), \mathbf{T}_j)$ 
8:       if  $\mathbf{V}_i \cdot \mathbf{V}_j < \text{acos}(\alpha_{\min})$  then
9:         if  $k_{\min} < (\|\mathbf{V}_i\| / \|\mathbf{V}_j\|) < 1/k_{\min}$  then
10:           mark as skeleton cell
11:         end if
12:       end if
13:     end for
14:   end for
15: end for

```

only to 59996 cells, since background mesh becomes much more complex contains more staircase-cells to be resolved.

The next test case is a volute geometry, where the distance map based algorithm produces a skeleton of 30391 cells out of 51131 non-intersecting inner cells. This ratio is quite normal, since the geometry contains a very thin gap at the outlet, where almost all cells belong to the skeleton. The algorithm provides better ratio of number of skeleton cells to number of mesh cells, as more refinement levels are applied to the mesh.

For this kind of geometries, thinning algorithm is very efficient. For example for the volute geometry, thinning algorithm has ended up with the skeleton of 23585 cells. The thinning algorithm is preferable over the distance map based one for thin geometries, since it is faster and uses less computer resources. The requirement to keep the database of intersecting triangles including distance calculation for every cell makes this difference. This is the main reason why the thinning algorithm is used for Natural Metric extraction, albeit it is a less accurate method. In distance map metric calculations the second method is used, since results of many calculation steps necessary to metric extraction is reused in the skeletonisation.

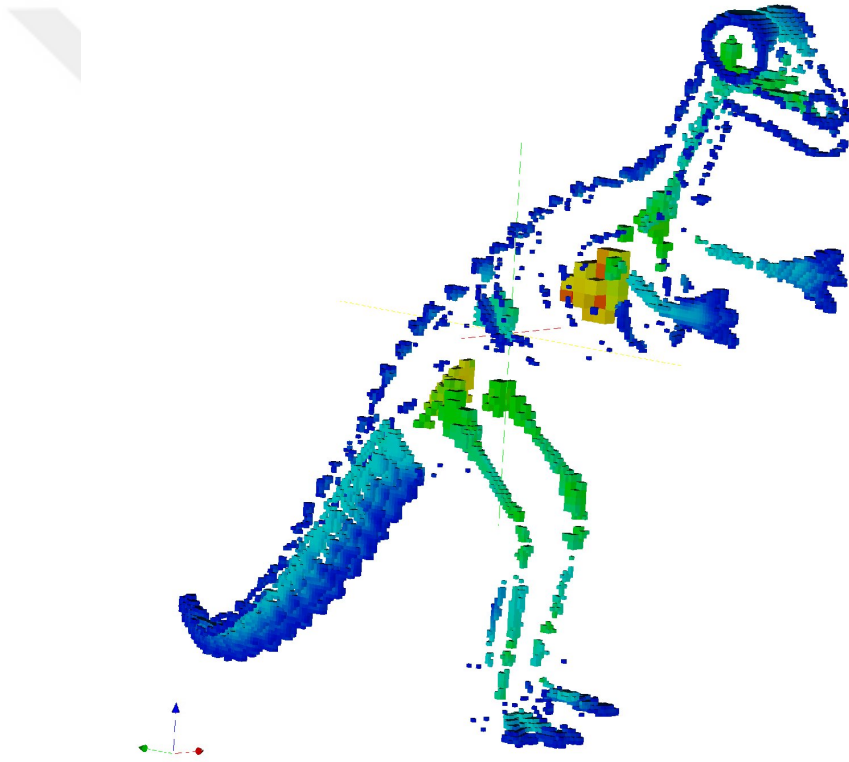


Figure B.1: Skeletonisation with distance transform for Toy dinosaur test case. Skeleton contains 9880 cell from 151042 leaf cells. Coloration shows the distance information.

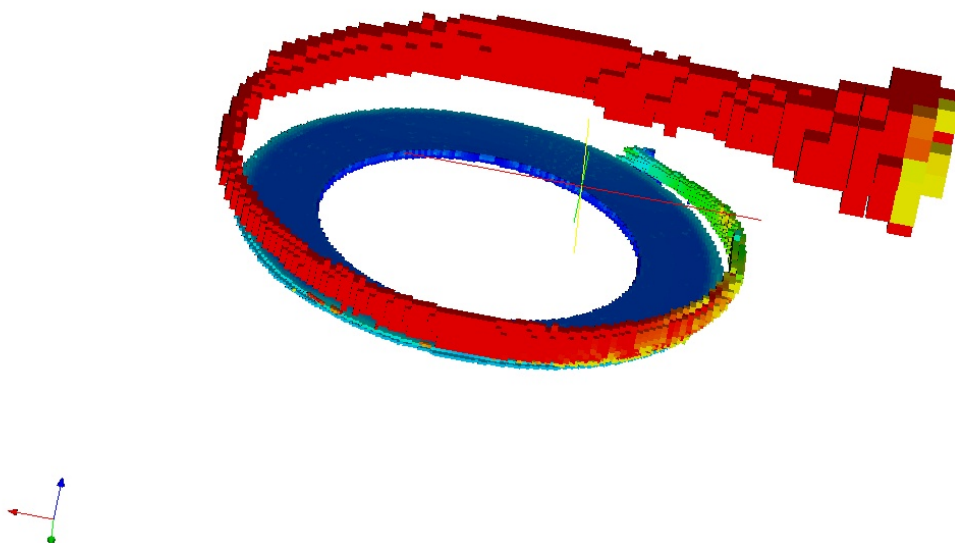


Figure B.2: Skeleton of volute geometry with distance transform.

Bibliography

- [1] R. Schneiders. Open problem. Unpublished web page <http://www-users.informatik.rwth-aachen.de/roberts/open.htm>.
- [2] S. E. Benzley, E. Perry, K. Merkley, B. Clark, and K. Sjaardema. A comparison of allhexahedral and all-tetrahedral finite element meshes for elastic and elasto-plastic analysis. In *4th Int. Meshing Roundtable*, pages 179–191, 1995.
- [3] A.O Cifuentes and A. Kalbag. A performance study of tetrahedral and hexahedral elements in 3-d finite element structural analysis. *Finite Elements in Analysis and Design*, 12:313–318, 1992.
- [4] Anish Malanthara and Walter Gerstle. Comparative study of unstructured meshes made of triangles and quadrilaterals. In *Proceedings of 6th Meshing Roundtable*, 1997.
- [5] Ted Blacker. Meeting the challenge for automated hexahedral meshing. *Engineering With Computers*, 17. Issue 3, 2001.
- [6] M. Halpern. Industrial requirements and practices in finite element meshing: a survey of trends. In *Proceedings of 6th International Meshing Roundtable*, Sandia National Laboratories, Albuquerque, New Mexico, 1997.
- [7] Rainald Löhner. Automatic unstructured grid generation. *Finite Element Analysis and Design*, 25:111–134, 1997.
- [8] Steven J. Owen. *Non-Simplicial Unstructured Mesh Generation*. PhD thesis, The Department of Civil and Environmental Engineering Carnegie Mellon University, 1999.
- [9] Steven Owen. A survey of unstructured mesh generation technology. In *Proceedings of 7th International Meshing Roundtable*, 1998.
- [10] Paul-Louis George and Houman Borouchaki. *Delaunay Triangulation and Meshing*. Hermes, 1998.
- [11] Timothy J. Baker. *Handbook of Grid Generation*, chapter Delaunay-Voronoi Methods, pages 16.1–10. CRC Press, 1990.

-
- [12] Boris N. Delaunay. Sur la sphere vide. *Izvestia Akademia Nauk SSSR, VII Seria, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7:793–800, 1934.
- [13] D Watson. Computing the n -dimensional delaunay tessellation with application to voronoï polytopes. *The Computer Journal*, 24:167–172, 1981.
- [14] M Tanemura, T Ogawa, and N Ogita. A new algorithm for three-dimensional voronoï tessellation. *J. Comp. Physics*, 51:91, 1983.
- [15] C L Lawson. Software for c1 surface interpolation. *Mathematical Software*, III:161–194, 1977.
- [16] Timothy J. Baker. Triangulations, mesh generation and point placement strategies. In *Frontiers of Computational Fluid Dynamics*. John Wiley & Sons, 1994.
- [17] J. P. Wright and A. G. Jack. Aspects of three-dimensional constrained delaunay meshing. *Int. J. Num. Meh. Eng.*, 37, 1994.
- [18] P Chew. Guaranteed quality mesh generation for curved surfaces. In *Proc. 9th Symp. On. Comp. Geom.*, 1993.
- [19] D. Sharov and K. Nakahashi. A boundary recovery algorithm for delaunay tetrahedral meshing. In *Proceedings 5th international conference on Numerical Grid Generation in Computational Field Simulations*, 1996.
- [20] N. P. Weatherhill N.O. Hassan. Efficient three-dimensional grid generation using delaunay triangulation. In *Proc. First European Computational Fluid Dynamics Conference, Brussels*, 1992.
- [21] J. Peraire, J. Peiro, and K. Morgan. *Handbook of Grid Generation*, chapter Advancing Front Grid Generation, pages 17.1–22. CRC Press, 1990.
- [22] H. Jin and R. I. Tanner. Generation of unstructured tetrahedral meshes by advancing front technique. *Int. Jour. Num. Meth. Engineering*, 31:987–997, 1991.
- [23] R. Löhner and P. Parikh. Three-dimensional grid generation by the advancing front method. *Int. J. Num. Meth. Fluids*, 8:1135–1149, 1988.
- [24] K. Nakahashi and D. Sharov. Direct surface triangulation using the advancing front method. Technical report, AIAA-95-1686-CP, 1995.
- [25] S. H. Lo. Volume discretization into tetrahedra-i. verification and orientation of boundary surfaces. *Computers and Structures*, 39(5):493–500, 1991.
- [26] S. H. Lo. Volume discretization into tetrahedra-ii. 3d triangulation by advancing front. *Computers and Structures*, 39(5):501–511, 1991.

- [27] J. Müller, P. L. Roe, and H. Deconinck. A frontal approach for internal node generation in Delaunay triangulations. *Int. J. Num. Meth. Eng.*, 17:241–256, 1993.
- [28] M. Merriam. An efficient advancing front algorithm for delaunay triangulation. Technical report, AIAA-95-0213, 1995.
- [29] D. J. Mavriplis. An advancing front delaunay triangulation algorithm designed for robustness. Technical report, ICASE report, Oct. 1992.
- [30] M. Sabin. Criteria for comparison of automatic mesh generation method. *Advances in Engineering Software*, 13(5-6), 1991.
- [31] Timothy J. Tautges. The generation of hexahedral meshes for assembly geometry: survey and progress. *Int. J. Numer. Meth. Engng*, pages 2617–2642, 2001.
- [32] Steven J. Owen, Matthew L. Staten, Scott A. Canann, and Sunil Saigal. Q-morph: An indirect approach to advancing front quad meshing. *International Journal for Numerical Methods in Engineering*, 44(9):1317–1340, 1999.
- [33] Steven J. Owen and Sunil Saigal. H-morph: An indirect approach to advancing front hex-meshing. *International Journal for Numerical Methods in Engineering*, 1999.
- [34] T. D. Blacker and M. B. Stephenson. Paving: A new approach to automated quadrilateral mesh generation. *Int. j. Numer. Methods in Eng.*, 32:1811–847, 1991.
- [35] R. J. Cass, S. E. Benzley, R. J. Meyers, and T. D. Blacker. Generalized 3-d paving: An automated quadrilateral surface mesh generation algorithm. *International Journal for Numerical Methods in Engineering*, 39:1475–1490, 1996.
- [36] David White and Paul Kinney. Redesign of the paving algorithm: Robustness enhancements through element by element meshing. In *Proceedings, 6th International Meshing Roundtable, Park City, Utah, USA*, 1997.
- [37] T. D. Blacker and R. J. Meyers. Seams and wedges in plastering: A 3-d hexahedral mesh generation algorithm. *Engineering with Computers*, 9(2):83–93, 1993.
- [38] Robert W. Leland, Darryl J. Melander, Ray W. Meyers, Scott A. Mitchell, and Timothy J. Tautges. The geode algorithm: Combining hex/tet plastering, dicing and transition elements for automatic, all-hex mesh generation. *Proceedings, 7th International Meshing Roundtable*, pages 515–521, 1998.
- [39] Steven Benzley, Ted D. Blacker, Scott A. Mitchell, Peter Murdoch, and Timothy J. Tautges. Hexahedral mesh generation via the dual. In *SCG '95: Proceedings of the eleventh annual symposium on Computational geometry*, pages 404–405, New York, NY, USA, 1995. ACM Press.

-
- [40] Timothy J. Tautges and Scott A. Mitchell. Whisker weaving: Invalid connectivity resolution and primal construction algorithm. In *Proceedings, 4th International Meshing Roundtable*, 1995.
 - [41] T. Tautges, T. Blacker, and S. Mitchell. The whisker weaving algorithm: A connectivity-based method for constructing all-hexahedral finite element meshes. *Int. J. Num. Meth. Eng.*, 39:3327–3349, 1996.
 - [42] Peter Murdoch, Steven Benzley, Ted Blacker, and Scott A. Mitchell. The spatial twist continuum: a connectivity based method for representing all-hexahedral finite element meshes. *Finite Elem. Anal. Des.*, 28(2):137–149, 1997.
 - [43] T Blacker, S A Mitchell, T J Tautges, P Murdoch, and S Benzley. Forming and resolving wedges in the spatial twist continuum. *Engineering With Computers*, 13(1):35–47, 1997.
 - [44] Matthias Müller-Hannemann and Karsten Weihe. Hexahedral mesh generation by successive dual cycle elimination. *Engineering with Computers*, 15(3):269–279, 1999.
 - [45] Matthias Müller-Hannemann. Quadrilateral surface meshes without self-intersecting dual cycles for hexahedral mesh generation. *Computational Geometry-Theory and Applications*, 22:75–97, 2002.
 - [46] Nestor A. Calvo and Sergio R. Idelsohn. All-hexahedral element meshing: Generation of the dual mesh by recurrent subdivision. *Computer Methods in Applied Mechanics and Engineering*, 182:371–378, 2000.
 - [47] Nestor A. Calvo and Sergio R. Idelsohn. All-hexahedral element meshing: automatic elimination of self-intersecting dual lines. *International Journal for Numerical Methods in Engineering*, 55:1439–1449, 2002.
 - [48] T. Nathan, T. Folwell, and Scott A. Mitchell. Reliable whisker weaving via curve contraction. In *Proceedings, 7th International Meshing Roundtable*, 1998.
 - [49] H. Blum. *A transformation for extracting new descriptors of shape, Models for the Perception of Speech and Visual Form*. MIT Press, 1967.
 - [50] F. Leymarie and M. D. Levine. Simulating the grassfire transform using an active contour model. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 14(1):56–72, 1992.
 - [51] Peter Sampl. Semi structured mesh generation based on medial axis. In *Proceedings of 9th Meshing Roundtable*, 2000.
 - [52] H. Nebi Gürsoy and Nicholas M. Patrikalalis. An automatic coarse and fine surface mesh generation scheme based on medial axis transform: Part i algorithms. *Engineering With Computers*, 8:121–137, 1992.

- [53] H. Nebi Gürsoy and Nicholas M. Patrikalalis. An automatic coarse and fine surface mesh generation scheme based on medial axis transform: Part ii implementation. *Engineering With Computers*, 8:179–196, 1992.
- [54] H. Nebi Gürsoy and Nicholas M. Patrikalalis. Automated interrogation and adaptive subdivision of shape using medial axis transform. *Advances in Engineering Software*, 13:287–3012, 1991.
- [55] C.G. Armstrong, D. J. Robinson, R.M. McKeag, T.S. Li, and S.J. Bridgett. Medials for meshing and more. In *Proceedings of 4th Meshing Roundtable, Albuquerque, New Mexico*, 1995.
- [56] Mark Price, Clive Stops, and Geoffrey Butlin. A medial object toolkit for meshing and other applications. In *Proceedings of 4th Meshing Roundtable, Albuquerque, New Mexico*, 1995.
- [57] M. A. Price and C. G. Armstrong. Hexahedral mesh generation by medial axis subdivision part i: solids with convex edges. *Int. J. Numer. Methods. Eng.*, 38:3335–33359, 1997.
- [58] M. A. Price and C. G. Armstrong. Hexahedral mesh generation by medial axis subdivision part ii: solids with flat and concave edges. *Int. J. Numer. Methods. Eng.*, 40:111–136, 1997.
- [59] Alla Sheffer and Michel Bercovier. Hexahedral meshing of non-linear volumes using voronoi faces and edges. *Int. J. Numer. Meth. Engng.*, 49:329–351, 2000.
- [60] Ted Blacker. The cooper tool. In *Proceedings of the 5th International Meshing RoundTable, Pittsburgh, Pennsylvania, U.S.A.*, pages 13–30, 1996.
- [61] Katsuhiro Miyoshi and Ted D. Blacker. Hexahedral mesh generation using multi-axis cooper algorithm. In *Proceedings, 9th International Meshing Roundtable*, pages 89–97, 2000.
- [62] Steven R. Jankovich, Steven E. Benzley, Jason F. Shepherd, and Scott A. Mitchell. The graft tool: An all-hexahedral transition algorithm for creating a multi-directional swept volume mesh. In *Proceedings, 8th International Meshing Roundtable*, pages 387–392, 1999.
- [63] Mingwu Lai, Steven E. Benzley, Greg Sjaardema, and Tim Tautges. A multiple source and target sweeping method for generating all hexahedral finite element meshes. In *Proceedings of 5th Meshing Roundtable, Pittsburgh, Pennsylvania, USA*, 1996.
- [64] Mingwu Lai, Steven Benzley, and David White. Automated hexahedral mesh generation by generalized multiple source to multiple target sweeping. *Int. J. Numer. Meth. Engng.*, 49:261–275, 2000.

- [65] Patrick M. Knupp. Next-generation sweep tool: A method for generating all-hex meshes on two-and-one-half dimensional geometries. In *7th Int. Meshing Roundtable, Dearborn, Michigan*, 1998.
- [66] Matthew L. Staten, Scott A. Canann, and Steven J. Owen. Bmsweep: Locating interior nodes during sweeping. *Eng. Comput. (Lond.)*, 15(3):212–218, 1999.
- [67] Michael Borden, Steven Benzley, Scott A. Mitchell, David R. White, and Ray Meyers. The cleave and fill tool: an all-hexahedral refinement algorithm for swept meshes. In *Proceedings, 9th International Meshing Roundtable*, 2000.
- [68] Scott Mitchell and Timothy J. Tautges. Pillowing doublets: Refining a mesh to ensure that faces share at most one edge. In *Proceedings of 4th Meshing Roundtable, Albuquerque, New Mexico*, 1995.
- [69] David R. White and Timothy J. Tautges. Automatic scheme selection for toolkit hex meshing. *Int. J. Numer. Meth. Engng*, 49:127–144, 2000.
- [70] S. Subrahmanyam and M. Wozny. An overview of automatic feature recognition techniques for computer-aided process planning. *Computers in Industry*, 26:1–21, 1995.
- [71] David White, Lai Mingwu, Steven Benzley, and Gregory Sjaardema. Automated hexahedral mesh generation by virtual decomposition. In *Proceedings of 4th Meshing Roundtable, Albuquerque, New Mexico*, 1995.
- [72] Bih-Yaw Shih and Hiroshi Sakurai. Automated hexahedral mesh generation by swept volume decomposition and recomposition. In *Proceedings of 5th Meshing Roundtable, Pittsburgh, Pennsylvania, USA*, 1996.
- [73] Shang-Sheng Liu and Rajit Gadh. Basic logical bulk shapes (blobs) for finite element hexahedral mesh generation. In *Proceedings of 5th Meshing Roundtable, Pittsburgh, Pennsylvania, USA*, 1996.
- [74] Shang sheng Liu and Rajit Gadh. Automatic hexahedral mesh generation by recursive convex and swept volume decomposition. In *Proceedings of 6th Meshing Roundtable*, 1997.
- [75] Shang-Sheng Liu, J. Uicker Jr., and Rajit Gadh. A dual geometry - topology constraint approach for determination of pseudo-swept shapes as applied to hexahedral mesh generation. *Computer-Aided Design*, 31(6):413–426, 1999.
- [76] Y. Lu, R. Gadh, and T.J. Tautges. Feature based hex meshing methodology: feature recognition and volume decomposition. *Computer-Aided Design*, 33:221–232, 2001.

- [77] N. Chiba, I. Nishigaki, Y. Yamashita, C. Takizawa, and K. Fujishiro. An automatic hexahedral mesh generation system based on the shape-recognition and boundary-fit methods. In *Proceedings of 5th Meshing Roundtable, Pittsburgh, Pennsylvania, USA*, pages 281–290, 1996.
- [78] N. Chiba, I. Nishigaki, Y. Yamashita, C. Takizawa, and K. Fujishiro. A flexible automatic hexahedral mesh generation by boundary-fit method. *Comput. Methods Appl. Mech. Engrg.*, 161:145–154, 1998.
- [79] Y. Su and A. Senthil Kumar. Block cartesian abstraction of a geometric model and its application in hexahedral mesh generation. *Computer-Aided Design*, 37:899–907, 2005.
- [80] M. Delanaye, A. Patel, and C. Hirsch B. Leonard. Automatic unstructured hexahedral grid generation and flow solutions. *proceedings ECCOMAS CFD 2001, Institution of Mathematics and its Application, Southend-on-Sea, UK*, 2001.
- [81] Robert Schneiders and Rolf Bünten. Automatic generation of hexahedral finite element meshes. *Computer Aided Geometric Design*, 12(7):693–707, 1995.
- [82] Robert Schneiders. Octree-based hexahedral mesh generation. *Int. J. Comput. Geometry Appl.*, 10(4):383–398, 2000.
- [83] K. Tchou, C. Hirsch, and R. Schneiders. Octreebased hexahedral mesh generation for viscous flow simulations. *13th AIAA Computational Fluid Dynamics Conference*, AIAA-971980., 1997.
- [84] K. Walton, S. Benzley, and J. Shepherd. Sculpting: An improved inside-out scheme for all-hexahedral meshing. *Proceedings of the 11th International Meshing Roundtable, Sandia National Laboratories*, pages 153–159, 2002.
- [85] Loic Marechal. A new approach to octree based hexahedral meshing. In *10th International meshing roundtable*, 2001.
- [86] Ko-Foa Tchou, Julien Dompierre, and Ricardo Camarero. Automated refinement of conformal quadrilateral and hexahedral meshes. *International Journal for Numerical Methods in Engineering*, 59:1539–1562, 2004.
- [87] Ko-Foa Tchou, M. Khachan, F. Guibault, and Ricardo Camarero. Constructing anisotropic metrics using octrees and skeletons. In *12th International Meshing Roundtable*, 2003.
- [88] Steven E. Benzley Harris, Nathan J. and Steven J. Owen. Conformal refinement of all-hexahedral element meshes based on multiple twist plane insertion”. In *Proceedings, 13th International Meshing Roundtable*, 2004.

-
- [89] Dae-Young Kwak and Yong-Taek Im. Hexahedral mesh generation for remeshing in three-dimensional metal forming analyses. *Journal of Materials Processing Technology*, 138:531–537, 2003.
- [90] Y. Su, K. H. Lee, and A. Senthil Kumar. Automatic hexahedral mesh generation for multi-domain composite models using a hybrid projective grid-based method. *Computer-Aided Design*, 36:203–215, 2004.
- [91] Stefan P. Spekreijse. Elliptic generation systems. In *Handbook of Grid Generation*, chapter 4. CRC Press, 1999.
- [92] Ahmed Khamayseh, Andrew Kuprat, and C. Wayne Mastin. Boundary orthogonality in elliptic grid generation. In *Handbook of Grid Generation*, chapter 6. CRC Press, 1999.
- [93] N. Jones and S. Wright. Algorithm for smoothing triangulated surfaces. *Journal of Computing in Civil Engineering*, ASCE 1:85–102, 1991.
- [94] D. Field. Laplacian smoothing and delaunay triangulations. *Communications and Applied Numerical Methods*, 4:709–712, 1988.
- [95] P. Hansbo. Generalized laplacian smoothing of unstructured grids. *Communications in Numerical Methods in Engineering*, 11:455–464, 1995.
- [96] 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.
- [97] 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.
- [98] 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.
- [99] J. Batina. Unsteady Euler spring method for three-dimensional unstructured dynamic meshes. *AIAA Paper*, 89-0150, 1989.
- [100] J. Batina. Unsteady Euler airfoil solutions using unstructured dynamic meshes. *AIAA Journal*, 28(8):1381–1388, 1990.
- [101] T. Tezduyar, M. Behr, and J. Liou. A new strategy for finite element computations involving moving boundaries and interfaces-the deforming spatial domain/space-time procedure: I. The concept and the preliminary numerical tests. *Comput Meth Appl Mech Eng*, 94:339–351, 1993.

- [102] 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.
- [103] C Farhat, M Lesoinne, and P. LeTallec. Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: momentum and energy conservation, optimal discretization and application to aeroelasticity. *Comput Meth Appl Mech Eng*, 157:95–114, 1998.
- [104] P. J. Slikkeveer, E. P. van Loohuizen, and S. B. G. O’Brien. An implicit surface tension algorithm for Picard solvers of surface-tension-dominated free and moving boundary problems. *International Journal of Numerical Methods of Fluids*, 22:851–865, 1996.
- [105] Wagdi G. Habashi, Julien Dompierre, Yves Bourgault, Djaffar Ait-Ali-Yahia, Michel Fortin, and Marie-Gabrielle Vallet. Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD, Part I: General principles. *International Journal For Numerical Methods in Fluids*, 32:725–744, 2000.
- [106] S. R. Chakravarthy and K. Y. Szema. Computational fluids dynamics capability for internally carried store separation. Sc-71039-tr, Rockwell Int, 1991.
- [107] C. Degand and C. Farhat. A three-dimensional torsional spring analogy method for unstructured dynamic meshes. *Computers and Structures*, 80:305–316, 2002.
- [108] 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.
- [109] 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.
- [110] 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.
- [111] L. Freitag and C. Ollivier-Gooch. Tetrahedral mesh improvement using swapping and smoothing. *International Journal For Numerical Methods In Engineering*, 40:3979–4002, 1997.
- [112] Marshall Bern and David Eppstein. Flipping cubical meshes. In *Proceedings 10th International Meshing Roundtable, Newport Beach, California, USA*, 2001.

- [113] Numeca International, Av. Fr. Roosevelt, 5 Brussels. *HEXPRESS Mesh Generator Manual*, v. 2.2 edition.
- [114] O. Hassan, E. J. Probert, and K. Morgan. Unstructured mesh procedures for the simulation of three-dimensional transient compressible inviscid flows with moving boundary problems. *International Journal For Numerical Methods in Fluids*, 27:41–55, 1998.
- [115] Djaffar Ait-Ali-Yahia, Guido Baruzzi, Wagdi G. Habashi, Michel Fortin, Julien Dompierre, and Marie-Gabrielle Vallet. Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD, Part II: Structured grids. *International Journal For Numerical Methods in Fluids*, 39:657–673, 2002.
- [116] Julien Dompierre, Marie-Gabrielle Vallet, Yves Bourgault, Michel Fortin, and Wagdi G. Habash. Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part III. Unstructured meshes. *International Journal for Numerical Methods in Fluids*, 39, Issue 8:675 – 702, 2002.
- [117] A. Tam, Djaffar Ait-Ali-Yahia, M. P. Robichaud, M. Moore, V. Kozel, and Wagdi G. Habashi. Anisotropic mesh adaptation for 3d flows on structured and unstructured grids. *Comput. Methods. Appl. Mech Engrg*, 189:1205–1230, 2000.
- [118] Frederic J. Blom. Considerations on the spring analogy. *International Journal For Numerical Methods in Fluids*, 32:647–668, 2000.
- [119] Cyril Gruau. *Génération de métriques pour adaptation anisotrope de maillages, applications á la mise en forme des matériaux*. PhD thesis, L’École Nationale Supérieure des Mines de Paris, 2004.
- [120] C. Gruau and T. Coupez. Anisotropic and multidomain mesh automatic generation for viscous flow finite element method. In *Adaptive Modeling and Simulation, International Center for Numerical Methods in Engineering (CIMNE)*, 2003.
- [121] R. Garimella and M. Shephard. Boundary layer meshing for viscous flows in complex domains. In *7th International Meshing Roundtable*, 1998.
- [122] J. Peraire and K. Morgan. Unstructured mesh generation including directional refinement for aerodynamic flow simulation. *Finite Elements in Analysis and Design*, 25:343–355, 1997.
- [123] Cyberware. <http://www.cybeware.com/samples/index.html>. Sample Models.
- [124] Ron Kimmel, Doron Shaked, Nahum Kiryati, and Alfred M. Bruckstein. Skeletonization via distance maps and level sets. *Computer Vision and Image Understanding*, 62:382–391, Nov. 1995.

- [125] Charles Hirsch. *Numerical Computation of Internal and External Flows. Volume 1: Fundamentals of Numerical Discretization*. John Wiley & Sons, 1988.
- [126] Charles Hirsch. *Numerical Computation of Internal and External Flows. Volume 2: Computational methods for inviscid and viscous flows*. John Wiley & Sons, 1988.
- [127] G. Redeker. DLR-F4 wing body configuration. Technical report, AGARD Advisory report No 303 A Selection of Experimental test cases for the validation of CFD Codes, 1994.
- [128] J Dunham. CFD validation for propulsion system components. Technical report, AGARD-AR-355, 1995.
- [129] P. R. Spalart and S. R. Allmaras. A one-equation turbulence model for aerodynamic flows. *AIAA Paper*, 92-0439.
- [130] D. W. Levy, T. Zickuhr, J. Vassberg, S. Agrawal, R. A. Wahls, J. Pirzadeh, and M. J. Hemsh. Summary of data from the first aiaa cfd drag prediction workshop. Technical report, *AIAA Paper* 2002-0841, 2002.
- [131] Alpesh Patel. *Development of an adaptive RANS solver for unstructured hexahedral meshes*. PhD thesis, Université Libre de Bruxelles Faculté des Sciences Appliquées, March 2003.
- [132] R. H. Landon. NACA 0012. oscillatory and transient pitching. Technical report, AGARD Report 702, Compendium of Unsteady Aerodynamic Measurements, Data Set 3, August 1982.
- [133] N. Hakimi. *Preconditioning Methods for Time Dependent Navier-Stokes Equations. Application to Environmental and Low Speed Flows*. PhD thesis, VUB, Brussels, Belgium, December 1997.
- [134] P. Eliasson. and J. Nordström. The development of an unsteady solver for moving meshes. Technical report, FFA TH 1995-39, 1995.
- [135] E.R. Davies and A.P.N. Plummer. Thinning algorithms: A critique and a new methodology. 14(1-6):53–63, 1981.
- [136] L. Lam, S.W. Lee, and C.Y. Suen. Thinning methodologies: A comprehensive survey. 14(9):869–885, September 1992.
- [137] L. Lam and C.Y. Suen. An evaluation of parallel thinning algorithms for character-recognition. 17(9):914–919, September 1995.

Index

- advancing front, 10
 - hexahedral, *see* plastering
 - quadrilateral, *see* paving
- advancing front
 - triangulation, 11
- anisotropy
 - local, 57
- Bresenham's algorithm, 99
- bubble growth, 58
 - metric, 62
- bubble quality criteria
 - for hexahedral meshes, 67
 - for tetrahedral meshes, 66
- CFD , *see* computational fluid dynamics
- computational fluid dynamics, 5
- conformal refinement, 28
- conformal refinement
 - hexahedral, 28
- conformisation, 28
- cooper tool, 23
- cooper tool
 - multi-axis, 23
- decomposition automatic, 25
- Delaunay
 - criterion, 7
 - meshing, 9
 - meshing
 - constrained, 9
 - tetrahedralization, 8
 - triangulation, 8
- Delaunay, Boris, 7
- detail suppression, 20
- digital skeleton, 72
- distance function, 82
 - modified, 88
- distance function
 - city-block type, 82, 90
- distance map, 82
- Distance Map metric, 88
- DLR-F4
 - mesh generation on, 105
- DLR-F4
 - simulation, 115
- dual mesh, 17
- elliptic metric, 48
- factor of maximum anisotropy, 96
- feature recognition, 25
- Flipping, 8
- geometry awareness, 36
- grid based method, *see* overlay grid
- hanging node, 44
- Hessian matrix, 86
- HEXPRESS, 39
- HEXSTREAM, 117
- Hooke's Law, 41
- initial grid, 29
- Laplace equation, 30
- lift coefficient, 127
- Mach number
 - relative, 115
- MAT, *see* medial axis transform
- medial axis, 19
- medial axis
 - Voronoi graph based, 20, 21

medial axis transform, 19, 55
medial surface, 19
medial surface
 reconstruction, 22
mesh quality, 5
modified distance function, 88

NACA0012, 126
natural layout
 definition, 55

optimisation, 33
oscillating airfoil, 126
over-relaxation, *see* relaxation
Overlay grid, 26

Paving, 15
physics based mesh control, 31
plastering, 15
Poisson equation, 31

reduced frequency, 127
refinement
 conformal, 34
 diffusion, 34
 non-conformal, 34
relaxation factor, 41
Riemannian structure, 49

seaming, 16
segment-spring, 42
segmentation, 26
skeleton, 72
skeleton
 digital, 72
skeletonisation, 19, 72
 by thinning, 85
skeletonisation
 by distance transform, 139
 thinning, 72
source surface, 22
spatial twist continuum, 16
spring analogy, 31, 39
spring analogy
 boundary treatment, 43
 non-conformal meshes, 43
STC, *see* spatial twist continuum
stiffness, 41
stiffness
 weight factor, 45
stiffness function, 42
Swapping, 8
Sweeping, 22
swept surface, 22

torsional springs analogy, 32
whisker-weaving, 16