Universitat Politècnica de Catalunya
ETSECCPB

Treball Fi de Màster

Simultaneous Untangling and Smoothing of Hexahedral Meshes

by

César Augusto Rivas Guerra

Advisor: Josep Sarrate Ramos

Barcelona, February 2010

‘‘*Carpe diem.* `Seize the day, boys.`
`Make your lives extraordinary.`’’
–John Keating (Dead Poets Society, 1989)

Abstract

Simultaneous Untangling and Smoothing of Hexahedral Meshes

César Augusto Rivas Guerra

Currently, there is not a general method such that any given geometry can generate a mesh with elements of hexahedral type. However, there are several methods used to create the numerical discretization for certain types of geometries. Unfortunately, these methods could generate meshes with highly distorted elements and in some cases the mesh obtained may include tangled elements. Therefore, it is of most importance to develop a procedure that could smooth and untangle hexahedral meshes.

This work will provide a method that improves the quality of unstructured meshes and, if required, untangle the *inverted* elements by the minimization of a smoothed objective function. This objective function is based on the shape quality index of the elements or in the conditioning number of the shape matrix. To illustrate the application of the proposed smoother, several example of tangled meshes are presented, showing how the smoother untangles and smoothes quadrilateral and hexahedral meshes. A comparison of the performance between several minimization methods is presented. Finally, the robustness of smoother is compared to one of the most used smoothing methods.

Keywords: Hexahedral mesh smoothing; Hexahedral mesh untangling; Hexahedral mesh generation; Adaptive meshes; Finite elements: Minimization methods.

Acknowledgments

First and foremost, I would like to thank my thesis advisor, Josep Sarrate Ramos, for sharing his knowledge, comments, suggestions and for his unconditional support during this investigation. I truly appreciate your invitation to be a mesh generation *rock-star*, but for the time being, I will be happy as an amateur.

I would like to thank to Xevi Roca. For your comments, suggestions and for the time you took to explain me about the inner working of `C++` and how to make my code faster. Also, I would like to thank the other members of `EZ4U` team, Eloi Ruiz and Joan Carreras.

A big thank you to all my friends at `LaCàN`, `CIMNE` and `UPC`. Too many names to write down, but I know who you are. Thank you, guys!!

I would like to acknowledge the administrative support of Lelia Zielonka and Adriana Stilmann, from `CIMNE`. Also, I would like to thank the `LaCàN` staff for the support provided for the creation of this work.

Finally, I would like to thank my family, for their support and motivation.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Simulations in engineering and sciences are performed with a numerical method that in most of the cases requires a *mesh*. The *mesh* is a numerical discretization of the spatial domain. It is obtained by splitting the domain into elements.

Several type of elements could be used to generate the mesh, being triangular, tetrahedral, quadrilateral and hexahedral the most popular elements used for the discretization of the domain. The algorithms for creating triangular or tetrahedral meshes are matured during the last decade. Nowadays, these algorithms are robust and able to generate million of elements per seconds. Using tetrahedral elements to perform a numerical simulation may be appropriate in some scenarios but not for all (Staten, 2007).

However, quadrilateral and hexahedral elements are preferred by many researchers for several reasons. For instance, being less stiffness and avoid locks in bending, less elements required for obtaining the same accuracy, better alignment of fibers for composite materials and stretched quadrilateral or hexahedral elements have better performance, to than triangular or tetrahedral elements (Roca, 2009).

Currently, there is not a general method such that given any geometry it can generate a mesh with elements of hexahedron type. However, there are several methods used to create the discretization for certain types of geometries. Unfortunately, these methods could generate meshes with highly distorted elements, which in turn can dramatically limit the precision of further calculations. Indeed, in some cases the mesh obtained may include tangled elements. In general, hexahedral meshes is a complicated process in comparison to the one for tetrahedral meshes, that may lead to the generation of elements with lower quality index.

It is therefore of the most importance to have a tool that given an initial mesh allows not only to improve the quality of its elements (getting less distorted elements) but also unravel or untangled the elements. In order to increase the quality index of these distorted elements, a process that smoothes the mesh (change to location of the inner nodes without modifying its connectivity) may be called upon the creation of the mesh. This processes will increase the quality of the worst elements (elements with low quality index) modifying the coordinates of their nodes, to obtain a better suited mesh for given simulation.

1.2 Objectives

This thesis aims to develop a method to improve the quality of a mesh created with hexahedral (or quadrilateral) elements by changing its geometry (i.e., the location of the nodes can change) without altering its topology (it can not modify the connectivity of the mesh, i.e., reconnect the elements). This method is based on the minimization of an objective function which is defined from an algebraic measure of the quality of the element.

Previous related investigations aim to smooth or untangle hexahedral (or quadrilateral) elements but do not try to smooth and untangle the elements during the same process, which is accomplished by this investigation. Knupp (2001) proposes the

shape quality index based on algebraic measures for triangular and tetrahedral elements. Knupp (2003a) extends it for quadrilateral and hexahedral elements. Based in his previous works, Knupp (2003b) proposes a smoothing method based on the optimization of an objective function, which is based in the shape quality index of the elements. An implementation for smoothing and untangling triangular and tetrahedral elements is proposed by Escobar et al. (2003) based on the shape quality index (Knupp, 2001).

1.3 Scope

The scope of the current investigation is to provide a method that allows for smoothing and untangling of hexahedral (and quadrilateral) meshes by changing the geometry of the elements that conforms the mesh. The coordinates of the nodes, that form the elements, are to be moved by a minimization algorithm, but the method will not try to modify the connectivity of the elements. The proposed method should be robust enough to untangled meshes. In addition, the method should also avoid to tangle complex convex meshes.

For quadrilateral elements, the space will be limited to $\mathbb{R}^2$, meaning that all nodes will be located in the $x - y$ plane and its movement will be limited to that given plane. Therefore, the movement of the nodes in the z axis is no allowed. For hexahedral elements, the space will be $\mathbb{R}^3$. The nodes will be free to move to any point in the $3D$ space. The proposed method will be limited to minimize the distortion of the mesh with only one type of elements. Smoothing and untangling mixed elements meshes is out of the scope of this work.

Chapter 2

Shape-based quality measures for quadrilateral and hexahedral elements

In this chapter, the shape-based metrics to measure the quality of a given element will be described. First, the requirements for this metric are revised. Second, the construction process of this metric is reviewed.

2.1 Requirements of the quality index

There are two points of view to measure the mesh quality. The first one is from the mesh generator user. The other, is from the developer of the algorithms of mesh generation.

There are several quality quantities that an analyst must keep in mind while creating (or checking) the computational model (see Thompson et al., 1999; Knupp and Steinberg, 1993). These are:

Jacobian. Represents a measure of how distorted is the element in comparison to a reference element. Values ranges from 0.0 (completely distorted element) to 1.0

(perfect element). For practical purposes, a warning should rise when the value gets around 0.7.

Aspect Ratio. Gives a ratio of the largest side over the smallest side of the element.

Element Length. Gives information about the distribution of the length of the elements in a given model.

Skew. Shows how deformed an element is, by taking into account that the edges of the element are not perpendicular.

Internal Angles. Measure the maximum and minimum angles between two edges sharing the a given node of the element. Depending on the application, the minimum angle should be around 15-20° and the maximum angle between 155-165°

Area/Volume. Provides information about the distribution of element areas or volumes over the model

Normals. The normals of the 2D element on a component should be oriented in the same direction.

Recall that these quantities may have units (length, area or degrees).

Canann et al. (1993) and Knupp (2001) present some requirements for quality index for a given element. These requirements are:

- The quantities reflecting the quality index should to be dimensionless.
- The quantities should not depend of the element or node index. The mesh numbering scheme should not have an effect on the element quality.
- The quantities should not reflect changes in the translation or rotation of the element. The element could undergo rigid body motion with having no effect on the quality metric. The quantity should be scale and orientation invariant.

- The quantities should have a ranges from 0.0 to 1.0. If the quantity is equal to 1.0, the element represent an ideal element and if it is equal to 0.0, the element is totally degenerated.

- The function representing the quality of the element should be a continuous over the domain of the element.

These quantities depend on the geometry of the element or on the location of the nodes to whom the element is attached. Similar to the quality indexes used by analyst, these value represent an index of the shape, size, skew and any combination of the latter, as a metric that combines information with regard of the shape and the skew of the element.

George and Borouchaki (1999) and Knupp (2003a, 2001) present several alternatives to compute dimensionless mesh quality indexes for several types of elements (triangular, tetrahedral, quadrilaterals and hexahedral).

Oddy et al. (1988) presented a quality measure based on one of the components of the Cauchy-Green deformation tensor which is computed from an normalized jacobian, which was used by Canann et al. (1993) for a mesh smoother.

Several algorithms have been developed to increase the quality of the mesh. However optimization techniques or algorithms based on the quality of the elements are used to reduce the number of "bad" elements, as the proposed by Canann et al. (1993) and Knupp (2003b).

2.2 Reference and ideal elements

Before going into the shape-base quality measure, the reference and ideal elements must be defined. The quality index will be defined by two affine mappings. The first mapping is from the *reference* element ($\mathbf{x}_r$) to the *ideal* element ($\mathbf{x}_i$), which is label as

W. The second is an affine mapping from the *reference* element ($\mathbf{x}_r$) to the *"physical"* element ($\mathbf{x}$), label as $\mathbf{A}$ (see Figure 2.1). The affine mapping from the *ideal* ($\mathbf{x}_i$) to the *physical* ($\mathbf{x}$) element is the shape matrix of the element, label as $\mathbf{S}$. (Knupp, 2003a; Escobar et al., 2003).

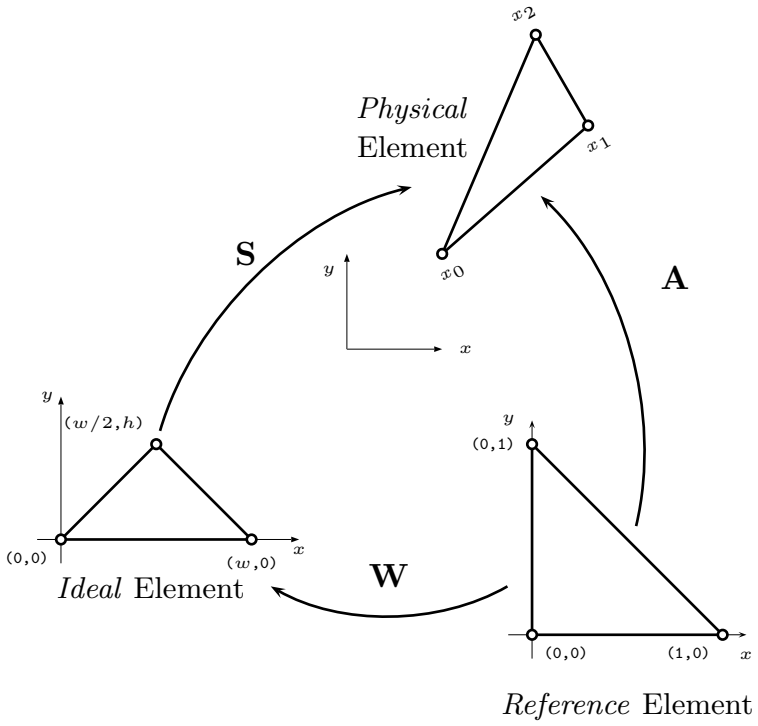


Figure 2.1: Affine mappings for triangular elements

The affine mapping from the *reference* to the *physical* element, defined as:

$$\mathbf{x} = \mathbf{A}\mathbf{x}_r - \mathbf{x}_0 \tag{2.1}$$

$\mathbf{A}$ denotes the jacobain matrix and $\mathbf{x}_0$ is the vector with the coordinates of node 0. The mapping from the *reference* to the *ideal* element $\mathbf{W}$ is given as:

$$\mathbf{x}_i = \mathbf{W}\mathbf{x}_r \tag{2.2}$$

The shape matrix (**S**) or the mapping from the *ideal* to the *physical* element, and it is computed as:

$$\mathbf{S} = \mathbf{A}\mathbf{W}^{-1} \tag{2.3}$$

The *ideal* element represent the shape that will provide a quality index equal to 1.0. Ideally, the mesh should be composed only of *ideal* elements. The shape of the *ideal* element depends on the phenomena that we would like to capture.

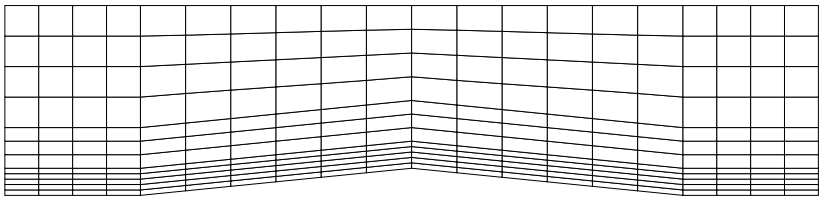
Figure 2.2(a) shows a mesh used in a *CFD*[1] simulation. Notice that elements near the lower boundary are stretched to properly capture the boundary layer. Meanwhile, Figure 2.2(b) shows a representation of a finite element mesh, in which the aspect ratio of the elements trends to 1.0. As an example, in a *CFD* simulation, the elements near a boundary will be very stretched, to properly capture the boundary layer. Elements far away of the boundary will have a aspect ratio close to 1.0, the shape of the elements will be close to a square. The ideal elements used near the boundary can be different than the one far away of the boundary.

2.3 Shape-based quality measures

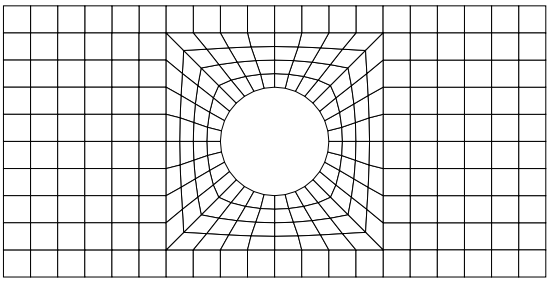
Knupp (2003b) lists the most important metrics about a element are: size, shape and a *invertibility* term. The latter means that the element has positive and non-zero *local* length, area or volume. If a inverted element is added into a mesh used in a structural analysis, that tangled element will have a negative contribution to the *total* stiffness matrix of the domain and the direction of the principal axis of the element will be inverted in comparison to the other elements in the mesh.

The size parameter refers to the quality of the discretization of the domain. If the discretization is too coarse (very large elements), the simulation may not capture the desired precision, and if the mesh is too fine (very small elements), the computational

[1] Acronym for Computational Fluid Dynamics



(a)



(b)

Figure 2.2: Difference between meshes depending on type of simulation (a) *CFD* simulation (b) Structural analysis

cost of the simulation will be too expensive for practical purposes. The shape refers to the distortion of the elements which is a combination of the aspect ratio and the skewness of the elements (see Section 2.1).

Knupp (2001, 2003a) proposes the shape metric (η) that provides information about the distortion of the element. This metric is based on the shape matrix ($\mathbf{S}$), see Equation (2.3), the mapping between the *ideal* and the *physical* element. Note that the definition of the shape matrix depends on the type of element (triangular, tetrahedral, quadrilateral or hexahedral).

2.3.1 Triangular and tetrahedral elements

The shape metric (η) for triangular and tetrahedral elements is defined as

$$\eta = \frac{|\mathbf{S}|^2}{n\sigma^{2/n}} \tag{2.4}$$

where $|\mathbf{S}| = \sqrt{\text{tr}(\mathbf{S}^T\mathbf{S})} = \sqrt{(\mathbf{S},\mathbf{S})}$ is the Frobenius norm, which provides information about the condition number of the shape matrix($\mathbf{S}$), σ is the determinant of the shape matrix, $\sigma = \det(\mathbf{S})$ and n is the dimension of the domain ($n = 2$ for triangular and quadrilateral elements and $n = 3$ for tetrahedra and hexahedra). See Knupp (2003a) or Escobar et al. (2003) for details.

The shape metric (η) has minimum value of 1.0 for the *ideal* elements and maximum value of infinity (∞) for elements with *null* area or volume.

The *ideal*, *reference* and *physical* elements for a triangular mesh are shown in Figure 2.3. Note that the *ideal* element could change depending on the requirements of the simulation.

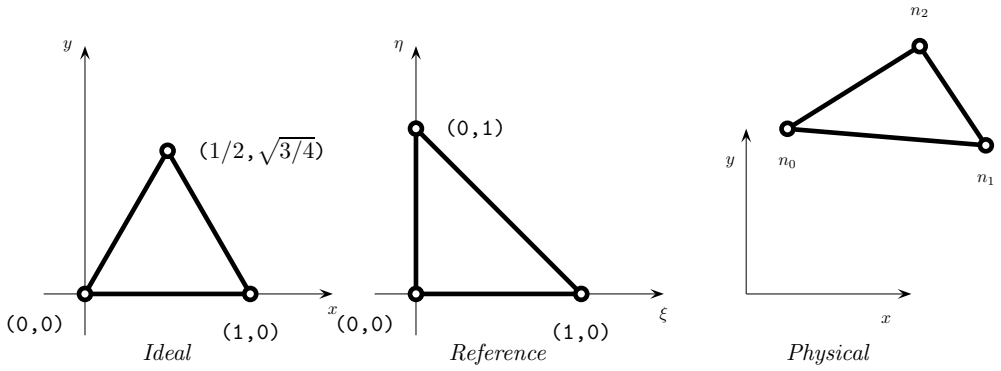


Figure 2.3: *Physical*, *reference* and *ideal* elements for triangular element

For tetrahedral elements, the *ideal* element is defined as an equilateral tetrahedron.

Therefore, we have :

$$\boldsymbol{A} = \begin{bmatrix} x_1 - x_0 & x_2 - x_0 & x_3 - x_0 \\ y_1 - y_0 & y_2 - y_0 & y_3 - y_0 \\ z_1 - z_0 & z_2 - z_0 & z_3 - z_0 \end{bmatrix} \tag{2.5}$$

$$\boldsymbol{W} = \begin{bmatrix} 1.0 & 1/2 & 1/2 \\ 0.0 & \sqrt{3/4} & \sqrt{3}/6 \\ 0.0 & 0.0 & \sqrt{2/3} \end{bmatrix} \tag{2.6}$$

For triangular elements, the *ideal* element is defined as an equilateral triangle. Therefore, we have :

$$\boldsymbol{A} = \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix} \tag{2.7}$$

$$\boldsymbol{W} = \begin{bmatrix} 1.0 & 1/2 \\ 0.0 & \sqrt{3/4} \end{bmatrix} \tag{2.8}$$

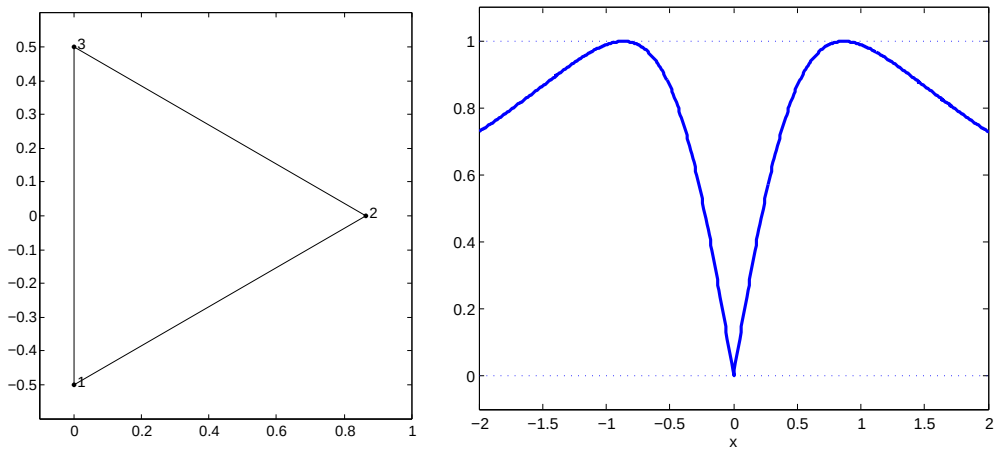
Knupp (2003a) defines the shape quality index as

$$q = \frac{1}{\eta} \tag{2.9}$$

where shape metric (η) is defined by Equation (2.4). The shape quality index (q) can only take values from 0.0 to 1.0 ($1 \leq \eta \leq \infty$), fulfilling all of the requirements for the quality index presented in Section 2.1. Note that the shape quality index (q) and the shape metric (η) are scalars ($\mathbb{R}$) created by the element spatial information ($\mathbb{R}^3$).

Figure 2.4(a) depictures a mesh composed by three nodes, being $\mathbf{x}_1 = (0, -1/2)$ and $\mathbf{x}_3 = (0, 1/2)$ fixed. Node $\mathbf{x}_2$ is allowed to move in the x coordinate from $x = 2.0$ to $x = -2.0$ (keeping the y coordinate equal to 0.0). The shape quality index q of the element conformed by nodes $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ is shown in Figure 2.4(b).

The minimum value obtained for the quality index, $q = 0$, is found at $x = 0.0$ where the element has a *null* area and all of the edges of the element are co-linear. Note that for $\mathbf{x}_2 = (0, 0)$, the shape metric takes the value of infinity ($\eta = \infty$). For $x = \pm\sqrt{3/4}$, the quality index is equal to 1.0, but only $x = \sqrt{3/4}$ fulfills the *invertibility* of the element. For $x = -\sqrt{3/4}$, the element in tangled and has a negative *local* area. In fact, for all $x \leq 0.0$, the area of the element is negative; the element is inverted or tangled.



(a) (b)

Figure 2.4: Shape quality index for a triangular element

A mesh with one tetrahedron is now considered. In this case, nodes $\mathbf{x}_1 = (0, 0, 0)$, $\mathbf{x}_2 = (1, 0, 0)$ and $\mathbf{x}_3 = (1/2, \sqrt{3/4}, 0)$ are fixed. Node $\mathbf{x}_4$ is allowed to move along the z axis. Figure 2.5 shows the shape quality index (q) of the tetrahedron.

The minimum value obtained for the quality index, $q = 0$, is found at $z = 0.0$ where the element has a null area and all of the edges and faces of the tetrahedron are co-linear. Note that for $\mathbf{x}_4 = (0, 0, 0)$, the shape metric takes the value of infinity ($\eta = \infty$).

The quality index for the tetrahedron has two maxima located at $z = \pm\sqrt{2/3}$, but

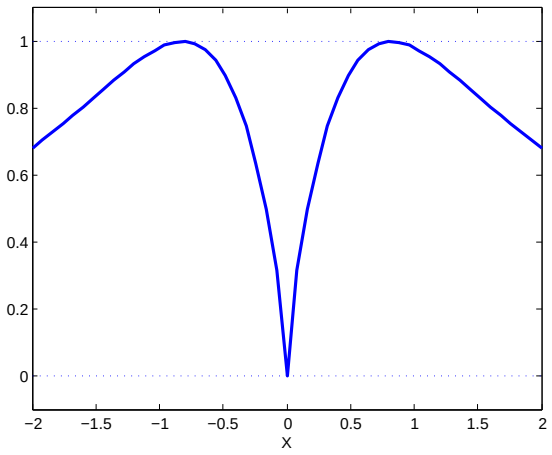


Figure 2.5: Shape quality index for a tetrahedron

only $z = \sqrt{2/3}$ fulfills the *invertibility* of the element. The maximum at $z = -\sqrt{2/3}$ is not valid, since the element is tangled. All of the element configurations where the $z \leq 0.0$ are not valid, since volume of the element is negative, the determinant of the shape matrix, $\sigma = \det(\mathbf{S})$, is less than zero.

2.3.2 Quadrilateral and hexahedral elements

For quadrilateral and hexahedral elements, Knupp (2003a) proposed that the quality index of a given element is a function of the sub-elements that could be obtained inside of the original shape. Quadrilateral elements are sub-divided into four right triangular elements, and hexahedrons, into eight right tetrahedrons.

Figure 2.6 shows how a quadrilateral element is divided into four triangles. Also, shows the numbering for creating the sub-shapes from the original element as proposed by Knupp (2003a,b). Figure 2.7 shows the splitting of an hexahedron into eight tetrahedral elements.

According to the requirements listed in Section 2.1, the quality of an element should not depend on the numbering of the nodes. The numbering of the elements does not affect the quality of the given element, but it is important to keep track of the

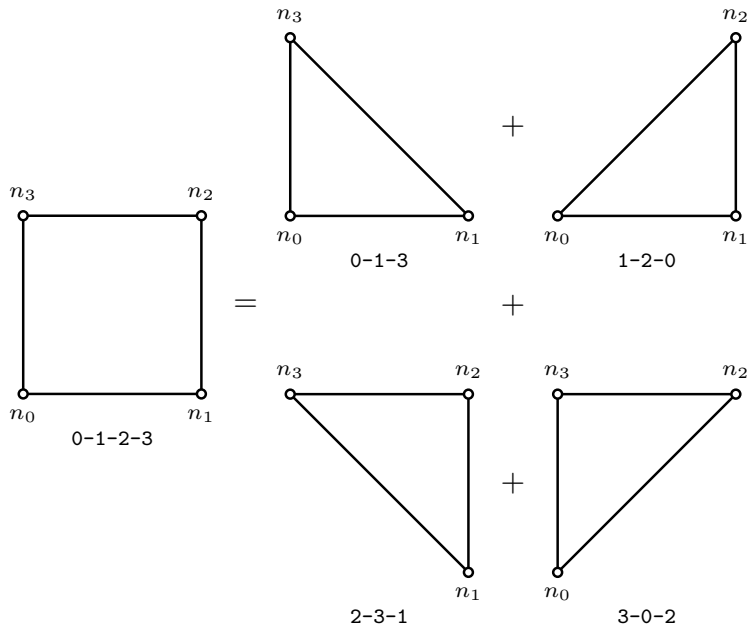


Figure 2.6: Division of a quadrilateral into triangular elements

numbering or order of the sub-elements to properly generate the quality index of quadrilateral or hexahedral elements. Tables 2.1 and 2.2 show the correct numbering of the sub-elements for quadrilateral and hexahedral elements (Knupp, 2003a,b).

Table 2.1: Nodal ordering for the sub-shapes of a quadrilateral element

	Node 1	Node 2	Node 3
Element 1	0	1	3
Element 2	1	2	0
Element 3	2	3	1
Element 4	3	0	2

To compute the quality index of the sub-elements, the equations presented in Section 2.3.1 are still valid. However, the *ideal* element needs to be modified to match the divisions performed to the hexahedral and quadrilateral elements.

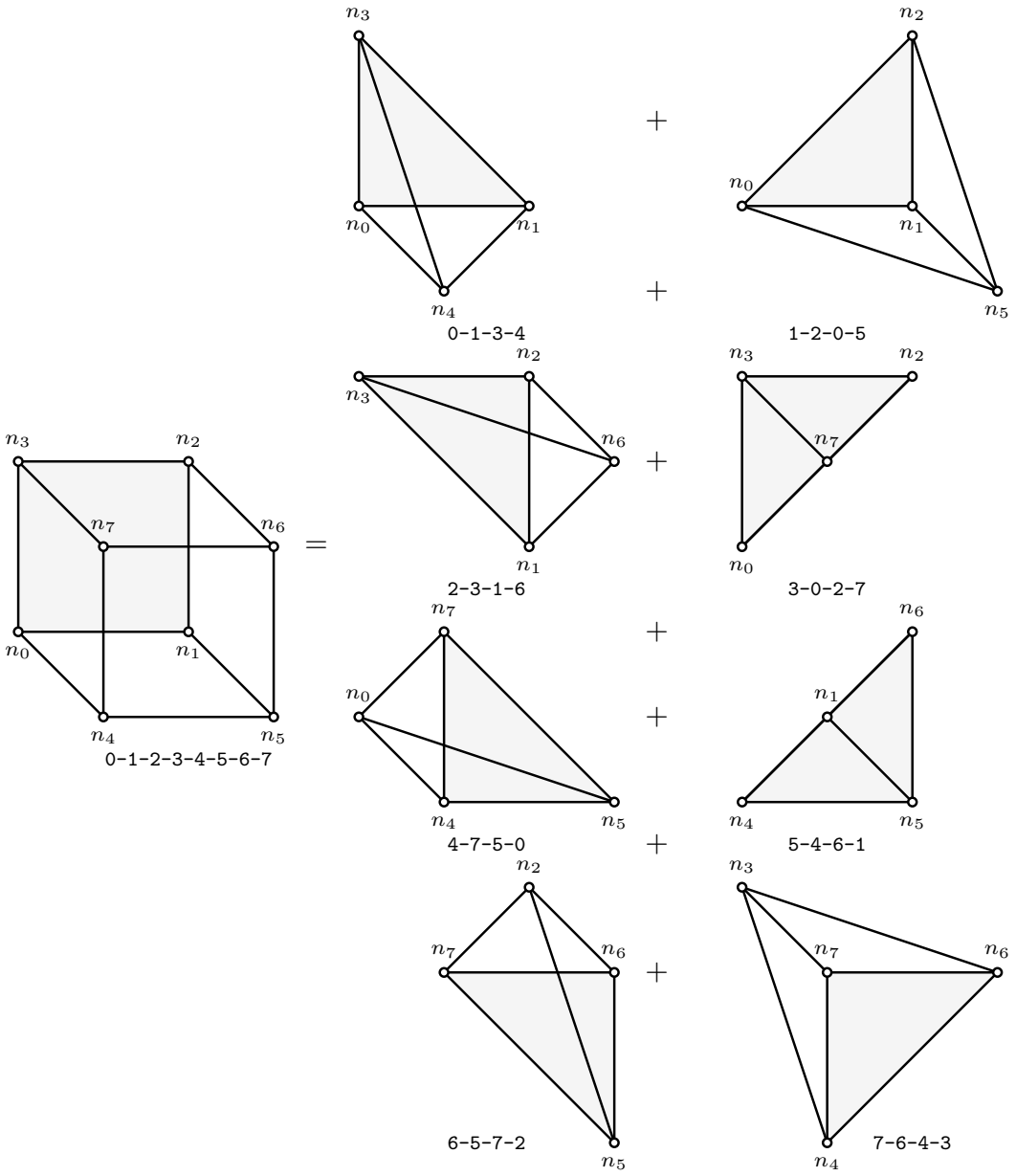


Figure 2.7: Division of a hexahedron into tetrahedral sub-elements

Table 2.2: Nodal ordering for the sub-shapes of a hexahedral element

	Node 1	Node 2	Node 3	Node 4
Element 1	0	1	3	4
Element 2	1	2	0	5
Element 3	2	3	1	6
Element 4	3	0	2	7
Element 5	4	7	5	0
Element 6	5	4	6	1
Element 7	6	5	7	2
Element 8	7	6	4	3

A right triangle or tetrahedron is used as the *ideal* elements to compute the quality index for the sub-elements, because they could represent the corners of the quadrilateral or hexahedral elements. Therefore, for each sub-element of a hexahedron (see Figure 2.7), the mapping between reference and ideal elements (Equation (2.2)) is given by:

$$\boldsymbol{W} = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{2.10}$$

and for each sub-element of a quadrilateral element (see Figure 2.6), by:

$$\boldsymbol{W} = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix} \tag{2.11}$$

since right tetrahedrons are used as reference are ideal elements[2]. For each tetrahedrons, $\mathbf{A}$ is defined according Equation (2.5).

The *physical*, *reference* and *ideal* elements for a quadrilateral mesh are shown in Figure 2.8. The elements used are right triangles, due the splitting of the element into sub-shapes, see Figure 2.6 to review the division of quadrilateral elements. For hexa-

[2]because the optimal shape for the hexahedron is set to be a cube, if another shape is to be used as the *ideal* element, the values for $\boldsymbol{W}$ in Equation (2.10) will be different.

hedrons, the *reference* and *ideal* elements will be right tetrahedrons and the *physical* element will be formed as described by Table 2.2.

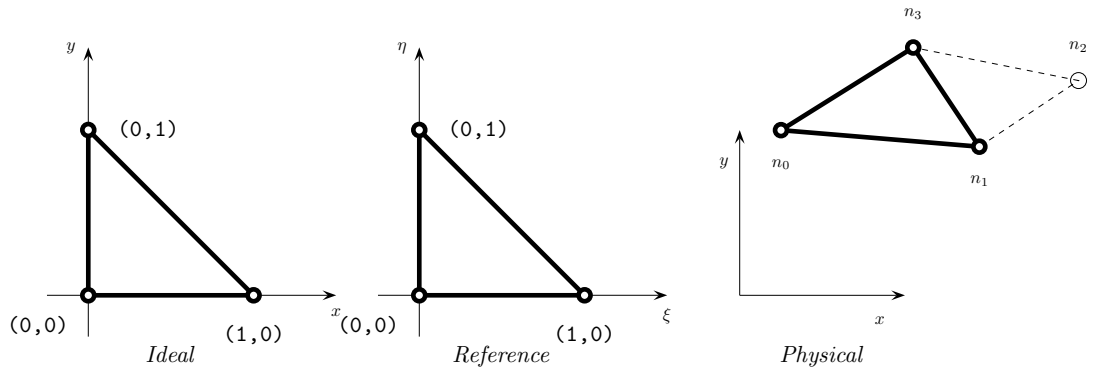


Figure 2.8: *Physical*, *reference* and *ideal* elements for quadrilateral element

For quadrilateral and hexahedral, Knupp (2003a) defines the shape quality index as

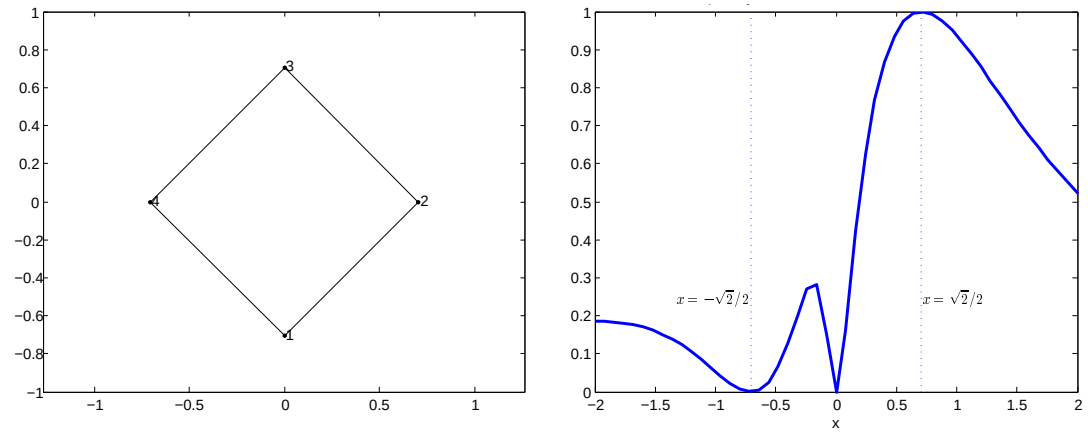
$$q = \frac{1}{m}\sum_{i=1}^{m} q_i^2 \tag{2.12}$$

where m is the number of sub-elements in the quadrilateral (4) or hexahedral element (8), and q_i is the quality index of the sub-shape inside the element. Expressed in terms of the shape metric of each sub-element (η_i), Equation (2.12) gives,

$$q = \frac{1}{m}\sum_{i=1}^{m} \left(\frac{1}{\eta_i}\right)^2 \tag{2.13}$$

The shape metric η_i for each of the sub-elements is computed using Equation (2.4).

Figure 2.9(a) depictures a mesh conformed with one quadrilateral element. This element is delimited by four nodes, being $\mathbf{x}_1 = (0, -\sqrt{2}/2)$, $\mathbf{x}_3 = (0, \sqrt{2}/2)$ and $\mathbf{x}_4 = (-\sqrt{2}/2, 0)$ fixed. Node $\mathbf{x}_2$ is allowed to move (the y coordinate is kept constant and equal to 0.0).



(a) (b)

Figure 2.9: Shape quality index for a quadrilateral element. (a) One element mesh. (b) Diagram of the quality index.

The shape quality index of the quadrilateral element is shown in Figure 2.9(b) as a function of $\mathbf{x}_2$. The maximum, $q = 1.0$, is located at $x = \sqrt{2}/2$, where a square element is obtained, as shown by Figure 2.9(a). The shape quality index has two minima, $q = 0$, located at $x = 0.0$ and $x = -\sqrt{2}/2$. When the angle between two element edges (attached to node $\mathbf{x}_2$) is equal to 180° one of the minima is obtained. The second minimum, located at $x = -\sqrt{2}/2$, occurs when the position of node $\mathbf{x}_2$ is the same as node $\mathbf{x}_4$, causing that the area of the element is zero. For values of $x \leq -\sqrt{2}/2$, the area of the element is negative; the element is inverted or tangled.

In general, the shape quality index (q) takes values from 0 (for an element that has *null* area or volume) to 1 (for an element that is a scaled version of the *ideal* element). Tangled elements (which have negative area or volume) will also have a non-zero and positive shape quality index, but the determinant of the shape matrix, $\sigma = \det(\mathbf{S})$, will be negative. This property will be used for untangling purposes.

Chapter 3

Minimization methods using the shape-based element quality for improving a mesh in a $\mathbb{R}^3$ domain

3.1 Objective function for mesh smoothing

In Section 2.3, expressions for the quality index for the mostly used elements were presented. If an element has low shape quality index, a modification to the element is required. There are two possibilities, change the connectivity of the elements (outside of the scope of the current work) or change the geometry of the element. The later means to changing the location of the nodes that are attached to the element. Therefore, the shape quality index of the surrounding will be affected. In this work, we will improve the shape quality index of the elements (we will increase the shape quality index of the elements by modifying their geometry).

Knupp (2003b) proposes an objective function that incorporates the shape quality index of all the elements in a given set of elements. For each of these elements, the shape quality index is defined by Equations (2.9) (for triangular and tetrahedral elements) and (2.13) (for quadrilateral and hexahedral elements). The objective function

is defined as

$$f = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{q_i} \right) - 1 \tag{3.1}$$

where N is the number of elements in the set elements and q_i is the shape quality index for the i element (see Equations (2.9) and (2.13)). Equation (3.1) has an offset of -1, so the objective function will have a minimum of 0.0 for elements matching the *ideal* element.

For low shape quality index elements, the objective function (Equation (3.1)) will take values much larger than zero ($f \gg 0.0$). For elements with good shape quality index, the values of the objective function will be close to zero ($f \approx 0.0$). Recall, the shape quality index (q) takes values from 0.0 (for worst quality elements) to 1.0 (best quality elements).

The objective function could be written in terms of the shape metric (η), see Equations (2.9) and (2.13). For triangular and tetrahedral elements as:

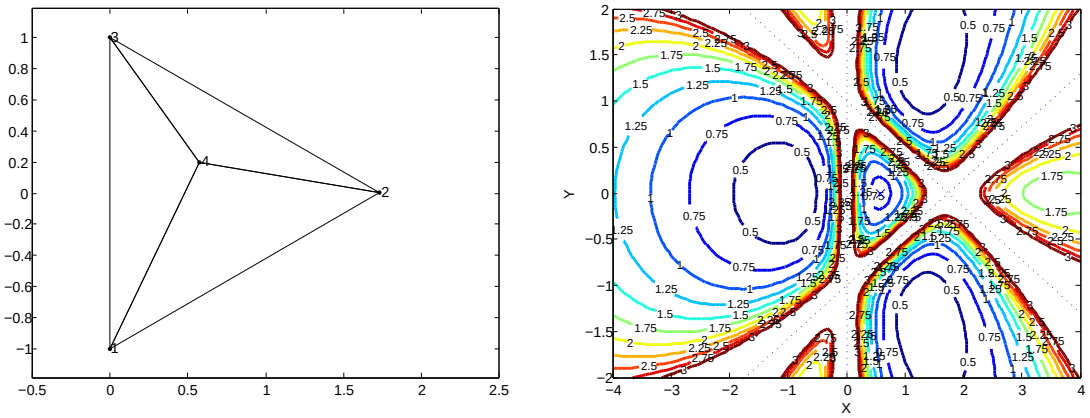
$$f = \frac{1}{N} \sum_{i=1}^{N} \eta_i - 1 \tag{3.2}$$

and, for quadrilateral and hexahedral elements as:

$$\begin{aligned} f &= \frac{1}{N} \sum_{i=1}^{N} \eta_i - 1 \\ &= \frac{1}{N} \sum_{i=1}^{N} \left[\frac{1}{m} \sum_{k=1}^{m} \left(\eta_{i,k} \right)^2 \right] - 1 \\ &= \frac{1}{mN} \sum_{i=1}^{N} \sum_{k=1}^{m} \left(\eta_{i,k} \right)^2 - 1 \end{aligned} \tag{3.3}$$

where N is the number of elements in the set elements, m is the number of sub-shapes (4 for quadrilateral elements and 8 for hexahedrons). $\eta_{i,k}$ is the shape metric of the kth node of the nth element.

Figure 3.1(a) depictures a mesh composed by four nodes, being $\mathbf{x}_1 = (0, -1)$, $\mathbf{x}_2 = (\sqrt{3}, 0)$ and $\mathbf{x}_3 = (0, 1)$ fixed. Figure 3.1(b) shows the objective function of a triangular mesh as a function of the coordinates (x, y) of node $\mathbf{x}_4$, which is the only node allowed to move. The objective function has several minima as shown in Figure 3.1(b), but only one of them is a valid minimum ($x = \sqrt{3}/3, y = 0.0$) since the quality index is minimized and also the determinant of the jacobian is positive (The optimal position is located inside the triangle delimited by the outer nodes ($\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$). All other minima are not valid since the mesh that is obtained is tangled.



(a) (b)

Figure 3.1: Objective function for a mesh with triangular elements. (a) Triangular Mesh. (b) Objective function contour plot.

Note that the objective function has several discontinuities (see Figure 3.1(b)). They appear when node $\mathbf{x}_4$ is located in the same location that nodes $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$; or when the moving node is places on the edges of the outer boundary (delimited by the outer nodes $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$).

The objective function of node $\mathbf{x}_4$ (see Figure 3.1(a)) is shown in Figure 3.2 as function only of the x coordinate of node $\mathbf{x}_4$ (keeping the y coordinate equal to 0.0). Asymptotes appear at $x = 0.0$ and $x = \sqrt{3}$, when the x coordinate of node $\mathbf{x}_4$ is equal to the x coordinate of the other nodes. Only for values of $0 < x < \sqrt{3}$, the *invertibility* of the elements is fulfilled. Out side of that range, the elements in the mesh are tangled or have a negative jacobian.

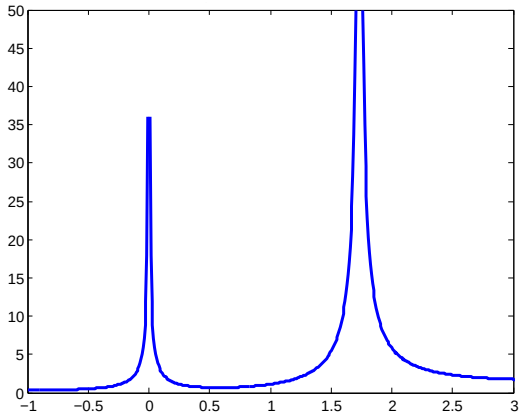
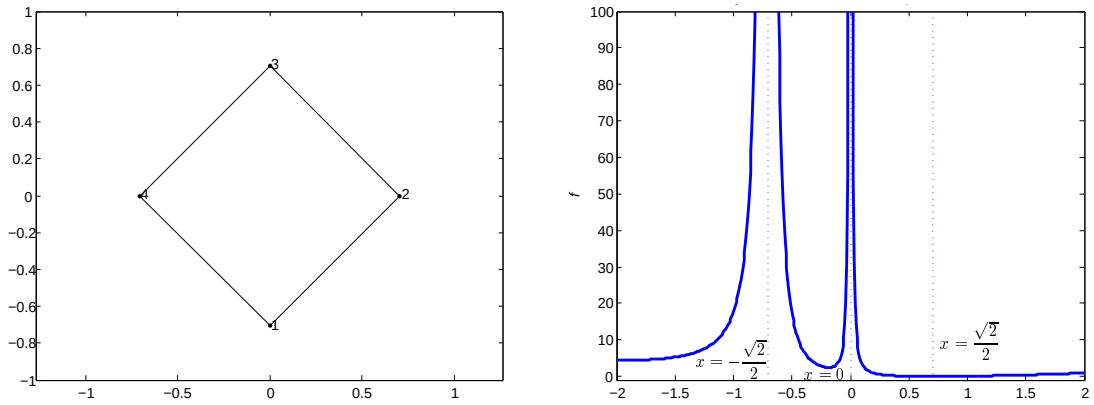


Figure 3.2: Objective function for a mesh with triangular elements

Figure 3.3(a) depicts a quadrilateral mesh composed by one single element. This element is delimited by four nodes, being $\mathbf{x}_1 = (0, -\sqrt{2}/2)$, $\mathbf{x}_3 = (0, \sqrt{2}/2)$ and $\mathbf{x}_4 = (-\sqrt{2}/2, 0)$ fixed. Figure 3.3(b) shows the objective function for a quadrilateral element as functions of node $\mathbf{x}_2$. The position of node $\mathbf{x}_2$ was modified by moving it from $x = 2.0$ to $x = -2.0$ (y coordinate is set to $y = 0.0$). The function has a minimum located at $x = \sqrt{2}/2$ (the quality index is equal to one) and two asymptotes located at $x = 0$ and $x = -\sqrt{2}/2$, where the quality of the element is equal to zero.

3.1.1 Modification to the objective function

As shown by Figures 3.1, 3.2 and 3.3, the objective function based on the shape index may contain asymptotes (when an element has *null* area or volume). Under this conditions, the minimization process may lead to a local minimum. Therefore, the global



(a) (b)

Figure 3.3: Objective function quadrilateral element. (a) Quadrilateral Element. (b) Objective function plot.

minimum may not be reached. Note that only the mesh configuration corresponding to the *global* minimum will provide an untangled mesh and the best quality for the mesh. On the contrary, mesh configuration corresponding to *local* minima, will lead to tangled elements.

Following the work of Escobar et al. (2003), a modification to the objective function (or to the quality index shown in Section 2.3) is necessary to eliminate the discontinuities and obtain only one minimum.

The discontinuities in the objective function are due to the fact that the determinant of the shape matrix is zero ($\sigma = 0$). To avoid the divisions by zero, Escobar et al. (2003) modify the denominator of Equation (2.4) with a function $h(\delta)$. The shape metric (η) will be a function that depends on the shape matrix ($\mathbf{S}$) and a function $h(\sigma)$. Equation (2.4) turns into:

$$\eta_{\text{mod}} = \frac{|\mathbf{S}|^2}{nh(\sigma)^{2/n}} \tag{3.4}$$

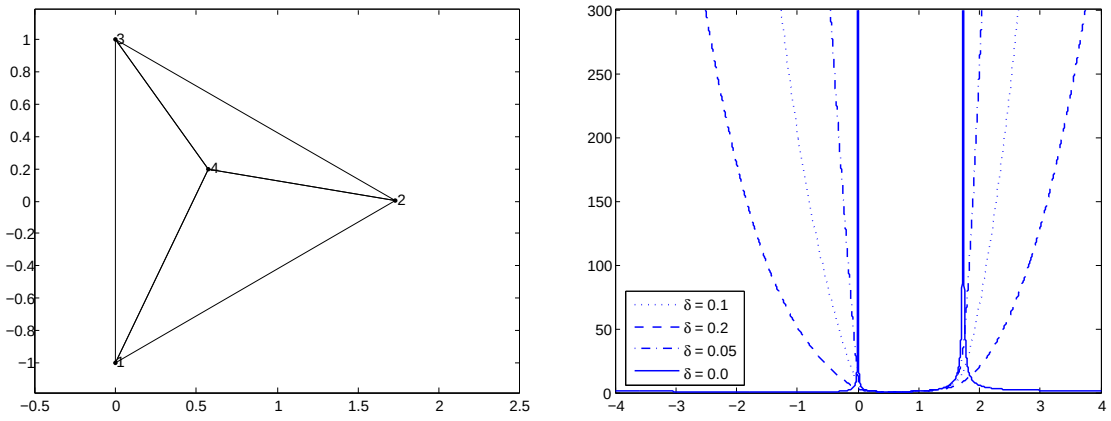
where

$$h(\sigma) = \frac{1}{2}\left(\sigma + \sqrt{\sigma^2 + 4\delta^2}\right) \tag{3.5}$$

being δ a parameter. See Escobar et al. (2003) for details on the selection of parameter δ.

If an element has *null* area or volume ($\sigma = 0$), the modified shape metric (η_{mod}) will not be indeterminated (Equation (3.4)), due to the δ parameter. Equations (3.4) and (3.5) provide a smoothed and convex objective function with no discontinuities that could be minimized with an optimization method.

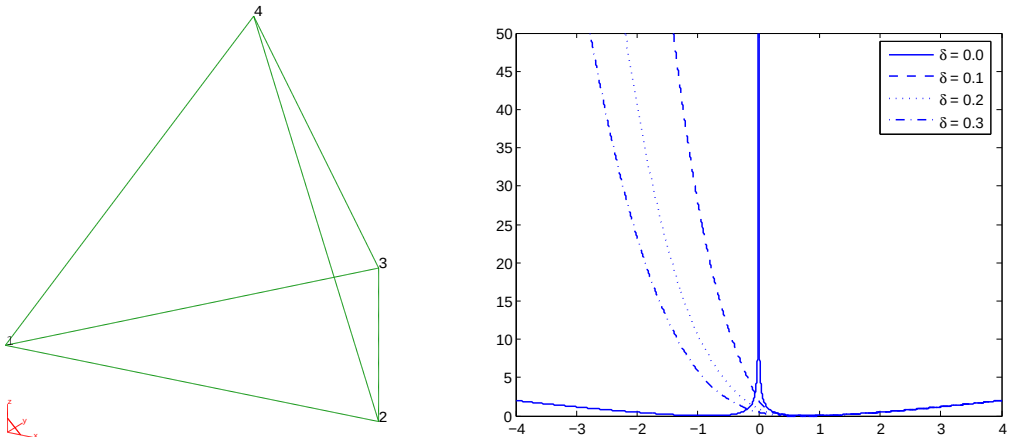
Figure 3.4(a) depictures a mesh composed by three triangular elements and four nodes, being $\mathbf{x}_1 = (0, -1)$, $\mathbf{x}_2 = (\sqrt{3}, 0)$ and $\mathbf{x}_3 = (0, 1)$ fixed. Figure 3.4(b) shows the original ($\delta = 0$) and the modified ($\delta = 0.1, 0.2, 0.3$) objective functions in terms of the x coordinate of node $\mathbf{x}_4$, while keeping the y coordinate fixed and equal to zero. Note that the modified objective functions are smoothed with no asymptotes.



(a) (b)

Figure 3.4: Modified objective function for triangular mesh. (a) Triangular elements. (b) Modified objective function in terms of position of node $\mathbf{x}_4$

Figure 3.5(a) shows a mesh with one tetrahedron, being $\mathbf{x}_1 = (0, 0, 0)$, $\mathbf{x}_2 = (1, 0, 0)$ and $\mathbf{x}_3 = (1/2, \sqrt{3/4}, 0)$ fixed. Node $\mathbf{x}_4$ is allowed to move over the z axis. Fig-



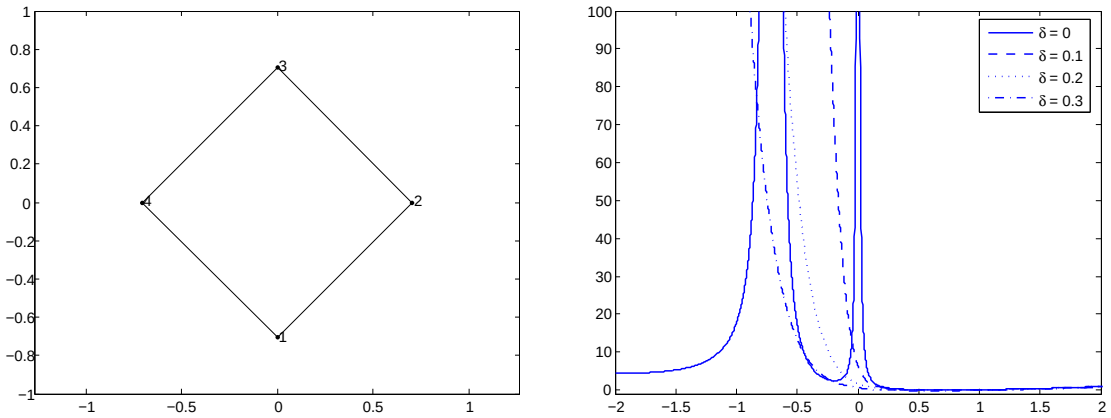
(a) (b)

Figure 3.5: Modified objective function for tetrahedral mesh. (a) Tetrahedral element. (b) Modified objective function in terms of position of node $\mathbf{x}_4$

ure 3.5(b) shows the original ($\delta = 0$) and the modified ($\delta = 0.1, 0.2, 0.3$) objective functions in terms of the z coordinate of node $\mathbf{x}_4$. Note that the modified objective functions are smoothed with no asymptotes.

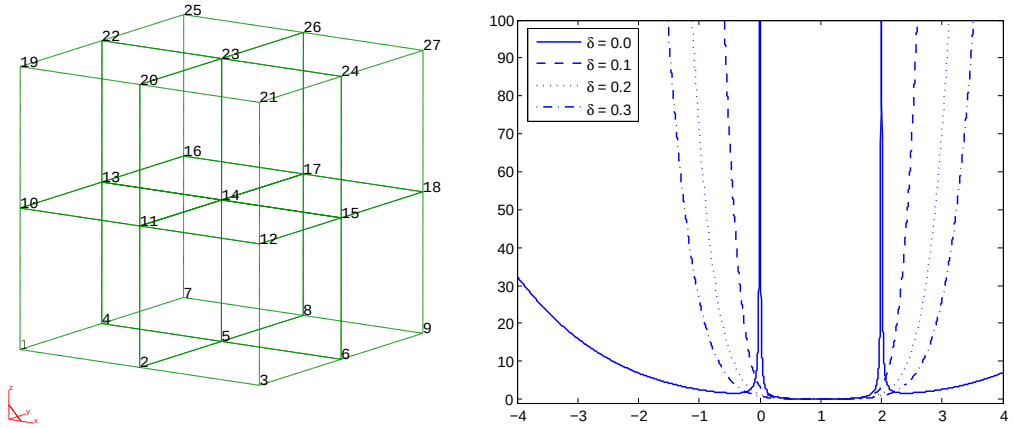
Figure 3.6(a) depictures a mesh with one single quadrilateral element. The element is conformed by four nodes, being $\mathbf{x}_1 = (0, -\sqrt{2}/2)$, $\mathbf{x}_3 = (0, \sqrt{2}/2)$ and $\mathbf{x}_4 = (-\sqrt{2}/2, 0)$ fixed. Figure 3.6(b) shows the original ($\delta = 0$) and the modified ($\delta = 0.1, 0.2, 0.3$) objective functions in terms of the x coordinate of node $\mathbf{x}_2$. Note that the asymptotes are removed from the modified objective function.

Figure 3.7(a) depictures a hexahedral mesh with eight elements and twenty-seven nodes. The mesh represent a cube with length equal to 2.0. All of nodes are fixed but node 14. The original ($\delta = 0$) and the modified ($\delta = 0.1, 0.2, 0.3$) objective functions in terms of the x coordinate of node 15 is shown in Figure 3.7(b). The original objective function contains two asymptotes when $\mathbf{x}_{14} = \mathbf{x}_5$ and $\mathbf{x}_{14} = \mathbf{x}_{23}$. These asymptotes do not appear in the modified objective functions.



(a) (b)

Figure 3.6: Modified objective function for quadrilateral mesh. (a) Quadrilateral element. (b) Modified objective function in terms of position of node $\mathbf{x}_2$



(a) (b)

Figure 3.7: Modified objective function for hexahedral mesh. (a) Hexahedral elements. (b) Modified objective function in terms of position of node $\mathbf{x}_1 4$

3.1.2 Derivatives of the objective function

The first and second derivatives of the objective function with respect to a coordinate component of a node are required for the minimization process. The *Newton-Raphson* method requires the calculation of the gradient vector and hessian matrix of the objective function. Other methods, like *BFGS*, just require the calculation of the gradient

vector of the objective function.

The derivatives of the objective function, Equation (3.1), are:

- For triangular and tetrahedral elements:

$$\frac{\partial f(\eta)}{\partial \alpha_i} = \frac{1}{N} \sum_{n=1}^{N} \frac{\partial \eta}{\partial \alpha_i} \tag{3.6}$$

$$\frac{\partial^2 f(\eta)}{\partial \alpha_i \partial \alpha_j} = \frac{1}{N} \sum_{n=1}^{N} \frac{\partial^2 \eta}{\partial \alpha_i \partial \alpha_j} \tag{3.7}$$

- For quadrilateral and hexahedral elements

$$\frac{\partial f(\eta)}{\partial \alpha_i} = \frac{2}{mN} \sum_{n=1}^{N} \sum_{k=1}^{m} \left(\eta \frac{\partial \eta}{\partial \alpha_i} \right) \tag{3.8}$$

$$\frac{\partial^2 f(\eta)}{\partial \alpha_i \partial \alpha_j} = \frac{2}{mN} \sum_{n=1}^{N} \sum_{k=1}^{m} \left(\frac{\partial \eta}{\partial \alpha_j} \frac{\partial \eta}{\partial \alpha_i} + \eta \frac{\partial^2 \eta}{\partial \alpha_i \partial \alpha_j} \right) \tag{3.9}$$

where η is the element shape metric, α_i is a coordinate component, N is the number of elements attached to the moving node and m is the number of sub-elements for quadrilaterals ($m = 4$) or Hexahedral ($m = 8$). See Appendix A for detailed steps to obtain Equations (3.6), (3.7), (3.8) and (3.9).

Figure 3.8 depictures a mesh conformed by four quadrilateral elements with nodes located at $\mathbf{x}_1 = (0, 0)$, $\mathbf{x}_2 = (1, 0)$, $\mathbf{x}_3 = (2, 0)$, $\mathbf{x}_4 = (0, 1)$, $\mathbf{x}_5 = (0.375, 1.25)$, $\mathbf{x}_6 = (2, 1)$, $\mathbf{x}_7 = (0, 2)$, $\mathbf{x}_8 = (1, 2)$ and $\mathbf{x}_9 = (2, 2)$. For the mesh configuration presented in Figure 3.8, the objective function, its gradient vector and its hessian matrix, follows

$$
\begin{aligned}
f(\mathbf{x}_5) &= 1.85 \\
\boldsymbol{\nabla} f(\mathbf{x}_5) &= \begin{bmatrix} -25.2415 \\ 19.6892 \end{bmatrix} \\
\mathbf{H}(\mathbf{x}_5) &= \begin{bmatrix} 576.258 & -494.178 \\ -494.178 & 454.939 \end{bmatrix}
\end{aligned}
$$

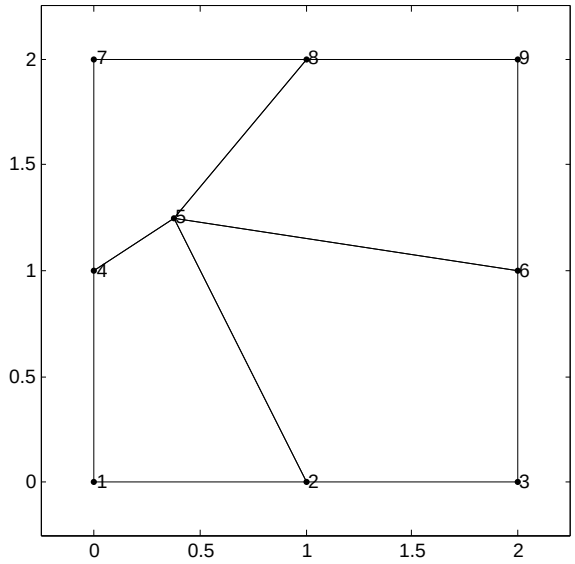


Figure 3.8: 4-quadrilateral element mesh. Node 5 is located at $(0.375, 1.25)$.

For the current configuration, $\frac{\partial f}{\partial x}$ is negative, meaning that node $\mathbf{x}_5$ will move in the direction of positive x. $\frac{\partial f}{\partial y}$ is positive, meaning that node $\mathbf{x}_5$ will move in the direction of negative y.

3.2 Minimization methods

A minimization method seeks to solve the problem of finding a set of values for the arguments of a function, such that, the function takes the lowest value possible.

An optimization problem can be expressed as:

> *Given*: a function $f : A \to \mathbb{R}$ from some set A to the real numbers
>
> *Sought*: an element x_0 in A such that $f(x_0) \leq f(x)$ for all x in A.

The optimization methods seeks for the roots in the gradient of functions. When a root in the gradient is found, a minimum or a maximum is obtained. The hessian defines if a minimum or a maximum is obtained. If the determinant of the hessian is positive, a minimum is obtained. If the determinant of the hessian is negative, a maximum.

There is a wide range of minimization methods, such as *Line-Search*, *Newton-Raphson*, *BFGS* among others (Venkataraman, 2001).

3.2.1 Line-Search

Line-Search is a minimization method that is based on following a search descent direction ($\mathbf{s}^k$) to obtain the minimum of a function. The descent direction is such that:

$$\left(\boldsymbol{\nabla} f\left(\mathbf{x}^k\right)\right)^{\mathrm{T}} \mathbf{s}^k < 0 \tag{3.10}$$

and can be expressed as

$$\min_{t \geq 0} \theta(t), \quad \text{where} \quad \theta(t) = f(\boldsymbol{x}^k + t\boldsymbol{s}^k) \tag{3.11}$$

Actually, the *Line-Search* is widely use inside other minimization methods to improve their convergence and reduce the number of iterations needed to obtain a minimum (Arroyo, 2009).

3.2.2 Newton-Raphson

The *Newton-Raphson* method is widely known for finding roots, I can be used for optimization purposes. At a minimum or maximum, the gradient of the function is zero ($\boldsymbol{\nabla} f = \mathbf{0}$), the *Newton-Raphson* could be used to obtain the location of the minima or maxima.

At each iteration, the *Newton-Raphson* can be expressed as

$$\mathbf{H}(\mathbf{x}_n)\delta\mathbf{x} = -\boldsymbol{\nabla} f(\mathbf{x}_n) \qquad \text{where} \qquad \delta\mathbf{x} = \mathbf{x}_{n+1} - \mathbf{x}_n \tag{3.12}$$

The disadvantages of this method are the computational cost of obtaining the hessian ($\mathbf{H}(\mathbf{x}_n)$) of the function to optimize and the requirement of the continuity of the hessian (the determinant of the hessian can only take non-zero values, $\det(\mathbf{H}) \neq 0$). The method has quadratic convergence rate.

3.2.3 Broyden–Fletcher–Goldfarb–Shanno (BFGS)

One of the Quasi-Newton minimization methods is the *Broyden–Fletcher–Goldfarb–Shanno* or simply *BFGS* method. To to reduce the computational cost of the hessian, the Quasi-Newton methods avoids the calculation of the hessian and perform an approximation to it but may require additional iterations as a trade-off. The Hessian is not computed but it is approximated by analyzing successive gradient vectors. The *BFGS* method has quadratic convergence and it is one of the most popular minimization methods in solving non-linear equations (Venkataraman, 2001).

For each iteration, the approximation hessian ($\mathbf{H}$) is updated as:

$$\mathbf{H}_{i+1} = \mathbf{H}_i + \mathbf{B}_i + \mathbf{C}_i \tag{3.13}$$

where

$$\mathbf{B}_i = \frac{\Big(\boldsymbol{\nabla} f_{i+1} - \boldsymbol{\nabla} f_i\Big)\Big(\boldsymbol{\nabla} f_{i+1} - \boldsymbol{\nabla} f_i\Big)^T}{\Big(\boldsymbol{\nabla} f_{i+1} - \boldsymbol{\nabla} f_i\Big)^T \delta\mathbf{x}} \tag{3.14}$$

$$\mathbf{C}_i = \frac{\nabla\mathbf{f}_i \nabla\mathbf{f}_i^T}{\nabla\mathbf{f}_i^T \delta\mathbf{x}} \tag{3.15}$$

$$\delta\mathbf{x} = \mathbf{x}_{n+1} - \mathbf{x}_n \tag{3.16}$$

being i is the iteration number and $\mathbf{H}_0$ equal to the identity matrix $\mathbf{I}$. (see Venkataraman, 2001)

3.3 Structure of the smoother

This section details the structure of the smoother, including its requirements and algorithm. In addition, a comparison with other methods to smooth meshes is presented.

3.3.1 Requirements

The list of requirements for the proposed smoother:

- The smoother must successfully improve the quality of triangular, tetrahedral, quadrilateral and hexahedral meshes.
- The smoother have to be able to smooth tangled meshes.
- Flexible choices in the minimization method to use (*Newton-Raphson*, *BFGS*, *Line-Search*).
- Written in `C++` in order to be incorporated into the `EZ4U` environment.
- The smoother will not try to optimize the position of the nodes using one *global* optimization. The smoother will perform multiple *local* optimizations (see Section 3.3.2 for details).

- The smoother will reuse existing software. For instance `OpenCascade`, the `VERDICT` libraries and the `EZ4U` environment.

3.3.2 Smoothing patches instead of global meshes

The computational cost of performing *global* mesh optimization may became too expensive. Therefore, in this work, we propose a local approach. In particular, we will iterate over the inner nodes. Following Knupp (2003b) and Escobar et al. (2003), for each inner node, a local patch will be considered to optimize the position of the considered node. Figure 3.9 shows a representation of how a mesh is divided into several quadrilateral patches.

3.3.3 The proposed smoother algorithm

Appendix 3.1 summarizes the proposed smoothing algorithm. The first step in the algorithm is to define the stop criterion of the smoother, which is defined by two parameters: a maximum number of iterations and a stopping distance. After one *global* iteration, the maximum movement of the nodes is compared with a tolerance defined as a fraction of the minimum edge length in the mesh. If the maximum movement is less than tolerance, the iteration process is stopped.

Algorithm 3.1 Mesh or *global* Smoothing algorithm

```
1: define iteration stop criteria (error and number of iterations)
2: while not mesh good enough do
3:    for all internal nodes do
4:       create sub-mesh
5:       if sub-mesh needs to be smoothed then
6:          carry out smoothing process
7:       end if
8:    end for
9: end while
```

For each of the internal nodes, the smoother considers the patch of elements that contains the node. Then, it checks the quality index of the elements in the patch (see

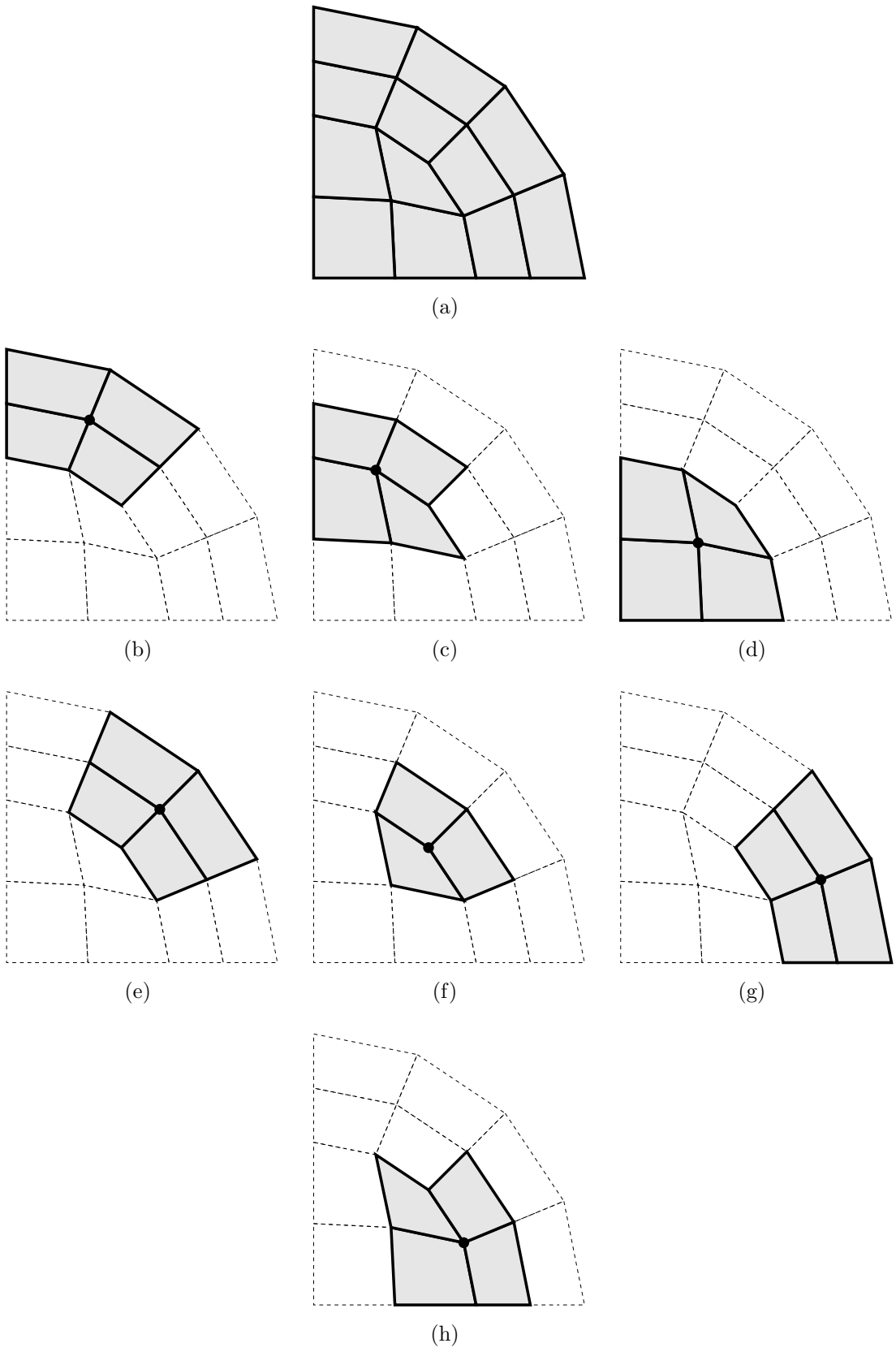


Figure 3.9: Division of a mesh into patches.

Appendix 3.2). Only if the quality index of at least one element attached to the node is below a predefined tolerance, the smoother will try to move the node.

Algorithm 3.2 Algorithm to check if a sub-mesh needs smoothing

Require: a node

```
for all elements attached to node do
   if the quality is lower than a tolerance then
      return true
   end if
end for
return false
```

Appendix 3.3 details the inner part of the optimization process. Similarly to the *global* smoothing (Appendix 3.1), the *local* smoothing has two parameters that control the optimization. These parameters are the maximum number of local iterations allowed and a stopping distance. The latter is defined as a fraction of the minimum edge length attached to the current node.

This algorithm requires two additional objects: a `Goal Function` and a `Minimizer`. The`Goal Function` contains the information about the objective function and its derivatives (gradient and hessian). The `Minimizer` defines the method used to improve the shape quality index of the patch (*Line-Search*, *BFGS*, *Newton-Raphson*) and iterates to obtain the location of the nodes for best shape quality index of the elements.

Algorithm 3.3 Node or *local* smoothing algorithm

Require: a node

```
define stop criteria (error and number of iterations)
init. Goal Function variable
init. Minimizer
perform minimization
if found minimum then
   update node location
end if
```

After one *global* iterations, all of the internal nodes have been relocated by the smoother.

Chapter 4

Results

This chapter contains the results of this work. First, we present several examples of the smoothing and untangling algorithm applied to quadrilateral and hexahedral meshes. Then, the computation cost of the minimization methods is analyzed. Finally, the robustness of the proposed method is illustrated by means of an example. In all the examples, the shape quality index for tangled elements is reported as 0.0, according to Knupp (2003a). Note that according to Equation (2.9), the shape quality index corresponding to tangled elements can be different than 0.0. For the examples with hexahedrons, some elements are hidden for clarity purposes.

4.1 Untangling meshes

4.1.1 Quadrilateral elements

Figure 4.1(a) depictures a mesh conformed by four quadrilateral elements, in a square domain with side length equal to 2.0. Node 5 location is set to be on top of node 8, $\mathbf{x}_5 = \mathbf{x}_8 = (1, 2)$, collapsing two of the elements. Figure 4.1(b) shows the untangled mesh, locating node 5 at $(1, 1)$. Figures 4.1(c) and 4.1(d) show the distribution of the elements according to the shape quality index before and after the smoothing process, respectively. Table 4.1 summarizes the statistical values for Figure 4.1. Note that using the proposed methods a mesh composed by square elements is obtained.

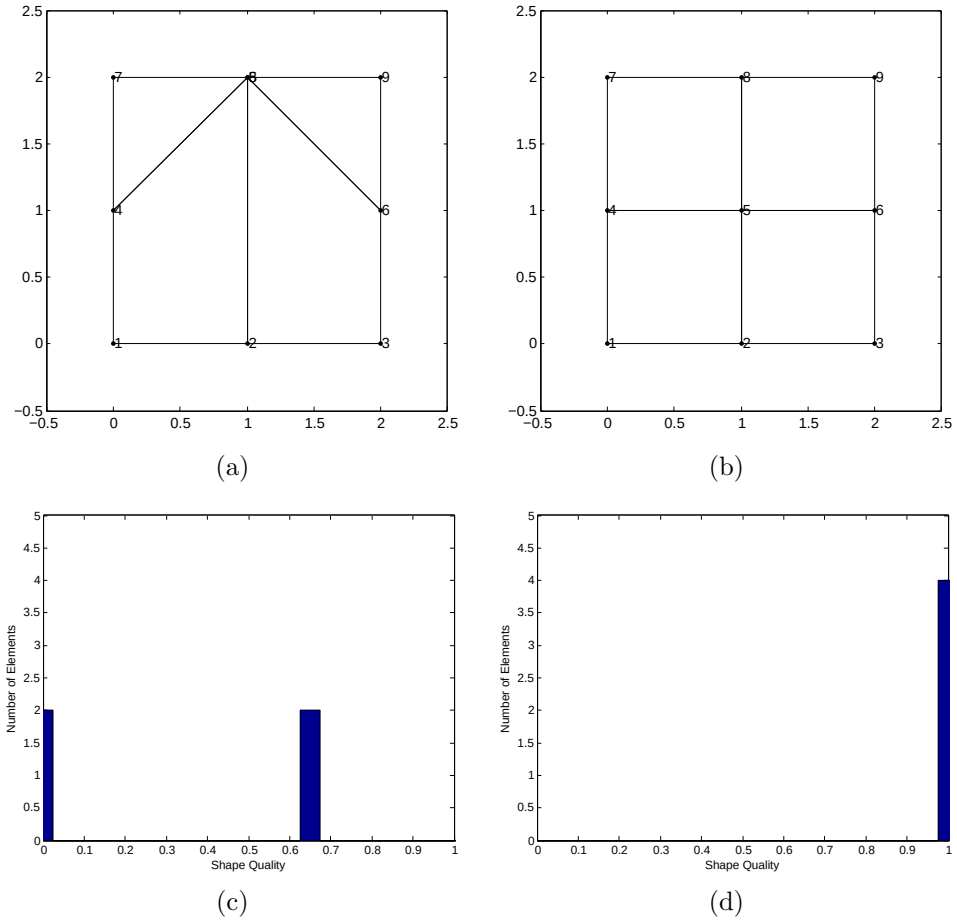


Figure 4.1: Example of smoothing a quadrilateral mesh. Mesh configuration before (a) and after (b) smoothing. Shape quality histogram before (c) and after (d) smoothing.

	Before	After
Minimum	0.000	1.000
Maximum	0.666	1.000
Mean	0.333	1.000
Std. dev.	0.333	0.000

Table 4.1: Shape-based quality statistical values for Figure 4.1

Figure 4.2(a) shows the same mesh configuration than Figure 4.1(a), but node 5 is located outside of the mesh domain, $\mathbf{x}_5 = (2.25, 1.75)$. The untangled mesh is shown in Figure 4.2(b) with $\mathbf{x}_5 = (1, 1)$. Figures 4.2(c) and 4.2(d) show the distribution of the elements according to the shape quality index before and after the smoothing

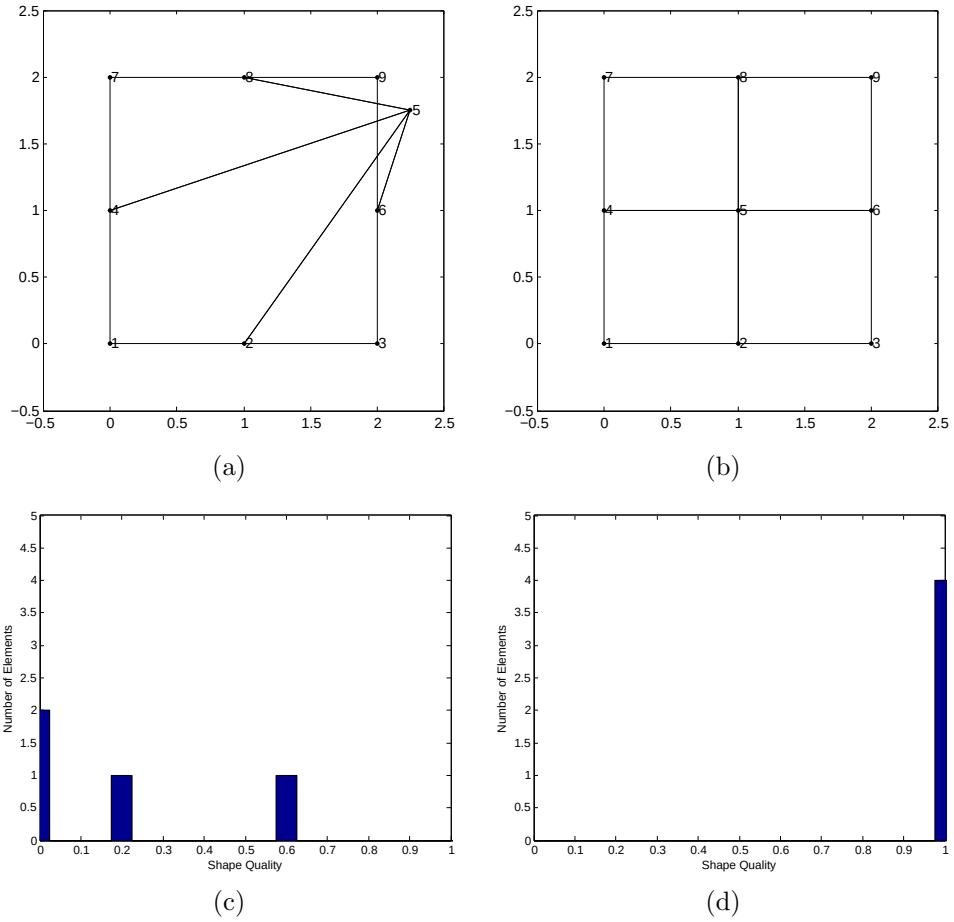


Figure 4.2: Example of smoothing a quadrilateral mesh. Mesh configuration before (a) and after (b) smoothing. Shape quality histogram before (c) and after (d) smoothing.

process, respectively. Table 4.2 summarizes the statistical values for Figure 4.2. Note that using the proposed methods a mesh composed by square elements is obtained.

	Before	After
Minimum	0.000	1.000
Maximum	0.585	1.000
Mean	0.193	1.000
Std. dev.	0.238	0.000

Table 4.2: Shape-based quality statistical values for Figure 4.2

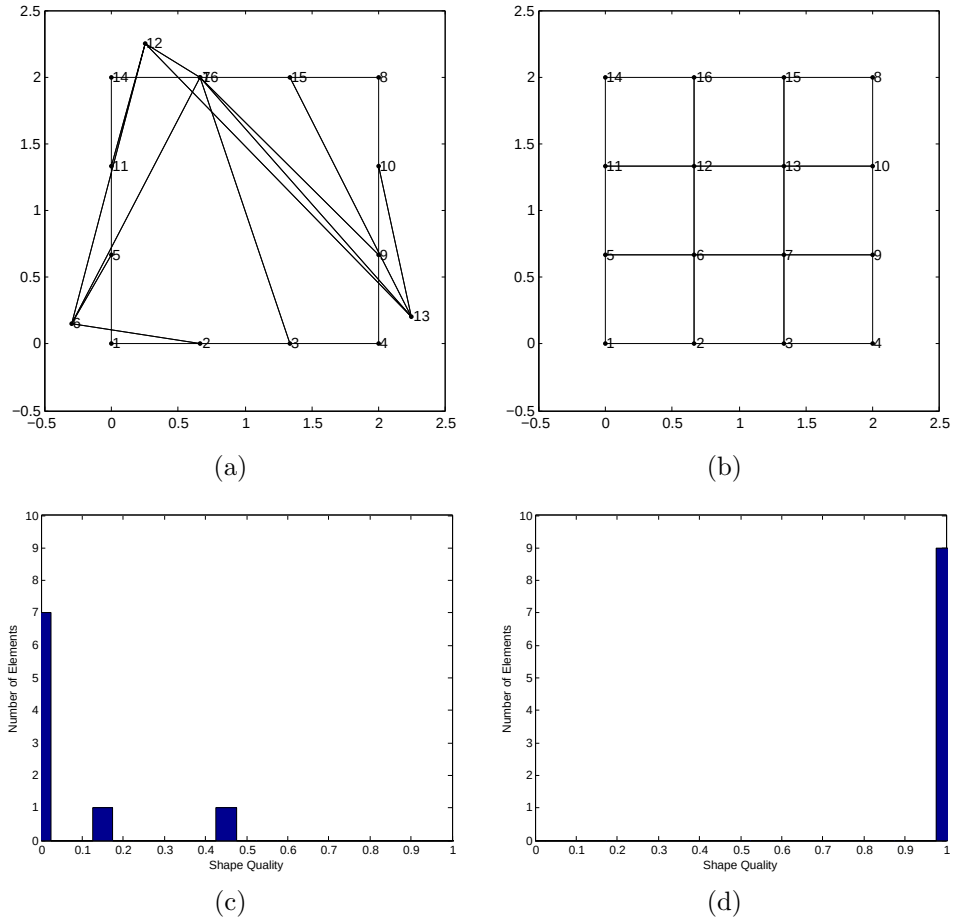


Figure 4.3: Example of smoothing a quadrilateral mesh. Mesh configuration before (a) and after (b) smoothing. Shape quality histogram before (c) and after (d) smoothing.

Figure 4.3(a) depictures a nine element mesh in a square domain of size 2. The internal nodes were randomized to tangle the mesh. The smoothed mesh is shown in Figure 4.3(b). Figures 4.3(c) and 4.3(d) show the distribution of the elements according to the shape quality index before and after the smoothing process, respectively. Table 4.3 summarizes the statistical values for Figure 4.3. As it is expected, a mesh composed by square elements is obtained.

	Before	After
Minimum	0.000	1.000
Maximum	0.444	1.000
Mean	0.065	1.000
Std. dev.	0.141	0.000

Table 4.3: Shape-based quality statistical values for Figure 4.3

Figure 4.4(a) shows one fourth of a ring with outer radius $r_{out} = 10$ and inner radius $r_{in} = 3$, meshed with quadrilateral elements. The interior elements were arranged in a square domain, decreasing the quality of the elements and in some cases, tangling them. The smoothed mesh is shown in Figure 4.4(b).

Figures 4.4(c) and 4.4(d) detail the distribution of the elements according to the shape quality index before and after the smoothing process, respectively. Table 4.4 summarizes the statistical values for Figure 4.2. Note that after smoothing, no element appears with shape quality index lower than 0.69.

Due to the lack of external node sliding, the smoother was not able to further improve the quality of the elements close to the inner radius, see Figure 4.4(b). Moreover, the smoother is still able to untangle and smooth the original mesh.

	Before	After
Minimum	0.000	0.648
Maximum	1.000	0.967
Mean	0.612	0.902
Std. dev.	0.445	0.079

Table 4.4: Shape-based quality statistical values for Figure 4.4

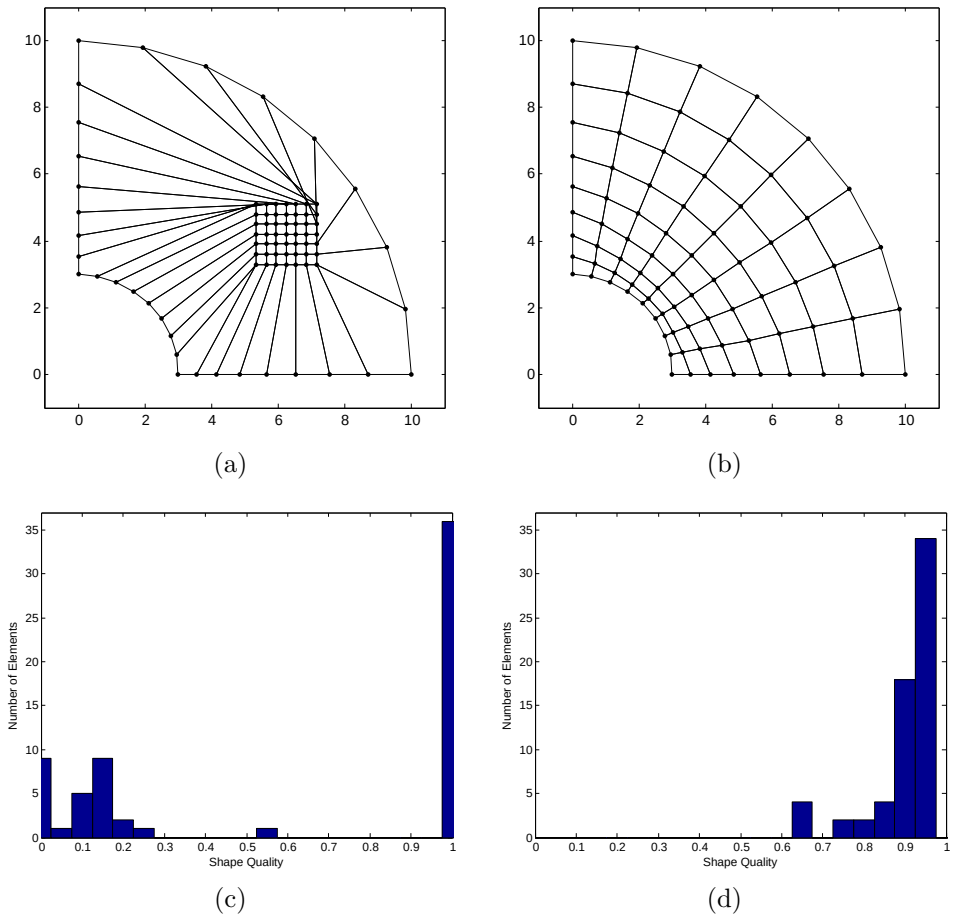


Figure 4.4: Example of smoothing a ring-shaped domain. Mesh configuration before (a) and after (b) smoothing. Shape quality histogram before (c) and after (d) smoothing.

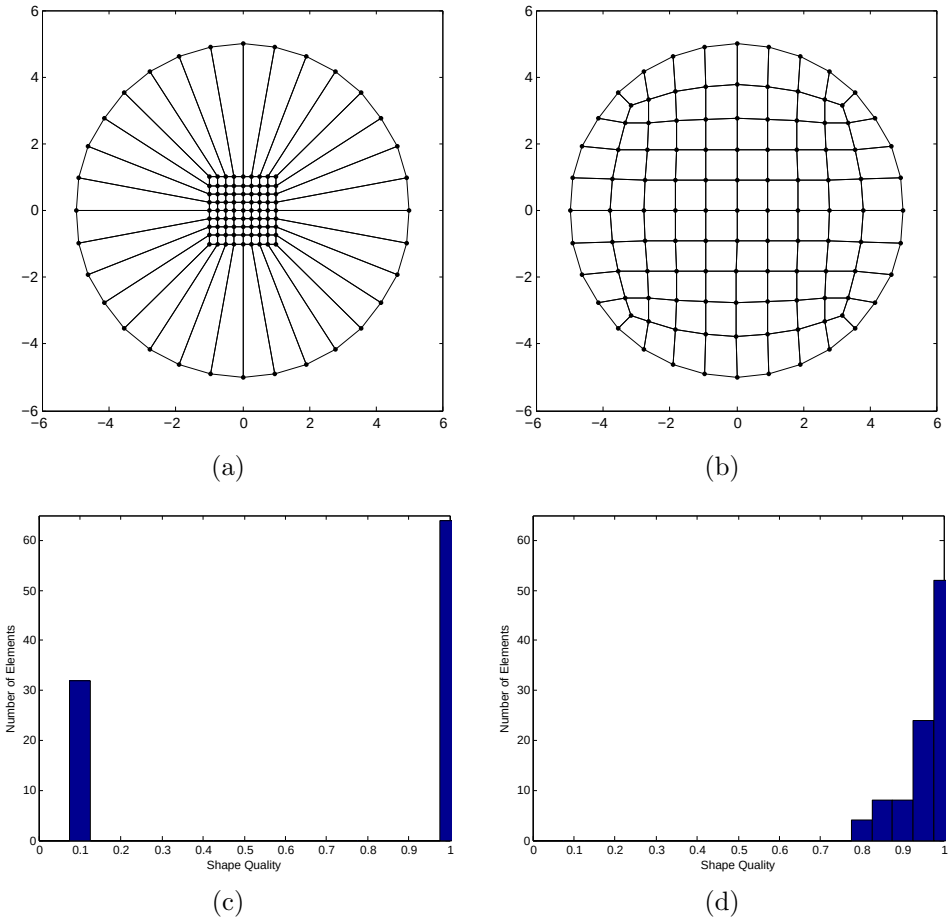


Figure 4.5: Quadrilateral mesh representing a circle with radius equals to 5 length units. Mesh configuration before (a) and after (b) smoothing. Shape quality histogram before (c) and after (d) smoothing.

Figure 4.5(a) depictures a quadrilateral mesh in a circular domain with radius $r = 5$. The interior domain was meshed as a square of edge size equal to 2 length units and 8 divisions per side. The smoothed mesh is shown in Figure 4.5(b). Figures 4.5(c) and 4.5(d) show the distribution of the elements according to the shape quality index before and after the smoothing process, respectively. Table 4.5 summarizes the statistical values for Figure 4.5. Note that in this case, the lowest value of the shape quality index after smoothing is 0.79.

	Before	After
Minimum	0.098	0.798
Maximum	1.000	1.000
Mean	0.704	0.957
Std. dev.	0.418	0.056

Table 4.5: Shape-based quality statistical values for Figure 4.5

4.1.2 Hexahedral elements

The example shown in Figure 4.6 is similar to the presented in Figure 4.1, but using hexahedra instead of quadrilateral elements. The mesh contains eight hexahedra and twenty-seven nodes in a square domain of size 2. The node located at the center of the domain was relocated to one of the boundary nodes, obtaining collapsed hexahedrons as shown in Figure 4.6(a). Four hexahedra have one edge with *null* length. Figure 4.6(b) shows the smoothed mesh. Figures 4.6(c) and 4.6(d) show the distribution of the elements according to the shape quality index before and after the smoothing process, respectively. Table 4.6 summarizes the statistical values for Figure 4.6. Note that eight cubes are obtained after smoothing the original mesh.

	Before	After
Minimum	0.000	1.000
Maximum	0.595	1.000
Mean	0.297	1.000
Std. dev.	0.297	0.000

Table 4.6: Shape-based quality statistical values for Figure 4.6

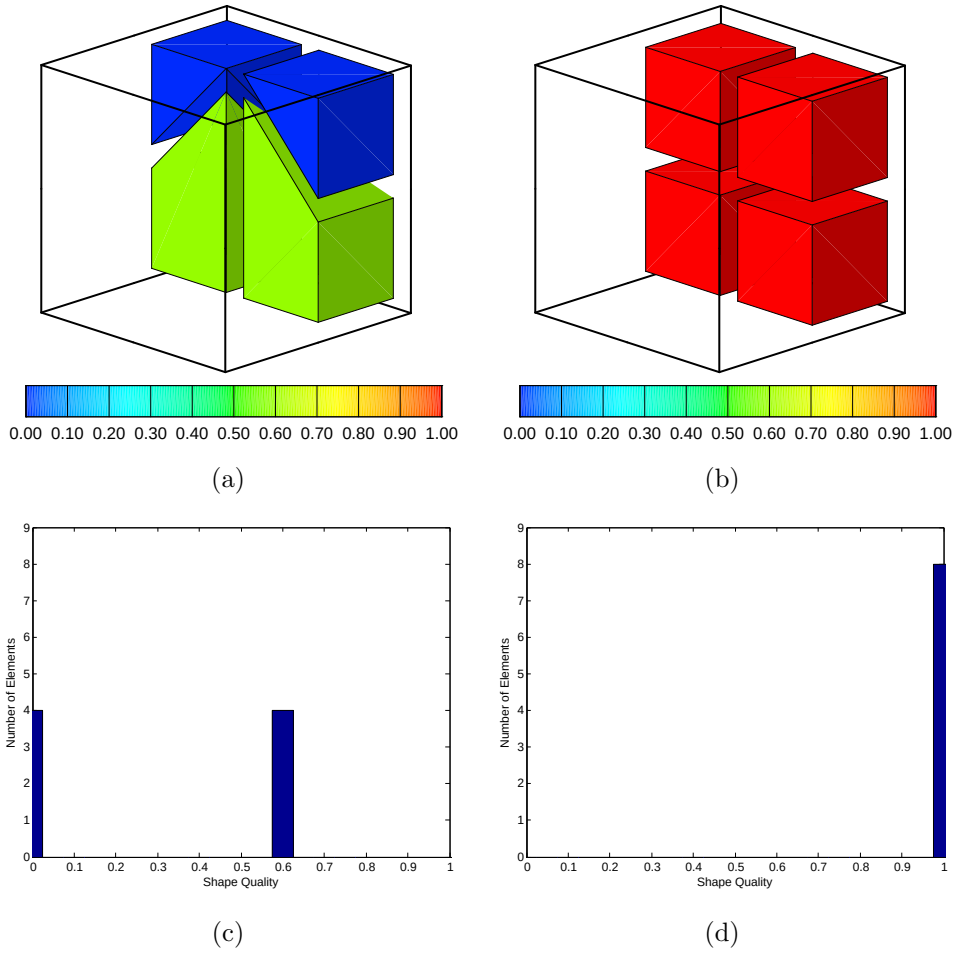


Figure 4.6: Example of smoothing a hexahedral mesh. Mesh configuration before (a) and after (b) smoothing. Shape quality histogram before (c) and after (d) smoothing.

Figure 4.7(a) depictures a hexahedral mesh, containing twenty-seven elements and sixty-four nodes, in a cubic domain of size 2. The location of the internal nodes is set in the boundary of the domain, obtaining collapsed elements. Figure 4.7(b) shows the smoothed mesh. Figures 4.7(c) and 4.7(d) show the distribution of the elements according to the shape quality index before and after the smoothing process, respectively. Table 4.7 summarizes the statistical values for Figure 4.7. As it is expected, twenty-seven cubes are obtained after smoothing the mesh.

Figure 4.8(a) shows a convex cylinder with radius $r = 5$ and height $z = 5$. Note that

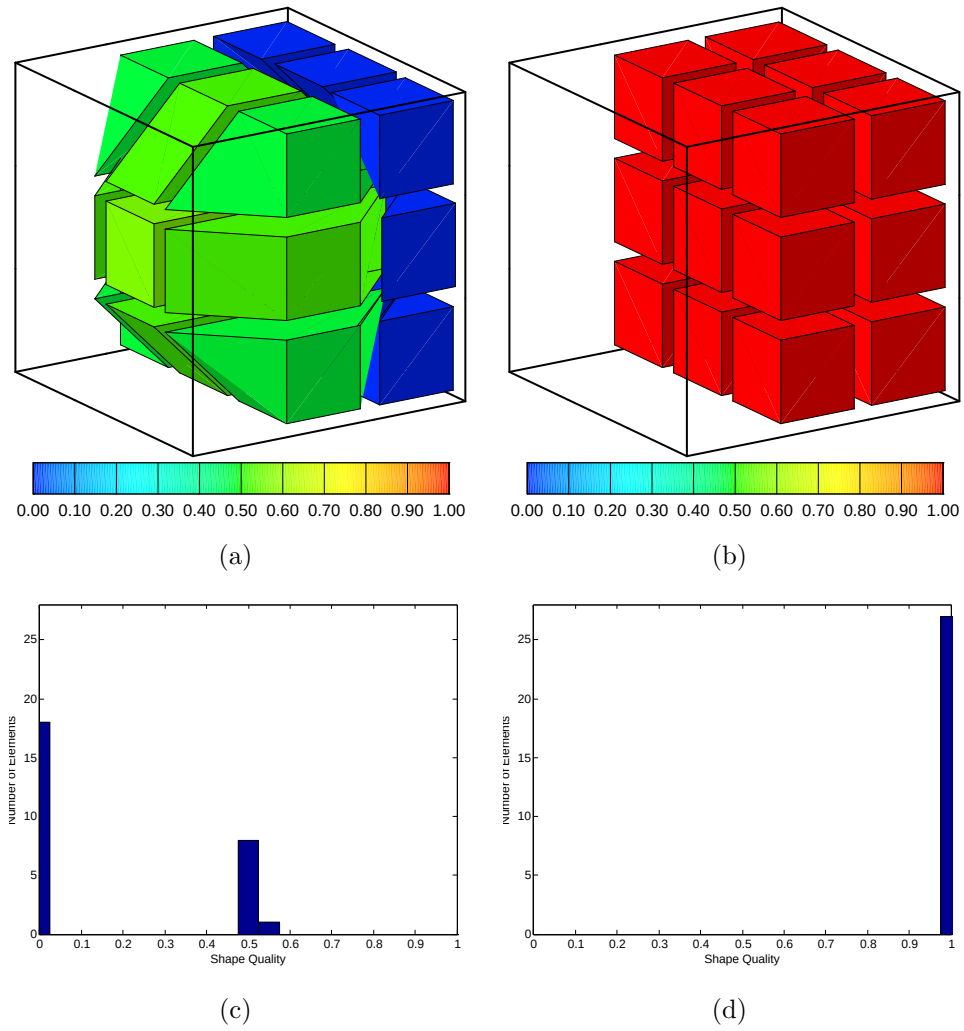


Figure 4.7: Example of smoothing a hexahedral mesh. Mesh configuration before (a) and after (b) smoothing. Shape quality histogram before (c) and after (d) smoothing.

	Before	After
Minimum	0.000	1.000
Maximum	0.567	1.000
Mean	0.169	1.000
Std. dev.	0.239	0.000

Table 4.7: Shape-based quality statistical values for Figure 4.7

the cap surfaces are non-planar. It is important to point out that several flat elements (with zero volume) are created, attached to each cap surface.

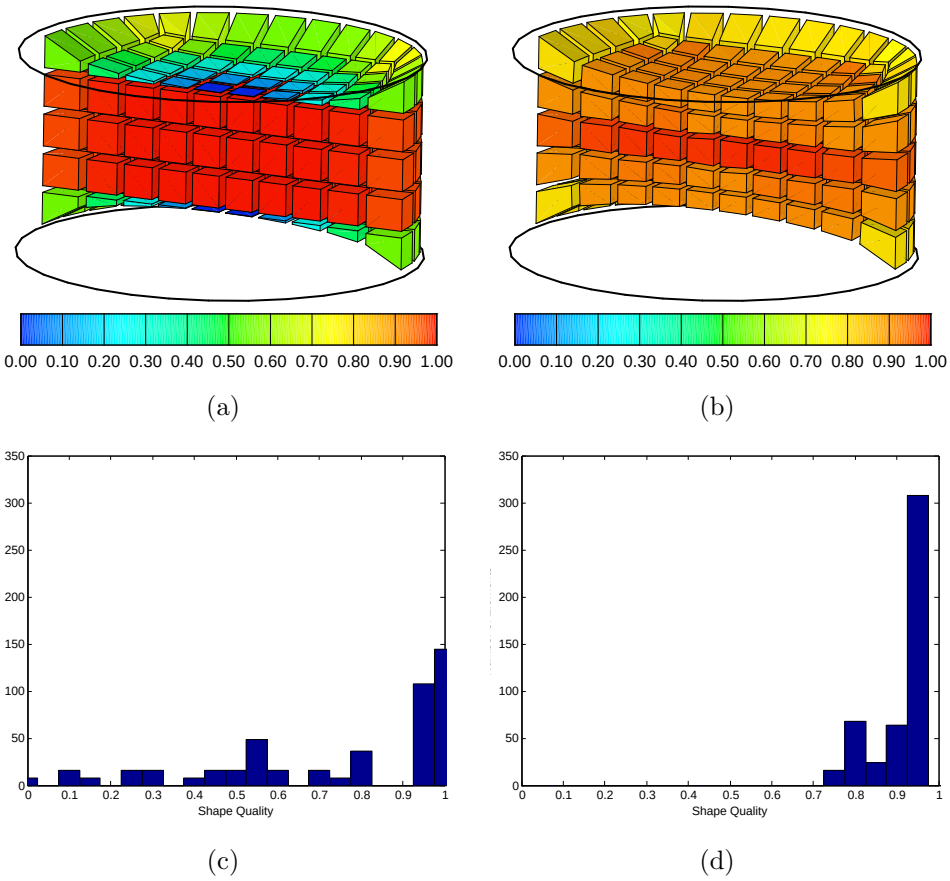


Figure 4.8: Example of smoothing a hexahedral mesh. Mesh configuration before (a) and after (b) smoothing. Shape quality histogram before (c) and after (d) smoothing.

	Before	After
Minimum	0.000	0.768
Maximum	0.994	0.971
Mean	0.749	0.907
Std. dev.	0.286	0.057

Table 4.8: Shape-based quality statistical values for Figure 4.8

Figure 4.9(a) shows a convex cylinder with radius $r = 5$ and height $z = 15$ with non-planar cap surfaces. An hexahedral mesh was created inside the domain with tangled elements. It is important to point out that in this example, several tangled elements are prescribed next to the cap surfaces.

Figure 4.9(b) shows the resulting mesh after the smoothing process. Figures 4.9(c)

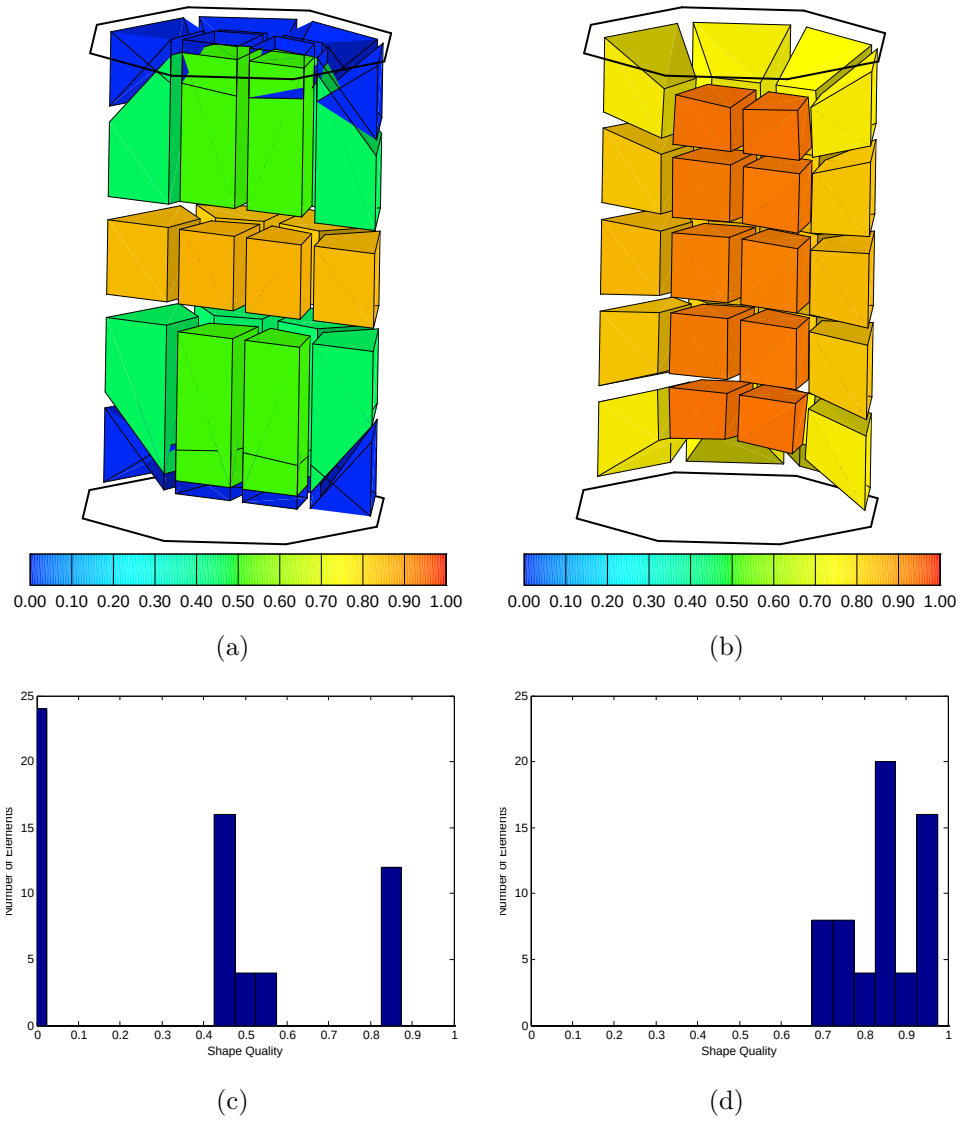


Figure 4.9: Example of smoothing a hexahedral mesh. Mesh configuration before (a) and after (b) smoothing. Shape quality histogram before (c) and after (d) smoothing.

and 4.9(d) show the distribution of the elements according to the shape quality index before and after the smoothing process, respectively. Table 4.9 summarizes the statistical values for Figure 4.9. Note that there are no elements have shape quality index lower than 0.71

	Before	After
Minimum	0.000	0.712
Maximum	0.874	0.945
Mean	0.357	0.844
Std. dev.	0.323	0.075

Table 4.9: Shape-based quality statistical values for Figure 4.9

4.2 Comparison between minimization method

The shape-based smoother incorporates several minimization methods for the objective function. These methods are *Line-Search*, *Newton-Raphson* and *BFGS*. In this section, a comparison between the performance of the minimization methods will be presented.

The number of iterations needed and the elapsed time[1] are the parameters used to compare the performance of the minimization methods. Two examples of meshes conformed by hexahedrons are presented (See Figures 4.10 and 4.11).

Table 4.10: Performance comparison between minimization methods for Figure 4.10. Mesh conformed by 120 nodes, 60 elements

Method	Iterations	Time (s)	Min shape quality
Newton-Raphson	9	2.382	0.7449
BFGS	9	1.178	0.7450
Line-Search	15	1.560	0.7459

[1]measure with the `time` linux command, averaging five runs. The comparison were performed on a Linux workstation with 4GB of RAM with two Intel Core2 Duo CPU E4800 running at 3.00GHZ, Linux kernel `2.6.24-26`

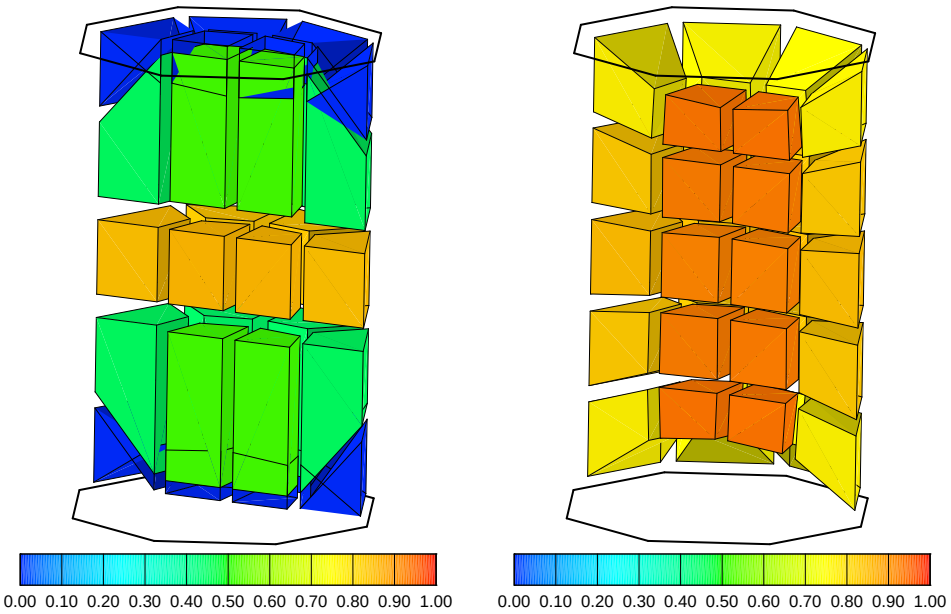


Figure 4.10: Performance comparison hexahedral mesh. Mesh conformed by 120 Nodes, 60 Elements. Mesh configuration before (a) and after (b) smoothing.

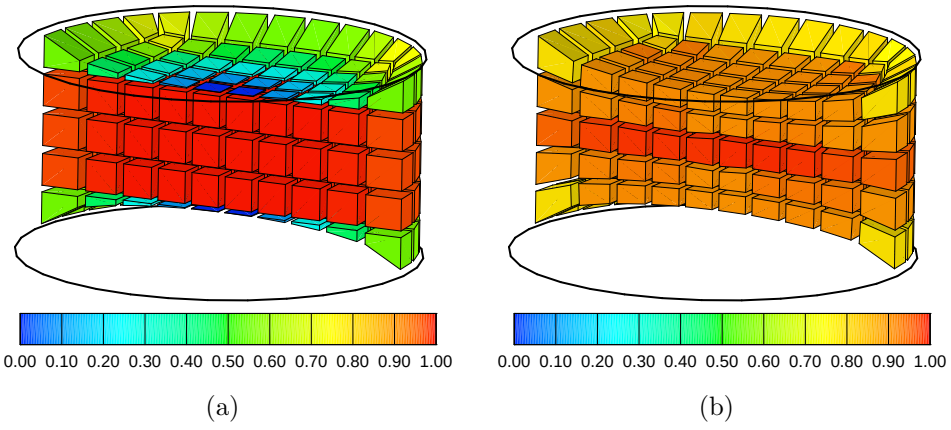


Figure 4.11: Performance comparison hexahedral mesh. Mesh conformed by 678 Nodes, 480 Elements. Mesh configuration before (a) and after (b) smoothing.

The *Newton-Raphson* method requires the most time to smooth a given mesh, due to the computation of the hessian (**H**) of the objective function, as shown in Tables 4.10 and 4.11, requiring up to 41% more time to smooth a mesh. For tangled mesh, the

Table 4.11: Performance comparison between minimization methods for Figure 4.10. Mesh conformed by 678 Nodes, 480 Elements

Method	Iterations	Time (s)	Min shape quality
Newton-Raphson	9	12.442	0.7666
BFGS	9	8.830	0.7667
Line-Search	11	7.372	0.7668

BFGS requires less time to smooth the hexahedral elements (see Table 4.10).

4.3 Robustness of the proposed method

Hermansson and Hansbo (2003) detail how the Giuliani (1982) method fails for convex domains with stretched elements. In the article, they present a mesh configuration in which the Giuliani smoothing method fails, which is duplicated in Figure 4.12. The shape-based smoother will be compared with to the Giuliani smoothing method. The Giuliani smoothing method is one of the most commonly used methods for smoothing meshes. It has high performance and low computational requirements. But, it fails for convex domains with stretched elements, as shown in Figure 4.13(a). The Giuliani method fails for convex meshes because it just considers geometrical information about the elements and not takes into account the quality of the mesh.

Figure 4.12 depictures a six quadrilateral mesh, conformed by eleven nodes, being all nodes fixed but node $\mathbf{x}_5$. The coordinates of the boundary nodes are shown in table 4.12.

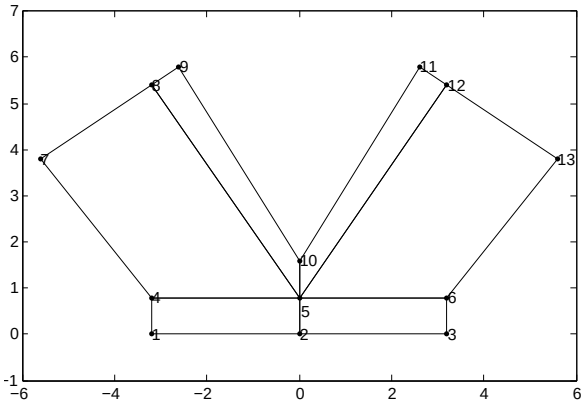


Figure 4.12: Mesh used for comparison to the Giuliani smoother.

$\mathbf{x}_1 = (-3.2, \quad 0.0)$
$\mathbf{x}_2 = (\quad 0.0, \quad 0.0)$
$\mathbf{x}_3 = (\quad 3.2, \quad 0.0)$
$\mathbf{x}_4 = (-3.2, \quad 0.8)$
$\mathbf{x}_6 = (\quad 3.2, \quad 0.8)$
$\mathbf{x}_7 = (-5.6, \quad 3.8)$
$\mathbf{x}_8 = (-3.2, \quad 5.4)$
$\mathbf{x}_9 = (-2.6, \quad 5.8)$
$\mathbf{x}_{10} = (\quad 0.0, \quad 1.6)$
$\mathbf{x}_{11} = (\quad 2.6, \quad 5.8)$
$\mathbf{x}_{12} = (\quad 3.2, \quad 5.4)$
$\mathbf{x}_{13} = (\quad 5.6, \quad 3.8)$

Table 4.12: Coordinates of boundary nodes

Table 4.13 shows the position of node $\mathbf{x}_5$ before and after the smoothing process. The location of node $\mathbf{x}_5$ obtained using the Giuliani smoother is outside of the domain ($\mathbf{x}_5 = (0.0, 2.014)$) generating a tangled mesh. The shape-based smoother locates the node 5 inside the domain ($\mathbf{x}_5 = (0.0, 0.545)$).

	Initial Pos.	Giuliani	Smoother-Shape
Node 5	0.800	2.014	0.545

Table 4.13: Position of node $\mathbf{x}_5$, y coordinate

The shape quality index of the elements of the mesh used for the comparison between smoothing methods are listed in Table 4.14.

Element No.	Initial Pos.	Giuliani	Smoother-Shape
1	0.159	0.000	0.193
2	0.470	0.414	0.329
3	0.707	0.880	0.667
4	0.707	0.880	0.667
5	0.159	0.000	0.193
6	0.470	0.414	0.329

Table 4.14: Shape quality index corresponding to all elements of initial and smoothed meshes

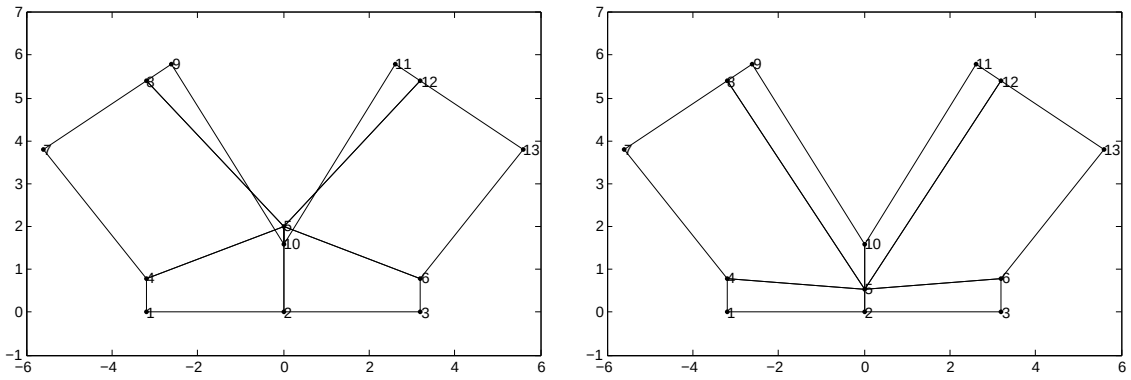


Figure 4.13: Mesh comparison between Giuliani and proposed shape-based smoother. (a) Giuliani's Method. (b) Shape-based Smoother.

Figures 4.13(a) and 4.13(b) show the resulting mesh after smoothing using the Giuliani and shape-based methods, respectively.

Chapter 5

Conclusions

This chapter presents the conclusions and contributions of this work. Also, the future work is listed.

5.1 Conclusions

The proposed procedure is able to untangle quadrilateral and hexahedral meshes, improving the quality of the elements by modifying the location of the inner nodes. The optimized location of the nodes is obtained by minimizing a smoothed objective function based on the shape quality index, developed by Knupp (2001), extending to quadrilateral and hexahedral meshes the method proposed by Escobar et al. (2003).

From the results presented in Section 4.2, the *BFGS* method and *Line-Search* are the best options for the minimization methods, due the computational cost of obtaining the hessian (**H**) of the objective function at each *local* iteration, which is needed by the *Newton-Raphson* method. By adding *Line-Search* to the *BFGS* method, the performance of the minimizer could be improved.

The robustness of the proposed smoother is proven by comparing it to one of the most used smoothing methods, as shown in Section 4.3.

5.2 Contributions

This works expand the work developed by Knupp (2003b) and Escobar et al. (2003) by adding the untangling of quadrilateral and hexahedral elements to the process of smoothing meshes.

5.3 Future Work

The work performed for this investigation leaves open some areas for future research and improvement, some of these areas are:

- *Optimize the code.* Optimize the procedure that computes the objective function and its derivatives. Use a profiler to optimize the code for performance.

- Use `Maple` or other symbolic software to obtain optimized `C++` code of the objective function derivatives.

- *Mixed element meshes.* The code properly smoothes and untangles meshes composed of just one type of elements. Implement the required modifications, to smooth and untangle meshes with mixed elements (i.e., mesh conformed of triangular and quadrilateral elements).

- *Higher order elements.* It can be of interest to extend the proposed method to smooth and untangle high order meshes.

- *2D elements in* $\mathbb{R}^3$. Implement the smoothing and untangling of triangular and quadrilateral elements in $\mathbb{R}^3$. Constrains to the minimization procedure to keep the elements on the target surface have to be added.

- *Node sliding.* The proposed smoother only relocates the internal nodes, but sliding of external nodes may be required to further improve the quality of the mesh (see Figure 5.1).

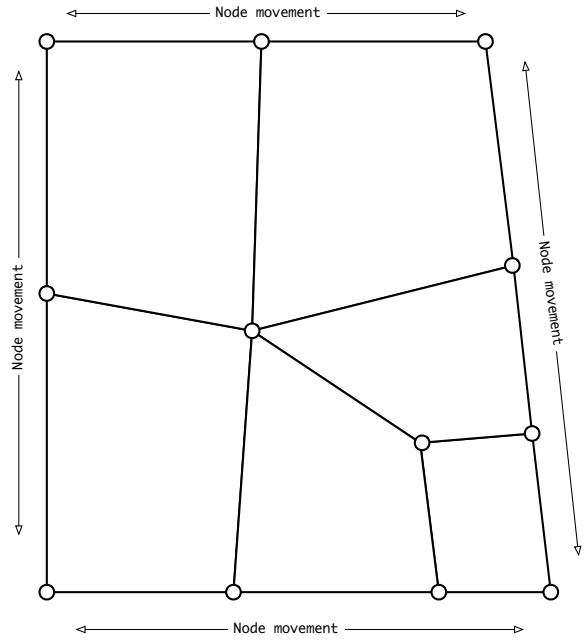


Figure 5.1: Node sliding option for the smoother.

- *Hybrid smoother.* Depending on the shape quality index of a given element (i.e., if it is tangled or not), the smoother should be able to switch between different smoothing techniques (i.e., Shape-based smoother, Giuliani).

- *Performance of the minimization method.* Improve the performance of the minimization methods, (i.e., add *Line-Search* procedure to the *BFGS* minimization method).

Appendices

Appendix A

Derivatives of the objective function

The procedure to obtain the derivatives used in chapter 3 will be detailed in this section. First, the derivatives of shape metric (η) will be shown, then the expression for the derivatives of the objective function will be presented.

A.1 Derivatives of the shape metric

Knupp (2001) and Escobar et al. (2003) define the shape metric (η) as

$$\eta = \frac{|\mathbf{S}|^2}{nh(\sigma)^{2/n}} \tag{A.1}$$

where n is equal to 2 for two dimensional elements (triangular or quadrilateral), or equal to 3 for three dimensional elements (tetrahedral or hexahedral). Recall that $h(\sigma)$, $\mathbf{S}$ and σ are defined as:

$$h(\sigma) = \frac{1}{2}\left(\sigma + \sqrt{\sigma^2 + 4\delta^2}\right) \tag{A.2}$$

$$\mathbf{S} = \mathbf{A}\mathbf{W}^{-\mathbf{1}} \tag{A.3}$$

$$\sigma = \det(\mathbf{S}) \tag{A.4}$$

where δ is a parameter used to smooth the objective function (Escobar et al., 2003).

To obtain the derivatives of equation A.1, we need to apply the chain rule and derivate against to generic coordinates α and β. α and β could be any coordinate (x, y, z) of any of the nodes of the element.

The derivative of equation A.3 with respect to α_i is defined as,

$$\frac{\partial \mathbf{S}}{\partial \alpha_i} = \frac{\partial \mathbf{A}}{\partial \alpha_i}\mathbf{W}^{-1} + \mathbf{A}\frac{\partial \mathbf{W}^{-1}}{\partial \alpha_i} \tag{A.5}$$

Since $\mathbf{W}$ does not change over the elements, we have

$$\frac{\partial \mathbf{W}^{-1}}{\partial \alpha_i} = 0$$

Recall that $\mathbf{W}$ is the affine mapping from the *reference* to *ideal* element and does not depend on the coordinates of the given element.

$$\frac{\partial \mathbf{S}}{\partial \alpha_i} = \frac{\partial \mathbf{A}}{\partial \alpha_i}\mathbf{W}^{-1} \tag{A.6}$$

The derivative of $\mathbf{S}$ (or the derivative of $\mathbf{A}$) will depend on the coordinates of the element and on the local numbering of the nodes of the element.

For a tetrahedral element, the derivatives of $\mathbf{A}$ with respect to α_i are given by

$$\frac{\partial \mathbf{A}}{\partial x_0} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \frac{\partial \mathbf{A}}{\partial y_0} = \begin{bmatrix} 0 & 0 & 0 \\ -1 & -1 & -1 \\ 0 & 0 & 0 \end{bmatrix} \qquad \frac{\partial \mathbf{A}}{\partial z_0} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

$$\frac{\partial \mathbf{A}}{\partial x_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \frac{\partial \mathbf{A}}{\partial y_1} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \frac{\partial \mathbf{A}}{\partial z_1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{A}}{\partial x_2} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \frac{\partial \mathbf{A}}{\partial y_2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \frac{\partial \mathbf{A}}{\partial z_2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{A}}{\partial x_3} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \frac{\partial \mathbf{A}}{\partial y_3} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \qquad \frac{\partial \mathbf{A}}{\partial z_3} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The derivative of σ with respect to α_i, it is required to compute the derivative of Equation (A.2), is defined as:

$$\frac{\partial \sigma}{\partial \alpha_i} = \frac{\partial \det(\mathbf{S})}{\partial \alpha_i} = \det(\mathbf{S}) \mathrm{tr} \left(\mathbf{S}^{-1} \frac{\partial \mathbf{S}}{\partial \alpha_i} \right) \tag{A.7}$$

The derivatives of shape metric (η) with respect to α_i is defined as

$$\frac{\partial \eta}{\partial \alpha_i} = \frac{\partial |\mathbf{S}|^2}{\partial \alpha_i} \frac{1}{n h^{2/n}} + |\mathbf{S}|^2 \frac{\partial}{\partial \alpha_i} \left(\frac{1}{n h^{2/n}} \right) \tag{A.8}$$

Since $|\mathbf{S}| = \sqrt{\mathrm{tr}(\mathbf{S}^T \mathbf{S})}$ or $\sqrt{(\mathbf{S}, \mathbf{S})}$, leading to

$$\frac{\partial \eta}{\partial \alpha_i} = \frac{1}{n} \left(\frac{\partial (\mathbf{S}, \mathbf{S})}{\partial \alpha_i} \frac{1}{h^{2/n}} + (\mathbf{S}, \mathbf{S}) \frac{\partial}{\partial \alpha_i} \left(\frac{1}{h^{2/n}} \right) \right) \tag{A.9}$$

To facilitate the development of Equation (A.9), it will separated in several terms. The first term of Equation (A.9) can be expressed as

$$\frac{\partial (\mathbf{S}, \mathbf{S})}{\partial \alpha_i} \frac{1}{h^{2/n}} = 2 \Big(\frac{\partial \mathbf{S}}{\partial \alpha_i}, \mathbf{S} \Big) \frac{1}{h^{2/n}} \tag{A.10}$$

The second term of Equation (A.9) follows

$$\frac{\partial}{\partial \alpha_i} \left(\frac{1}{h^{2/n}} \right) = -\frac{2}{n} h^{-2/n-1} \left(\frac{1}{2} \left(\frac{\partial \sigma}{\partial \alpha_i} + \frac{1}{2} \left(\sigma^2 + 4\delta^2 \right)^{-1/2} 2\sigma \frac{\partial \sigma}{\partial \alpha_i} \right) \right) \tag{A.11}$$

where h is defined by Equation (A.2),

$$\begin{aligned}
\frac{\partial}{\partial\alpha_i}\left(\frac{1}{h^{2/n}}\right) &= -\frac{2}{n}h^{-\frac{2+n}{n}}\left(\frac{1}{2}\frac{\partial\sigma}{\partial\alpha_i}\right)\left(\frac{\sqrt{\sigma^2+4\delta^2}+\sigma}{\sqrt{\sigma^2+4\delta^2}}\right) \\
&= -\frac{1}{n}\frac{1}{h^{\frac{2+n}{n}}}\left(\frac{\partial\sigma}{\partial\alpha_i}\right)\left(\frac{\sqrt{\sigma^2+4\delta^2}+\sigma}{\sqrt{\sigma^2+4\delta^2}}\right) \\
&= -\frac{1}{n}\frac{1}{\left[\frac{1}{2}\left(\sigma+\sqrt{\sigma^2+4\delta^2}\right)\right]^{\frac{2+n}{n}}}\left(\frac{\partial\sigma}{\partial\alpha_i}\right)\left(\frac{\sqrt{\sigma^2+4\delta^2}+\sigma}{\sqrt{\sigma^2+4\delta^2}}\right) \\
&= -\frac{2}{n}\frac{1}{\frac{1}{2}^{\frac{2}{n}}\left(\sigma+\sqrt{\sigma^2+4\delta^2}\right)^{\frac{2}{n}}}\left(\frac{\partial\sigma}{\partial\alpha_i}\right)\left(\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right) \\
&= -\frac{2}{n}\frac{1}{h^{\frac{2}{n}}}\left(\frac{\partial\sigma}{\partial\alpha_i}\right)\left(\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right) \qquad \text{(A.12)}
\end{aligned}$$

Using Equations (A.10) and (A.12), an expression for the first derivative of the shape metric (η) with respect to α_i is obtained,

$$\frac{\partial\eta}{\partial\alpha_i} = 2\eta\left[\left(\frac{\partial\mathbf{S}}{\partial\alpha_i},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2} - \frac{1}{n}\frac{\partial\sigma}{\partial\alpha_i}\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right] \qquad \text{(A.13)}$$

To generate any given derivative, α_i is replaced by x, y or z and $i = 0, 1, .., j-1$, where i is the local identification number of a node attached to a given element[1].

The second derivatives of the shape metric (η) are computed following the same procedure used for obtaining the expressions for the first derivatives and using Equation (A.13) as a starting point,

$$\begin{aligned}
\frac{\partial^2\eta}{\partial\alpha_i\partial\alpha_j} = 2\frac{\partial\eta}{\partial\alpha_j}&\left[\left(\frac{\partial\mathbf{S}}{\partial\alpha_i},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2} - \frac{1}{n}\frac{\partial\sigma}{\partial\alpha_i}\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right] + \\
&2\eta\frac{\partial}{\partial\alpha_j}\left[\left(\frac{\partial\mathbf{S}}{\partial\alpha_i},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2} - \frac{1}{n}\frac{\partial\sigma}{\partial\alpha_i}\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right] \qquad \text{(A.14)}
\end{aligned}$$

Equation (A.14) will be separated into several terms, in order to facilitate the process

[1] $j = 3$ for triangular elements, $j = 4$ for tetrahedral and quadrilateral elements, $j = 8$ for hexahedral elements.

of the computation of the second derivatives of the shape metric (η). The first term of Equation (A.14) follows:

$$2\frac{\partial\eta}{\partial\alpha_j}\left[\left(\frac{\partial\mathbf{S}}{\partial\alpha_i},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2}-\frac{1}{n}\frac{\partial\sigma}{\partial\alpha_i}\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right]=A\times B \tag{A.15}$$

where A is defined as

$$\begin{aligned}A&=2\eta\left[\left(\frac{\partial\mathbf{S}}{\partial\alpha_j},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2}-\frac{1}{n}\frac{\partial\sigma}{\partial\alpha_j}\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right]\\&=\frac{\partial\eta}{\partial\alpha_j}\end{aligned} \tag{A.16}$$

and B is defined as

$$\begin{aligned}B&=2\left[\left(\frac{\partial\mathbf{S}}{\partial\alpha_i},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2}-\frac{1}{n}\frac{\partial\sigma}{\partial\alpha_i}\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right]\\&=\frac{1}{\eta}\frac{\partial\eta}{\partial\alpha_i}\end{aligned} \tag{A.17}$$

Replacing Equations (A.16) and (A.17) into Equation (A.15), we obtain

$$2\frac{\partial\eta}{\partial\alpha_j}\left[\left(\frac{\partial\mathbf{S}}{\partial\alpha_i},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2}-\frac{1}{n}\frac{\partial\sigma}{\partial\alpha_i}\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right]=\frac{1}{\eta}\frac{\partial\eta}{\partial\alpha_i}\frac{\partial\eta}{\partial\alpha_j} \tag{A.18}$$

The second term of equation A.14 will be splitted again in two terms. The first term follows:

$$2\eta\frac{\partial}{\partial\alpha_j}\left[\left(\frac{\partial\mathbf{S}}{\partial\alpha_i},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2}\right] = 2\eta\left[\left(\frac{\partial^2\mathbf{S}}{\partial\alpha_i\partial\alpha_j},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2} + \left(\frac{\partial\mathbf{S}}{\partial\alpha_i},\frac{\partial\mathbf{S}}{\partial\alpha_j}\right)\frac{1}{|\mathbf{S}|^2} + \left(\frac{\partial\mathbf{S}}{\partial\alpha_i},\mathbf{S}\right)\frac{\partial}{\partial\alpha_j}\left(|\mathbf{S}|^{-2}\right)\right]$$
$$= 2\eta\left[\left(\frac{\partial^2\mathbf{S}}{\partial\alpha_i\partial\alpha_j},\mathbf{S}\right)\frac{1}{|\mathbf{S}|^2} + \left(\frac{\partial\mathbf{S}}{\partial\alpha_i},\frac{\partial\mathbf{S}}{\partial\alpha_j}\right)\frac{1}{|\mathbf{S}|^2} - 2\frac{1}{|\mathbf{S}|^4}\left(\frac{\partial\mathbf{S}}{\partial\alpha_j},\mathbf{S}\right)\right] \tag{A.19}$$

Now, the second term is defined as:

$$2\eta\frac{\partial}{\partial\alpha_j}\left(-\frac{1}{n}\frac{\partial\sigma}{\partial\alpha_i}\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right) = -\frac{2\eta}{n}\left(\frac{\partial^2\sigma}{\partial\alpha_i\partial\alpha_j}\frac{1}{\sqrt{\sigma^2+4\delta^2}} + \frac{\partial\sigma}{\partial\alpha_i}\frac{\partial}{\partial\alpha_j}\left(\frac{1}{\sqrt{\sigma^2+4\delta^2}}\right)\right)$$
$$= -\frac{2\eta}{n}\left(\frac{\partial^2\sigma}{\partial\alpha_i\partial\alpha_j}\frac{1}{\sqrt{\sigma^2+4\delta^2}} - \frac{1}{2}\sigma\frac{\partial\sigma}{\partial\alpha_i}\frac{\partial\sigma}{\partial\alpha_j}\frac{1}{\sqrt[3]{\sigma^2+4\delta^2}}\right) \tag{A.20}$$

In Equations (A.19) and (A.20), the second order derivatives of $\mathbf{S}$ and σ are zero, since they are linear functions of the coordinate components

$$\frac{\partial^2\mathbf{S}}{\partial\alpha_i\partial\alpha_j} = 0.0 \qquad \frac{\partial^2\sigma}{\partial\alpha_i\partial\alpha_j} = 0.0$$

Replacing Equations (A.18), (A.19) and (A.20) into Equation (A.15), we obtain an expression for the second derivative of the shape metric

$$\frac{\partial^2\eta}{\partial\alpha_i\partial\alpha_j} = \frac{1}{\eta}\frac{\partial\eta}{\partial\alpha_i}\frac{\partial\eta}{\partial\alpha_j} + 2\eta\left[\left(\frac{\partial\mathbf{S}}{\partial\alpha_i},\frac{\partial\mathbf{S}}{\partial\alpha_j}\right)\frac{1}{|\mathbf{S}|^2} - 2\frac{1}{|\mathbf{S}|^4}\left(\frac{\partial\mathbf{S}}{\partial\alpha_i},\mathbf{S}\right)\left(\frac{\partial\mathbf{S}}{\partial\alpha_j},\mathbf{S}\right) + \frac{\sigma}{n}\frac{\partial\sigma}{\partial\alpha_i}\frac{\partial\sigma}{\partial\alpha_j}\frac{1}{\sqrt[3]{\sigma^2+4\delta^2}}\right] \tag{A.21}$$

A.2 Derivatives of the objective function

In this section, the derivatives for the objective function will be presented. First for triangular and tetrahedral elements, and then, for the quadrilateral and hexahedral elements.

Triangular and tetrahedral elements objective function

The objective function for a triangular or tetrahedral element is shown in Equation (A.22), the function depends on the inverse of the shape quality index:

$$f(q_n) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{q_n} - 1.0 \tag{A.22}$$

Since the shape quality index is defined as the inverse of the shape metric ($q_n = 1/\eta$) and N is the number of element surrounding the node, Equation (A.22), could be expressed as

$$f(\eta) = \frac{1}{N}\sum_{n=1}^{N}\eta_n - 1.0 \tag{A.23}$$

The derivatives for the objective functions are straight forward and have the following shape,

$$\frac{\partial f(\eta)}{\partial \alpha_i} = \frac{1}{N}\sum_{n=1}^{N}\frac{\partial \eta_n}{\partial \alpha_i} \tag{A.24}$$

Equation (A.24) shows that the derivative of the objective function with respect to α is the average of the derivatives of shape metric (η) with respect to α of each of the elements attached to the node.

For minimization methods that require the calculation of the Hessian, as *Newton-Raphson* method, the second derivatives of the objective function are required. When dealing with triangular or tetrahedral elements, the the second derivatives of the

objective function are given by:

$$\frac{\partial^2 f(\eta)}{\partial \alpha_i \partial \alpha_j} = \frac{1}{N} \sum_{n=1}^{N} \frac{\partial^2 \eta}{\partial \alpha_i \partial \alpha_j} \tag{A.25}$$

where, N is the number of elements attached the given node and i is the coordinate on which the derivative is computed with respect to. The second derivatives of the shape metric ($\frac{\partial^2 \eta}{\partial \alpha_i \partial \alpha_j}$) are shown in Equation (A.21).

Quadrilateral and hexahedral elements objective function

The objective function expression (Knupp, 2003a,b) for quadrilateral or hexahedral elements is:

$$f(\eta) = \frac{1}{mN} \sum_{n=1}^{N} \sum_{k=1}^{m} \left(\eta_{(n,k)}\right)^2 - 1.0 \tag{A.26}$$

where $m = 4$ for quadrilateral elements and $m = 8$ for hexahedrons. N is the number of elements attached to the node.

The derivatives of the objective function $f(\eta)$ with respect to α_i are equal to

$$\begin{aligned} \frac{\partial f(\eta)}{\partial \alpha_i} &= \frac{1}{mN} \sum_{n=1}^{N} \sum_{k=1}^{K} \frac{\partial}{\partial \alpha_i} \left(\left(\eta_{(n,k)}\right)^2 \right) \\ &= \frac{2}{mN} \sum_{n=1}^{N} \sum_{k=1}^{K} \left(\left(\eta_{(n,k)}\right) \frac{\partial \eta_{(n,k)}}{\partial \alpha_i} \right) \end{aligned} \tag{A.27}$$

where the term $\frac{\partial \eta}{\partial \alpha_i}$ is shown in Equation (A.13). The derivatives of the objective function $f(\eta)$ with respect to α_i will depend on two components, the shape metric (η) and its derivative ($\frac{\partial \eta}{\partial \alpha_i}$).

The second derivatives of the objective function for quadrilateral and hexahedral elements are defined as:

$$\frac{\partial^2 f(\eta)}{\partial \alpha_i \partial \alpha_j} = \frac{2}{mN} \sum_{n=1}^{N} \sum_{k=1}^{m} \left(\frac{\partial \eta_{(n,k)}}{\partial \alpha_j} \frac{\partial \eta_{(n,k)}}{\partial \alpha_i} + \eta_{(n,k)} \frac{\partial^2 \eta_{(n,k)}}{\partial \alpha_i \partial \alpha_j} \right) \tag{A.28}$$

Equation (A.28) could be expressed in a hessian form as

$$\frac{\partial^2 f(\eta)}{\partial \alpha^2} = \frac{2}{mN} \sum_{n=1}^{N} \sum_{k=1}^{m} \left(\mathbf{G}\mathbf{G}^{\mathrm{T}} + \eta \mathbf{H} \right) \tag{A.29}$$

where $\mathbf{G}$ and $\mathbf{H}$ are the gradient and hessian of shape metric (η), respectively, defined as

$$\mathbf{G} = \begin{bmatrix} \frac{\partial \eta}{\partial x_i} \\ \frac{\partial \eta}{\partial y_i} \\ \frac{\partial \eta}{\partial z_i} \end{bmatrix} \tag{A.30}$$

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 \eta}{\partial x_i^2} & \frac{\partial^2 \eta}{\partial x_i \partial y_i} & \frac{\partial^2 \eta}{\partial x_i \partial z_i} \\ \frac{\partial^2 \eta}{\partial y_i \partial x_i} & \frac{\partial^2 \eta}{\partial y_i^2} & \frac{\partial^2 \eta}{\partial y_i \partial z_i} \\ \frac{\partial^2 \eta}{\partial z_i \partial x_i} & \frac{\partial^2 \eta}{\partial z_i \partial y_i} & \frac{\partial^2 \eta}{\partial z_i^2} \end{bmatrix} \tag{A.31}$$

where i stands for the index of the node in the mesh which is moving.

Appendix B

Using OpenCASCADE libraries as a minimization tool

This chapter is used to explain how the OpenCASCADE libraries are used as a minimization tool. OpenCASCADE is described as a 3D surface and solid modeling, visualization and data exchange framework, but also contains math libraries for minimization, among other features.

The steps required to use the minimization libraries included with OpenCASCADE in a C++ project are:

1. Create a class derived from `math_MultipleVarFunctionWithHessian` or `math_MultipleVarFunctionWithGradient` (depending on the type of minimization desired), add the proper header files with the `#include` statement.

2. In the implementation of the class, override the methods `NbVariables`, `Value`, `Gradient`, `Hessian` and `Values`, to match the function to be minimized.

   - `NbVariables`. Must return the number of independent variables.
   - `Value`. Returns the value of the function to minimize, (f).
   - `Gradient`. Evaluates the gradient of the function to minimize,(∇f).
   - `Hessian`. Evaluates the hessian of the function to minimize, $(\mathbf{H})$.
   - `Values`. Evaluates the methods `Value`, `Gradient` and `Hessian`.

3. Declare a new object using the class created above (Item 1), which contains the information about the function to minimize.

4. Create a new object (the minimizer) of type `math_NewtonMinimum` or `math_BFGS`, depending on which minimization method will be used. During the declaration of the minimizer, the following variables should be passed as arguments: the object which holds the function to minimize (Item 3), the starting point of the minimization (declared as `math_Vector` object), a convergence tolerance, and the maximum number of iterations to perform. Recall that it is required to include the proper header files, using the `#include` statement.

5. Call the method `Perform()` of minimizer object. The code will execute and try to obtain a minimum for the provided function.

6. Check that the minimization reached a solution by using the minimizer object `IsDone()`, which will return a boolean stating if a minimum was found.

7. If a minimum was successfully found, the location of the minimum can be using the method `Location()`, which return a variable of type `math_Vector`. The number of iterations required to obtain a minimum is obtained by calling the method `NbIterations()`..

The following `C++` example was created to show how to use the `OpenCASCADE` libraries as a minimization tool. The code below tries to minimize the function by using the *Newton-Raphson* and the *BFGS* methods:

$$f(x_1, x_2) = 3.0 \sin\left(\frac{1}{2} + \frac{1}{4}x_1 x_2\right) \cos(x_1) \tag{B.1}$$

The location of the minimum for $f(x_1, x_2)$, the value of the function f at the minimum and the number of iterations required are shown in table B.1. The starting point for the minimization process example was set to $(0.5, 0.5)$ (see Venkataraman, 2001).

Method	x_1	x_2	$f_{\min}(x_1, x_2)$	No. of Iterations
Analytical	π	$2\left(\frac{\pi-1}{\pi}\right)$	-3	-
Newton-Raphson	3.14159	1.36338	-3	9
BFGS	3.14159	1.36338	-3	8

Table B.1: Minimization results for $f(x_1, x_2)$ (Equation B.1)

The C++ code follows,

```
#include <math_BFGS.hxx>
#include <math_NewtonMinimum.hxx>
#include <math_MultipleVarFunctionWithHessian.hxx>
#include <math_MultipleVarFunctionWithGradient.hxx>

class GoalFunctionNR : public math_MultipleVarFunctionWithHessian
{
public:

    GoalFunctionNR();

    Standard_Integer
        NbVariables() const;

    Standard_Boolean
        Value(const math_Vector& X,Standard_Real& F);

    Standard_Boolean
        Gradient(const math_Vector& X, math_Vector& G) ;

    Standard_Boolean
        Values(const math_Vector&   X,Standard_Real& F, math_Vector& G) ;

    Standard_Boolean
        Values(const math_Vector&   X,Standard_Real& F, math_Vector& G,
                math_Matrix& H) ;

    Standard_Boolean
        Hessian(const math_Vector& X, math_Matrix& H) ;
};
// ---------------------------------------------------------------------------
GoalFunctionNR::GoalFunctionNR()
{
}

Standard_Integer
GoalFunctionNR::NbVariables() const
{
    return 2;
}

Standard_Boolean
GoalFunctionNR::Value(const math_Vector& X,Standard_Real& F)
{
    F = 3.0 * sin( 0.5 + 0.25 * X(1) * X(2) ) * cos( X(1) );
    return true;
```

```
}

Standard_Boolean
GoalFunctionNR::Gradient(const math_Vector& X, math_Vector& G)
{
    G(1) = 3.0/4.0*cos(1.0/2.0 + 1.0/4.0 * X(1)*X(2) ) *X(2) * cos(X(1)) -
           3.0 * sin( 0.5+0.25 *X(1)*X(2) )* sin(X(1)) ;
    G(2) = 3.0/4.0*cos(1.0/2.0 + 1.0/4.0 * X(1)*X(2) ) *X(1) * cos(X(1)) ;
    return true;
}

Standard_Boolean
GoalFunctionNR::Values(const math_Vector& X,Standard_Real& F, math_Vector& G)
{
    Standard_Boolean v1, g1;
    v1 = Value(X,F);
    g1 = Gradient(X,G);
    return ( v1 && g1);
}

Standard_Boolean
GoalFunctionNR::Values(const math_Vector& X,Standard_Real& F, math_Vector& G,
                       math_Matrix& H)
{
    Standard_Boolean v1, g1 , h1;
    v1 = Value(X,F);
    g1 = Gradient(X,G);
    h1 = Hessian(X,H) ;
    return ( v1 && g1 && h1 );
}

Standard_Boolean
GoalFunctionNR::Hessian(const math_Vector& X, math_Matrix& H)
{
    H(1,1) =    -3.0/16.0*sin(1.0/2.0+1.0/4.0*X(1)*X(2))*X(2)*X(2)*cos(X(1))-
                3/2*cos(1.0/2.0+1.0/4.0*X(1)*X(2))*X(2)*sin(X(1))-
                3*sin(1.0/2.0+1.0/4.0*X(1)*X(2))*cos(X(1)) ;

    H(1,2) =    -3.0/16.0*sin(1.0/2.0+1.0/4.0*X(1)*X(2))*X(1)*X(2)*cos(X(1))+
                3.0/4.0*cos(1.0/2.0+1.0/4.0*X(1)*X(2))*cos(X(1))-
                3.0/4.0*cos(1.0/2.0+1.0/4.0*X(1)*X(2))*X(1)*sin(X(1)) ;

    H(2,2) =    -3.0/16.0*sin(1.0/2.0+1.0/4.0*X(1)*X(2))*X(1)*X(1)*cos(X(1)) ;

    H(2,1) =    -3.0/16.0*sin(1.0/2.0+1.0/4.0*X(1)*X(2))*X(1)*X(2)*cos(X(1))+
                3.0/4.0*cos(1.0/2.0+1.0/4.0*X(1)*X(2))*cos(X(1))-
                3.0/4.0*cos(1.0/2.0+1.0/4.0*X(1)*X(2))*X(1)*sin(X(1)) ;
    return true;
}

class GoalFunction : public math_MultipleVarFunctionWithGradient
{
public:

    GoalFunction();

    Standard_Integer
        NbVariables() const;

    Standard_Boolean
        Value(const math_Vector& X,Standard_Real& F);
```

```
    Standard_Boolean
        Gradient(const math_Vector& X, math_Vector& G) ;

    Standard_Boolean
        Values(const math_Vector&   X,Standard_Real& F, math_Vector& G) ;
};

// ---------------------------------------------------------------------------

GoalFunction::GoalFunction()
{
}

Standard_Integer
GoalFunction::NbVariables() const
{
    return 2;
}

Standard_Boolean
GoalFunction::Value(const math_Vector& X,Standard_Real& F)
{
    F = 3.0 * sin( 0.5 + 0.25 * X(1) * X(2) ) * cos( X(1) );
    return true;
}

Standard_Boolean
GoalFunction::Gradient(const math_Vector& X, math_Vector& G)
{
    G(1) = 3.0/4.0*cos(1.0/2.0 + 1.0/4.0 * X(1)*X(2) ) *X(2) * cos(X(1)) -
           3.0 * sin( 0.5+0.25 *X(1)*X(2) )* sin(X(1)) ;
    G(2) = 3.0/4.0*cos(1.0/2.0 + 1.0/4.0 * X(1)*X(2) ) *X(1) * cos(X(1)) ;

    return true;
}

Standard_Boolean
GoalFunction::Values(const math_Vector& X,Standard_Real& F, math_Vector& G)
{
    Standard_Boolean v1, g1;
    v1 = Value(X,F);
    g1 = Gradient(X,G);
    return ( v1 && g1);
}

// ---------------------------------------------------------------------------

int main( int argc, char **argv)
{
// Minimization using Newtown Raphson
    GoalFunctionNR F;
    GoalFunction F1;
    math_Vector X0(1,2);
    X0(1) = ( 1.0 / 2.0 ) ;    X0(2) = ( 1.0 / 2.0 ) ;

    math_NewtonMinimum minimizer(F, X0);
    minimizer.Perform(F,X0);
    math_Vector optimalX(1,2);

    if(minimizer.IsDone()){
        optimalX=minimizer.Location();
        std::cout << "NR:" << std::endl;
```

```
        std::cout << "X: " << optimalX(1) << " Y: " << optimalX(2)
                  << std::endl ;
        std::cout << "fmin: " << minimizer.Minimum() << " with " <<
                    minimizer.NbIterations()  <<
                    ((minimizer.NbIterations()== 1) ?" iteration":" iterations")
                    << std::endl ;
    } else {
        std::cerr << "Minimization has not been accomplised" << std::cout;
    }

// Minimization using BFGS
    math_BFGS minimizerBFGS(F, X0);
    minimizerBFGS.Perform(F,X0);
    math_Vector optimalXBFGS(1,2);

    if(minimizerBFGS.IsDone()){
        optimalXBFGS=minimizerBFGS.Location();
        std::cout << "BFGS: " << std::endl;
        std::cout << "X: " << optimalXBFGS(1) << " Y: " << optimalXBFGS(2)
        << std::endl ;
        std::cout << "fmin: " << minimizerBFGS.Minimum() << " with " <<
        minimizerBFGS.NbIterations()  <<
        ((minimizer.NbIterations()== 1) ?" iteration":" iterations")
        << std::endl ;
    } else {
        std::cerr << "Minimization has not been accomplised" << std::cout;
    }
}
```

For more information about OpenCASCADE (2010), visit `http://www.opencascade.org/`

Bibliography

Arroyo, M. (2009). Lecture Notes: Solution methods for nonlinear problems, Computational Solid Mechanics. `http://www-lacan.upc.es/arroyo/`.

Canann, S. A., M. B. Stephenson, and T. Blacker (1993). Optismoothing: an optimization-driven approach to mesh smoothing. *Finite Elements in Analysis and Design 13*(2-3), 185–190.

Escobar, J. M., E. Rodríguez, R. Montenegro, G. Montero, and J. M. González-Yuste (2003). Simultaneous untangling and smoothing of tetrahedral meshes. *Computer Methods in Applied Mechanics and Engineering 192*(25), 2775–2787.

George, P. and H. Borouchaki (1999). *Delaunay Triangulation and Meshing: Application to Finite Element*. John Wiley & Sons Inc.

Giuliani, S. (1982). An algorithm for continuous rezoning of the hydrodynamic grid in arbitrary lagrangian-eulerian computer codes. *Nucl. Eng. Des. 72*(205), 205–212.

Hermansson, J. and P. Hansbo (2003). A variable diffusion method for mesh smoothing. *Communications in Numerical Methods in Engineering 19*(11), 897–908.

Herrmann, L. R. (1976, October). Laplacian-isoparametric grid generation scheme. *Journal of the Engineering Mechanics Division 102*(5), 749–907.

Knupp, P. M. (2001). Algebraic mesh quality metrics. *Society for Industrial and Applied Mathematics J. Sci. Comput. 23*(1), 193–218.

Knupp, P. M. (2003a). Algebraic mesh quality metrics for unstructured initial meshes. *Finite Elem. Anal. Des. 39*(3), 217–241.

Knupp, P. M. (2003b). A method for hexahedral mesh shape optimization. *International Journal for Numerical Methods in Engineering 58*(2), 319–332.

Knupp, P. M. and S. Steinberg (1993). *Fundamentals of Grid Generation*. CRC Press.

Laboratori de Càlcul Numèric (2010). EZ4U. Mesh Generation Environment. `http://www-lacan.upc.es/ez4u.htm`.

Oddy, A., G. J., M. McDill, and M. Bibby (1988). A Distortion Metric for Isoparametric Finite Elements. *Transactions of Canadian Society for Mechanical 12*(4), 213–217.

OpenCASCADE (2010). `OpenCASCADE` technology, 3d modeling & numerical simulation. `http://www.opencascade.org`.

Roca, X. (2009, July). *Paving the Path Towards Automatic Hexahedral Mesh Generation*. Ph. D. thesis, Facultat de Matemàtiques i Estadística. Universitat Politècnica de Catalunya.

Sandia National Laboratories (2009). CUBIT Geometry and Mesh Generation Toolkit. `http://cubit.sandia.gov/`.

Staten, M. (2007). Why is Hex Meshing so Hard? What is the Cubit Team Doing About it? `http://cubit.sandia.gov/user_meetings.html`.

Thompson, J. F., B. K. Soni, and N. P. Weatherill (1999). *Handbook of Grid Generation*. CRC Press.

Venkataraman, P. (2001). *Applied Optimization with MATLAB Programming*. Wiley-Interscience.