PROJECTE TECNOLÒGIC

Llicenciatura en Matemàtiques

Títol: Smoothing and untangling triangular meshes on surfaces

Autor: Abel Gargallo Peiró

Director: Josep Sarrate Ramos

Departament: MA3

Convocatòria: juny 2010

Facultat de Matemàtiques i Estadística

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Projecte tecnològic

Smoothing and untangling triangular meshes on surfaces

Author:
Abel Gargallo Peiró

Advisor:
Josep Sarrate Ramos

Barcelona
May 26, 2010

Contents

1 Introduction

The Finite Element Method is nowadays one of the most used techniques in applied sciences and engeneering. The application of the method requires a previous discretization of the geometry into a certain type of element depending on the requirements of the problem. This discretization has then to capture the geometry of the object in which the problem is stated, but it also has to approximate the geometry dividing it into elements with certain geometrical requirements in order to carry out the necessary calculations with enough precision. The use of FEM in industrial applications is then slowed down by the need of a generation of a good mesh.

It is well known that the precision of the numerical solution obtained by the FEM depends on the size and the shape of the elements of the mesh. Thus, the precision of the solution is directly related to the quality of the mesh: in order to carry out accurate calculations we have to ensure a good approximation of the domain subjected to a design of a mesh with elements with an acceptable size and shape. On the one hand, several shape quality measures have been defined in order to quantify the deviation of the shape of an element respect to an "ideal" shape. For instance, [Field] presents a comparative analysis of several shape quality measures for triangles and tetrahedrons.

Moreover it is impotant to point out that meshing algorithms are hierarchic procedures. Thus, in order to mesh a 3D object we first have to mesh its 2D boundary. Consequently, the 2D mesh will also require a previous 1D discretization. Therefore, the quality of a 3D mesh is directly affected by the quality of the boundary discretization (surface mesh). Thus, it is of the major importance to generate a high quality surface mesh.

Several techniques have been developed in order to improve the shape of the elements of a given mesh. They can be classified in two main categories. The first one is composed by those procedures that modify the topolgy (the connectivity) of the mesh. That is, some elements are removed or modified in order to generate new ones that have better shape quality. The second ones, are composed by those techniques that modify the geometry of the mesh (the location of the nodes), without changing its topology, in order to obtain a better configuration of the mesh. That is, they "smooth" the location of the nodes.

A wide range of smoothing algorithms have been developed during the last decades (see for instance [Herrman] and [Giuliani], among others). It is important to point out that these methods are based on geometrical and/or numerical reasoning. In general, these algorithms are fast from the computational point of view. However, they are not robust, in the sense that they can move nodes outside of the domain in complex geometries (for instance on convex corners). In addition, these algorithms are not designed to maximize a given quality measure.

[Knupp 01] introduced a family of quality measures placed within an algebraic framework that have been intensively used during the last decade. Later,

[Knupp 03b] proposed a smoothing method based on an optimization of these measures. In fact, this optimization procedure is transformed into a continuous minimization problem.

These optimization algorithms are more robust than the previous ones. However, they are still not able to untangle inverted elements. [Escobar 03] introduced a modification of the measures developed by Knupp in which this lack was covered. The optimization of the new objective function was able to simultaniously untangle and smooth a tetrahedral mesh, saving time and effort in order to obtain the final mesh.

Later, [Escobar 06] extended this algorithms to non-planar triangular meshes. The proposed method introduces an additional optimization problem that increases the computational cost of the global smooth algorithm.

The aim of this work is to work out the basis of shape quality metrics for triangular meshes on surfaces. Then we will develop a simultanious smoothing-untangling procedure based on the method proposed by [Escobar 06], that avoids the additional minimization problem. Finally, several examples will be presented in order to illustrate the capabilities of the proposed method.

2 Algebraic quality measure

2.1 Basics on the quality of an element

The *quality metric* of an element (triangle in 2D problems, tetrahedron in 3D) is a scalar function such that measures a given geometric property of the analysed element. It is usually a function defined on the vertices of the element.

To fix notation, we denote the physical space dimension by n ($n = 2$ for 2D problems and $n = 3$ for 3D problems). Moreover, let m be the number of vertices of the element ($m = 3$ for triangles, $m = 4$ for tetrahedrons), and $\mathbf{x_k} \in \mathbb{R}^n$ the coordinates of those vertices. Taking into account this notation, the quality metric is defined as the following scalar function

$$\begin{array}{rcl} q : \mathbb{R}^n \times \overset{(m)}{\ldots} \times \mathbb{R}^n & \longrightarrow & \mathbb{R} \\ (\mathbf{x_0}, \ldots, \mathbf{x_{m-1}}) & \longrightarrow & q(\mathbf{x_0}, \ldots, \mathbf{x_{m-1}}) \end{array}$$

According to [Knupp 01], any quality metric should hold the following properties:

- q is dimension-free.
- q is going to be referenced to an ideal element that describes the desired shape of the element.
- For all $\mathbf{x_k}$ in the domain, $q(\mathbf{x_0}, \ldots, \mathbf{x_{m-1}}) \in [0, 1]$. That is:

  $$q : \mathbb{R}^n \times \overset{(m)}{\ldots} \times \mathbb{R}^n \longrightarrow [0, 1].$$

  Note that q will only be 1 if the element achieves its ideal configuration, and it will also only be 0 if the element is degenerated[1].
- q is invariable under translations or rotations of the element.
- q does not depend on the numbering of the nodes of the element.

In this work we will use a *shape quality metric* introduced in [Knupp 03]. The aim of this metric is to detect the distorsions in the shape of the element, letting appart its size.

2.2 Jacobian matrix and reference and ideal elements

Let t_R be a *reference element* delimited by vertices $\mathbf{u_0} = (0, 0)$, $\mathbf{u_1} = (1, 0)$ and $\mathbf{u_2} = (0, 1)$ in a logical space. We want to find an affine mapping $\mathbf{f}$ that maps this

[1] A 2D element will be considered degenerated if it has area 0. Equivalently, it will be degenerated in 3D if its volume is 0.

reference element onto a given triangle, t, in the physical space, defined by nodes $\mathbf{x_0} = (x_0, y_0)$, $\mathbf{x_1} = (x_1, y_1)$ and $\mathbf{x_2} = (x_2, y_2)$, see Figure 2.1.

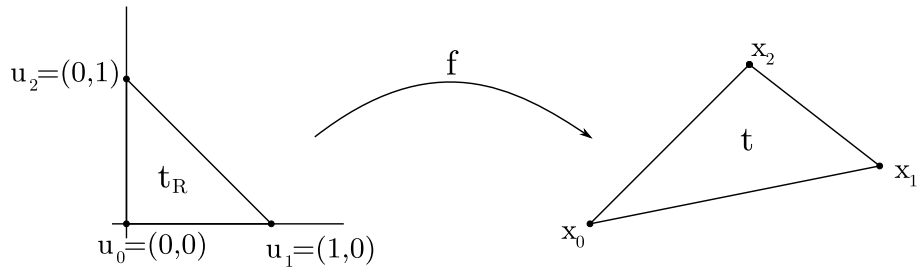

Figure 2.1: Affine mapping between the reference and the physical element.

Let ξ_k, $k = 0, 1, 2$, be the barycentric coordinates of any point inside the reference triangle. Recall that the barycentric coordinates verify that $0 \leq \xi_k \leq 1$ for $k = 0, 1, 2$ and $\sum_{k=0}^{2} \xi_k = 1$. Then, taking into account these properties the affine mapping can be written as

$$\mathbf{f}(\xi_1, \xi_2) \equiv \mathbf{f}(\xi_0, \xi_1, \xi_2) = \sum_{k=0}^{2} \xi_k \mathbf{x_k} \overset{\xi_0+\xi_1+\xi_2=1}{=} (1 - \xi_1 - \xi_2)\mathbf{x_0} + \xi_1 \mathbf{x_1} + \xi_2 \mathbf{x_2}. \quad (2.1)$$

This mapping can also be written in matrix form as

$$\begin{array}{rccl} \mathbf{f}: & t_R & \longrightarrow & t \\ & \mathbf{u} & \longrightarrow & \mathbf{x} = \mathbf{A_0 u} + \mathbf{x_0}, \end{array} \quad (2.2)$$

where

$$\mathbf{A_0} = \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix},$$

and $\mathbf{x} = (x, y)^t$, $\mathbf{u} = (\xi_1, \xi_2)^t$.

Note that this application maps $\mathbf{u_0}$, $\mathbf{u_1}$ and $\mathbf{u_2}$ in the logical space onto $\mathbf{x_0}$, $\mathbf{x_1}$ and $\mathbf{x_2}$ in the physical space. The vector $\mathbf{x_0}$ controls the translation of the element, and the matrix $\mathbf{A_0}$ controls its area, shape and orientation. Matrix $\mathbf{A_0}$ is called the *Jacobian matrix*, because it is indeed the Jacobian of the affine mapping with respect to $\{\xi_k\}_{k=0,1,2}$.

We denote by *ideal element*, t_I, the element that represents the desired shape to achieve. Therefore, to measure the deviation from the ideal triangle, t_I, of any triangle t in the physical space, we want to find an affine mapping, $\mathbf{f_S}$, such that $\mathbf{f_S} : t_I \longrightarrow t$ (see Figure 2.2).

According to equation (2.2) it is really simple to go from the reference triangle in the logical coordinates to any other triangle in the physical space. Therefore, we use this idea to determine $\mathbf{f_S}$ by the composition of two functions.

To this end we first define the affine mapping between the reference element, t_R, and the ideal one, t_I. If $\tilde{\mathbf{x}}_\mathbf{k}$ are the coordinates of the ideal element, this affine

mapping can be written as (see Figure 2.2)

$$\begin{array}{rccl} \mathbf{f_W}: & t_R & \longrightarrow & t_I \\ & \mathbf{u} & \longrightarrow & \mathbf{x} = \mathbf{W}\mathbf{u} + \tilde{\mathbf{x}}_\mathbf{0}, \end{array}$$

where

$$\mathbf{W} = \begin{pmatrix} \tilde{x}_1 - \tilde{x}_0 & \tilde{x}_2 - \tilde{x}_0 \\ \tilde{y}_1 - \tilde{y}_0 & \tilde{y}_2 - \tilde{y}_0 \end{pmatrix}.$$

Similarly we define the affine mapping $\mathbf{f_A}$, that maps the reference element t_R to the physical triangle t, with coordinates $\mathbf{x_k},\ k = 0, 1, 2$, as (see Figure 2.2)

$$\begin{array}{rccl} \mathbf{f_A}: & t_R & \longrightarrow & t \\ & \mathbf{u} & \longrightarrow & \mathbf{x} = \mathbf{A}\mathbf{u} + \mathbf{x}_\mathbf{0}, \end{array}$$

where

$$\mathbf{A} = \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix},$$

Thus the desired affine mapping $\mathbf{f_S}$ that maps the ideal element onto the physical triangle can be defined as:

$$\begin{array}{rccccc} \mathbf{f_S} = \mathbf{f_A} \circ \mathbf{f_W^{-1}}: & t_I & \xrightarrow{\mathbf{f_W^{-1}}} & t_R & \xrightarrow{\mathbf{f_A}} & t \\ & \tilde{\mathbf{x}} & \longrightarrow & \mathbf{u} = \mathbf{f_W^{-1}}(\tilde{\mathbf{x}}) & \longrightarrow & \mathbf{x} = \mathbf{f_A}(\mathbf{u}). \end{array} \tag{2.3}$$

Therefore the analytical expression of affine mapping $\mathbf{f_S}$ is

$$\mathbf{f_S}(\tilde{\mathbf{x}}) \;=\; \mathbf{f_A}(\mathbf{f_W^{-1}}(\tilde{\mathbf{x}})) = \mathbf{A}\mathbf{f_W^{-1}}(\tilde{\mathbf{x}}) + \mathbf{x_0} = \mathbf{A}(\mathbf{W^{-1}}\tilde{\mathbf{x}} + \tilde{\mathbf{v}}) + \mathbf{x_0} = \mathbf{A}\mathbf{W^{-1}}\tilde{\mathbf{x}} + \mathbf{v},$$

for a given translation vector $\mathbf{v} = \mathbf{A}\tilde{\mathbf{v}} + \mathbf{x_0}$.

The Jacobian matrix of this application

$$\mathbf{S} = \mathbf{A}\mathbf{W^{-1}} \tag{2.4}$$

is called *shape matrix* in references [Knupp 03] and [Escobar 03].

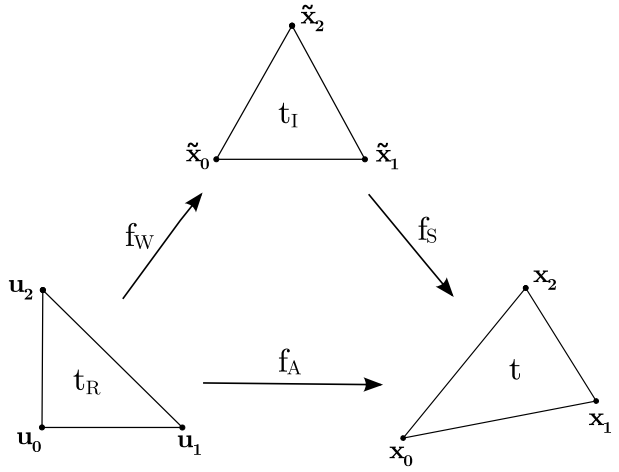

Figure 2.2: Affine mappings for triangular elements.

[Knupp 01] proves that the affine mapping $\mathbf{f_S}$ defined in (2.3) does not depend on the node that has been choosen as translation vector or on the numbering of the nodes.

Note that we can select different ideal elements in order to generate elements with different geometric properties. The following examples will illustrate this idea.

- For isotropic triangular meshes we choose as ideal element the equilateral triangle. Taking $\tilde{\mathbf{x}}_\mathbf{0} = (0,0)^t$, $\tilde{\mathbf{x}}_\mathbf{1} = (1,0)^t$ and $\tilde{\mathbf{x}}_\mathbf{2} = (\frac{1}{2}, \frac{\sqrt{3}}{2})^t$ as the coordinates of the equilateral triangle, the resulting Jacobian matrix is

$$\mathbf{W} = \begin{pmatrix} 1 & \frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} \end{pmatrix}. \tag{2.5}$$

- For quadrilateral meshes, each element, delimited by the nodes $\{\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}\}$, is divided into four triangles $\{\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}\}$, $\{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}\}$, $\{\mathbf{x_2}, \mathbf{x_3}, \mathbf{x_0}\}$ and $\{\mathbf{x_3}, \mathbf{x_0}, \mathbf{x_1}\}$ (see Figure 2.3). The quality of the quadrilateral is a weighting of the quality of this four triangles (see references [Knupp 03] and [Knupp 03b]). The ideal quadrilateral element is the square. Thus, if we subdivide it this way into four triangles, we get four triangles that are indeed rectangle and isosceles. Therefore, in this case the triangle that represents the geometric property that we want to achieve is not the equilateral but the rectangle isosceles. Then, if we choose $\tilde{\mathbf{x}}_\mathbf{0} = (0,0)$, $\tilde{\mathbf{x}}_\mathbf{1} = (1,0)$ and $\tilde{\mathbf{x}}_\mathbf{2} = (0,1)$, the Jacobian matrix becomes the identity matrix

$$\mathbf{W} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \tag{2.6}$$

because the logical coordinates are indeed those of the ideal triangle.

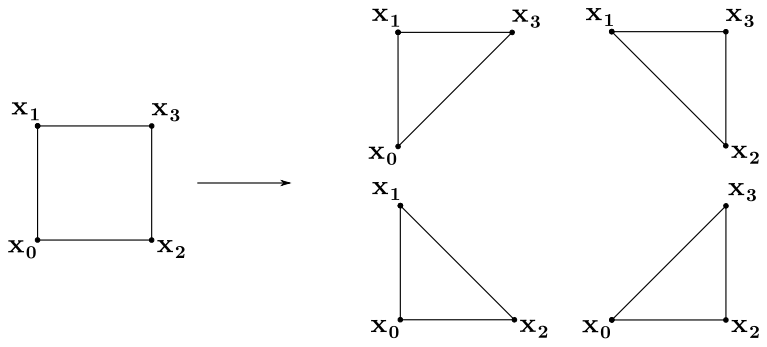

Figure 2.3: Division of a square to compute its objective function

2.3 Planar quality measure for triangular elements

The shape matrix $\mathbf{S}$ defined in (2.4) contains information about how much we have "distorted" the ideal element to become the physical one. In this section we will

analyse a quality metric for planar triangular meshes based on $\mathbf{S}$. Later, this planar measure will be extended to surface problems.

In this work we will use the quality metric, introduced by [Knupp 01],

$$q(\mathbf{S}) = \frac{2\sigma(\mathbf{S})}{|\mathbf{S}|^2}, \tag{2.7}$$

where $\sigma(\mathbf{S}) = \det(\mathbf{S})$ is the determinant of $\mathbf{S}$, and $|\mathbf{S}| = \sqrt{(\mathbf{S},\mathbf{S})} = \sqrt{\mathrm{tr}(\mathbf{S}^t\mathbf{S})}$ is its Frobenius norm[2].

The shape metric (2.7) behaves as we have previously commented. It reaches a maximum value of 1 for the ideal element, and a minimum value 0 for degenerated elements.

From this quality metric, also called *shape metric* in [Knupp 01], we are going to extract the objective function. Recall that we want to improve the quality of the mesh by means of a minimizing problem. Since the quality metric takes its maximum value for the ideal element and its minimum for the degenerated case, the objective function is defined as (see [Escobar 03] and [Escobar 06])

$$\eta(\mathbf{S}) = \frac{1}{q(\mathbf{S})} = \frac{|\mathbf{S}|^2}{2\sigma(\mathbf{S})}. \tag{2.8}$$

The image of this function is the interval $[1, \infty]$, achieving ∞ only when the physical element is degenerated, and 1 when it becomes the ideal triangle.

To illustrate the behavior of the quality metric $q(\mathbf{S})$, defined in (2.7), and the objective function $\eta(\mathbf{S})$, defined in (2.8), we introduce the following example. Consider a triangular element with two fixed nodes, $\mathbf{x_0} = (0, 0.5)$ and $\mathbf{x_1} = (0, -0.5)$, and a third node $\mathbf{x_2}(x) = (x, 0)$ that we move in the x-axis, see Figure 2.4.

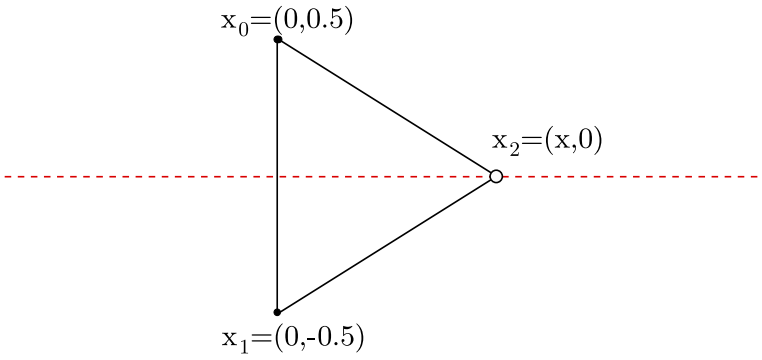

Figure 2.4: Triangle moving the node $\mathbf{x_2}$.

In Figures 2.5 and 2.6 the shape metric and the objective function are displayed when node $\mathbf{x_2}$ moves from $x = -5$ to $x = 5$. Note that we have selected the equilateral triangle as ideal element.

[2]We use the notation $(\mathbf{A}, \mathbf{B}) = \mathrm{tr}(\mathbf{A}^t\mathbf{B})$.

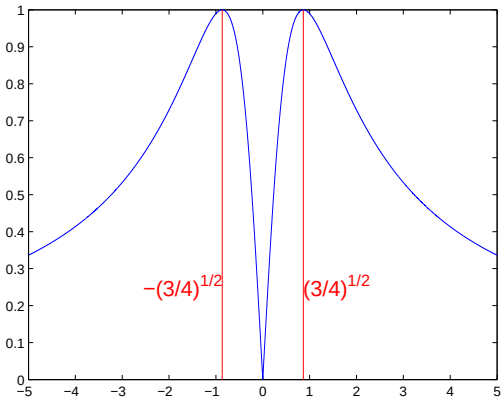

Figure 2.5: Quality index moving node $\mathbf{x_2}$: $q(\mathbf{S}(\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}(x)))$, $x \in [-5, 5]$.

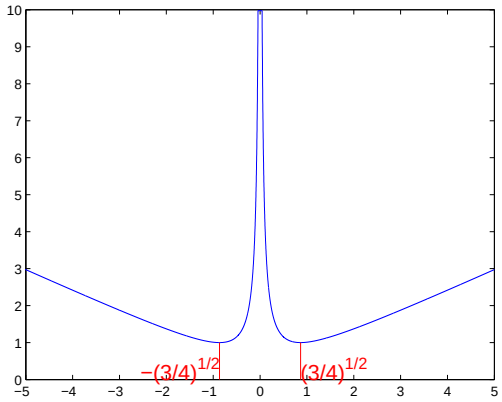

Figure 2.6: Objective function moving $\mathbf{x_2}$: $\eta(\mathbf{S}(\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}(x)))$, $x \in [-5, 5]$.

Figure 2.5 shows that there is just a point with quality zero. This is the point at which the element achieves its degenerated configuration. Note that the quality metric tends to zero when $x \to \pm\infty$, because the limit can also be considered as a degenerated position.

Figure 2.5 also shows that the quality metric has two maximums, achieved for $x = \pm\sqrt{3}/2$. In these two points, the value of the measure is 1, because the equilateral (ideal) configuration is achieved. However, when we have the triangle in a mesh, it inherits an order in the nodes from the connectivity matrix. Lets consider that the analysed triangle inherits the nodal order $\{\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}\}$. Then the only valid position to place the nodes is in the right side domain ($x = \sqrt{3}/2$). The other maximum ($x = -\sqrt{3}/2$) leads to an inverted configuration that is not acceptable in a mesh.

Figure 2.6 shows that the optimal locations for node $\mathbf{x_2}$ are at the two minimums. Moreover, an asymptote has appeared at $x = 0$, due to the achievement of the degenerated configuration ($\sigma(\mathbf{S}) = 0$).

Note that function (2.8) is not able to untangle elements because it only measures the shape of the elements, but it does not consider their orientation. Thus it can not distinguish the correct position among the two minimus. These difficulties will be overcome on Section 2.5.

2.4 The objective function for a mesh

Suppose that we have a given mesh M, instead of just an element as we were doing in Section 2.3. To improve the quality of all the elements of the mesh (smooth the mesh) we will modify the location of the inner nodes. Thus, all boundary nodes will be fixed. Let $\mathcal{V}$ be the set of inner nodes and let v be a given node $v \in \mathcal{V}$.

Given a node $v \in \mathcal{V}$ we define the *local submesh* associated to it, $N(v)$, as the set of elements that contain node v. Figure 2.7 shows the local mesh associated to

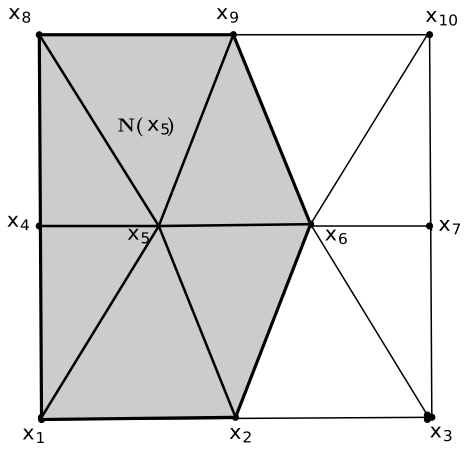

Figure 2.7: Triangular mesh with 2 inner nodes. Marked in grey the local submesh $N(v)$, for $v \equiv \mathbf{x_5}$.

the inner node $\mathbf{x_5}$. Note that the coordinates of the nodes of the mesh in Figure 2.7 are:

$$\begin{array}{ll} \mathbf{x_1} = (0,0) & \mathbf{x_2} = (0.5,0) \\ \mathbf{x_3} = (1,0) & \mathbf{x_4} = (0,0.5) \\ \mathbf{x_5} = (\frac{1}{3},0.5) & \mathbf{x_6} = (\frac{1}{6},0.5) \\ \mathbf{x_7} = (1,0.5) & \mathbf{x_8} = (1,0) \\ \mathbf{x_9} = (1,0.5) & \mathbf{x_{10}} = (1,1) \end{array}$$

The objective function on node v will be computed as a weighting of the contribution of all the elements that belong to its local mesh.

Let $\mathbf{x} = (x, y)$ be the coordinates of the inner node v, and let $N(v)$ be the associated local submesh. Assume that $N(v)$ is composed by m elements (triangles in our case).

Let $\mathbf{S_k}$ be the Jacobian matrix of the kth triangle of $N(v)$. Then, according to [Knupp 03b] we write the objective function on the kth triangle as

$$\eta_k(\mathbf{x}) = \frac{|\mathbf{S_k}(\mathbf{x})|^2}{2\sigma(\mathbf{S_k}(\mathbf{x}))}. \tag{2.9}$$

We define the objective function on node v as the p-norm[3] of the objective function of all the triangles of its local submesh:

$$|K_\eta|_p(\mathbf{x}) = \left(\sum_{k=1}^{m} (\eta_k)^p (\mathbf{x}) \right)^{1/p}. \tag{2.10}$$

In this context, we define the feasible region as the set of points where the free node can be located to get a valid mesh. Concretely, the feasible region is the

[3]In this project we consider the 2-norm, p=2.

interior of the poligonal set $\mathcal{H} = \cap_{k=1}^{k=m} H_k$ where H_k are the half-planes defined by $\sigma_k(\mathbf{x}) \geq 0$. We say that a triangle of the mesh is *inverted* if $\sigma(\mathbf{x}) < 0$, and *degenerated* if $\sigma(\mathbf{x}) = 0$.

Figure 2.8 presents the surface and the contour plots of the objective function corresponding to node 5 of the mesh presented in Figure 2.7.

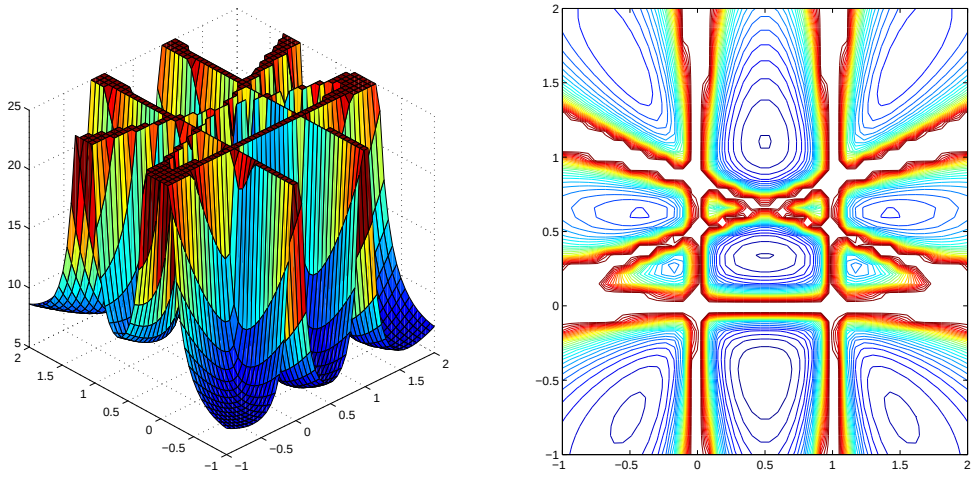

Figure 2.8: Objective function of node 5 of the mesh of Figure 2.7, with a fixed treshold equal to 25.

The objective function (2.10) has several asymptotes at the boundary of the feasible region, where $\sigma = 0$. This avoids the optimization algorithm creating a tangled mesh when it starts from a valid position of node $\mathbf{x_5}$. However, these asymptotes do not allow the optimization algorithm to reach a valid position when it starts from a tangled one ($\mathbf{x_5}$ initially placed in a position that defines a tangled triangle).

Moreover the objective function also presents local minimums outside the feasible region. Then, the otpimization algorithm, that is indeed a minimization, will find a minimum that is not the optimal position. Furthermore this local minimum will generate a tangled triangle.

2.5 Modified objective function: a function for untangling and smoothing elements

In this section we are going to introduce a modification on the objective function developed in [Escobar 03] in order to avoid the asymptotes and the false local minimum that appear using the original objective function (2.10).

Figure 2.8 shows that despite the fact that the objective function is smooth in the

region that defines a valid local mesh $N(v)$, it has discontinuities at the boundary. This happens due to the fact that when σ_k tends to zero, η_k tends to infinity.

If we want to smooth the mesh and the minimizing node is inside the feasible region, function (2.10) behaves well, because we are in the smooth region and we can find the global minimum. But if there exist tangled elements, function (2.10) can not be used due to the existence of these asymptotes.

For this purpose and according to [Escobar 03], we are going to modify the objective function (2.10) in order to obtain a new one that is smooth all over $\mathbb{R}^2$. As stated in [Escobar 03], the new objective function will achieve its minimum near to the one of the original function.

The modification that is going to be applied consists on replacing σ in (2.9) by

$$h(\sigma) = \frac{1}{2}\left(\sigma + \sqrt{\sigma^2 + 4\delta^2}\right), \tag{2.11}$$

where δ is an arbitrary parameter that is chosen depending on the problem (see [Escobar 03] for further details). Note that function (2.11) is a positive increasing function that verifies $h(0) = \delta$. Figure 2.9 presents a plot of $h(\sigma)$ versus σ that illustrates the properties of $h(\sigma)$.

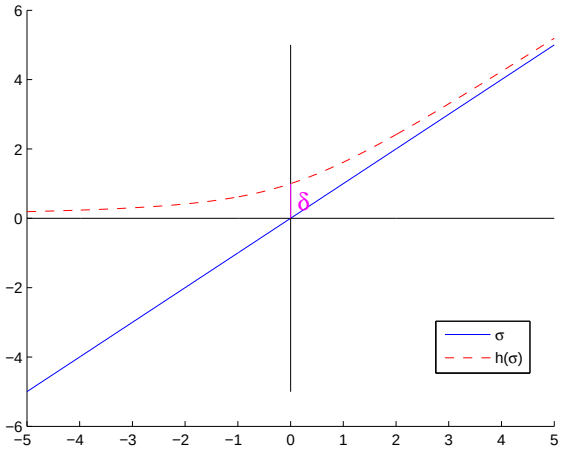

Figure 2.9: $h(\sigma)$ vs σ

Note that $\lim_{\delta\to 0} h(\sigma) = \sigma$, $\forall\sigma \geq 0$, and $\lim_{\delta\to 0} h(\sigma) = 0$, $\forall\sigma \leq 0$. Then, on the one hand, for small values of δ when the free node is in the feasible region, the modified function is similar to the original one. On the other hand, if it is in a tangled position, as it gets further from the feasible region ($\sigma \to \pm\infty$), the function never loses its smoothness but has limit equal to infinity ($\eta \overset{h(\sigma)\to 0}{\longrightarrow} \infty$). Further details on the behaviour of $h(\sigma)$ and on the selection of the value of δ can be found in [Escobar 03].

Taking into account this modification, see equation (2.11), the new objective function for the local mesh is

$$|K_\eta^*|(\mathbf{x}) = \left(\sum_{k=1}^{m} (\eta_k^*)^p(\mathbf{x}) \right)^{1/p}, \tag{2.12}$$

where

$$\eta_k^* = \frac{|\mathbf{S_k}|}{2h(\sigma_k)}. \tag{2.13}$$

Figure 2.10 presents the plots of the new objective function, equation (2.12), for the node $\mathbf{x_5}$ of the mesh presented in Figure 2.7.

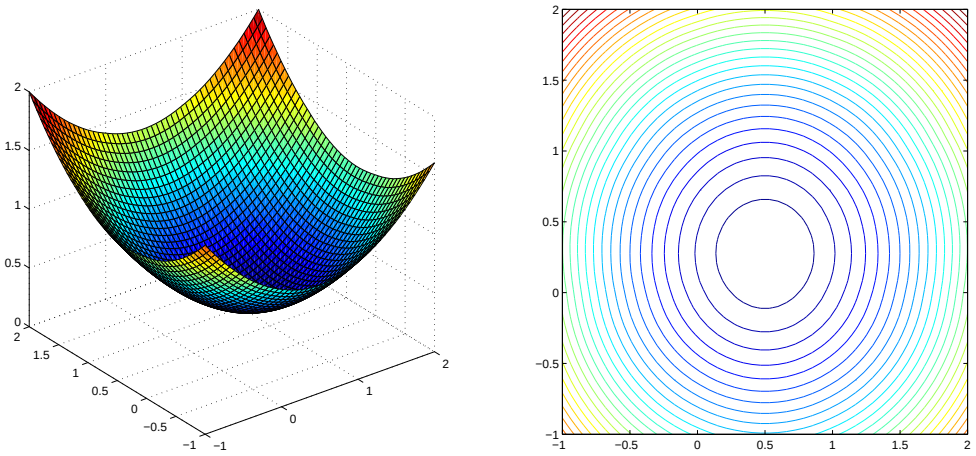

Figure 2.10: Modified objective function of node 5 of the mesh of Figure 2.7.

Note that this new function has no asymptotes and is smooth all over the domain. The minimum of the function is approximately $\left(\frac{1}{3}, \frac{1}{2}\right)$, that is the same that the original function had in the feasible zone. Then, using this function instead of the original (2.8) will let us untangle as well as smooth the mesh.

3 Algebraic quality measure for surfaces

3.1 Surface quality measure for triangular elements

As it has been stated in Section 1, the objective of this work is to develop an algorithm to smooth and untangle triangular surface meshes. The work presented in this section is based on the work developed in [Escobar 06]. We deduce the algorithm again for completness.

Let Σ be a surface, and $M(p)$ a local submesh associated to node p placed on Σ. Suppose that it is posible to find a projection plane P such that the orthogonal projection of $M(p)$ on P is a valid mesh $N(q)$, where q is the orthogonal projection of p on P. We denote by $N(q)$ the *parametric mesh*, where $q = \mathbf{f_\Pi}(p)$, being $\mathbf{f_\Pi}$ the orthogonal projection from the surface to P, see Figure (3.1).

Moreover, suppose that we choose the coordinate system such that plane $z = 0$ coincides with P (in Section 3.3 we will extend this work to a general plane P). Finally assume that there exists a parametrization of Σ

$$\begin{array}{rccl} \mathbf{s}: & P & \longrightarrow & \Sigma \\ & (x,y) & \longrightarrow & \mathbf{s}(x,y) = (x,y,f(x,y)), \end{array} \tag{3.1}$$

with f a continous function.

We want to smooth and untangle the mesh on the surface. To do so, we aim to find the position $\tilde{q}$ in the feasible region of $N(q)$ such that when we map this position to the surface mesh $M(p)$, the quality of the surface mesh is optimum.

We want to work on the parametric mesh in order to modify the surface mesh. Then, we will define as ideal elements in $N(q)$ those that become equilateral on $M(p)$. From this idea we are going to design the whole theory for the surface smoother.

Let $\tau \in M(p)$ be a triangular element on Σ, and let $t \in N(q)$ be its projection on P, $t = \mathbf{f_\Pi}(\tau)$. Since we have selected the plane P as $\{z = 0\}$, the orthogonal projection can be written as

$$\mathbf{f_\Pi}(\mathbf{y}) = \Pi\mathbf{y} = (\mathbf{e_1}, \mathbf{e_2})^T\mathbf{y}, \tag{3.2}$$

where $\{\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}\}$ is the canonical basis in $\mathbb{R}^3$ and $\Pi = (\mathbf{e_1}, \mathbf{e_2})^t$. Then, if $\mathbf{y_k} = (x_k, y_k, z_k)$, $k = 0, 1, 2$, are the vertices of triangle τ, we can express the vertices of triangle t as $\mathbf{x_k} = (x_k, y_k) = \mathbf{f_\Pi}(\mathbf{y_k})$, $k = 0, 1, 2$.

We can define the affine mapping that maps the reference triangle, t_R, to triangle τ on the surface mesh as

$$\mathbf{f_{A_\pi}}: \quad t_R \longrightarrow \tau \tag{3.3}$$

$$\mathbf{u} \longrightarrow \mathbf{y} = \mathbf{A}_\pi\mathbf{u} + \mathbf{y_0}, \tag{3.4}$$

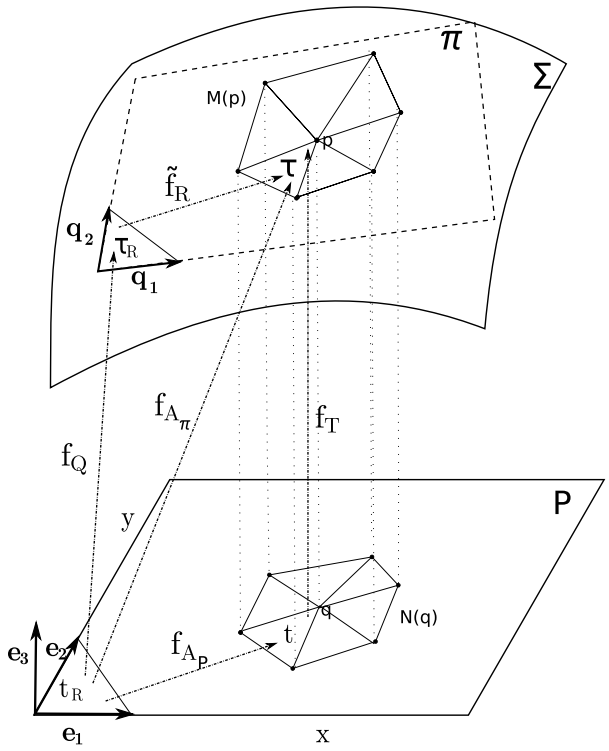

Figure 3.1: Applications between the local surface mesh $M(p)$ and the associated parametric mesh $N(q)$.

where $\mathbf{A}_\pi$ is the Jacobian matrix,

$$\mathbf{A}_\pi = (\mathbf{y_1} - \mathbf{y_0} \quad \mathbf{y_2} - \mathbf{y_0}) = \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \\ z_1 - z_0 & z_2 - z_0 \end{pmatrix}, \tag{3.5}$$

and $\mathbf{y_0}$ is choosen as translation vector.

Then, we can define the affine mapping that maps the reference triangle to the projected physical triangle on P as

$$\mathbf{f_{A_P}}: \quad t_R \longrightarrow \quad t \tag{3.6}$$

$$\mathbf{u} \longrightarrow \quad \mathbf{x} = \mathbf{A_P u} + \mathbf{x_0}, \tag{3.7}$$

where $\mathbf{A_P} = \Pi \mathbf{A}_\pi$ is the Jacobian matrix,

$$\mathbf{A_P} = (\mathbf{x_1} - \mathbf{x_0} \quad \mathbf{x_2} - \mathbf{x_0}) = \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix}, \tag{3.8}$$

and $\mathbf{x_0} = \mathbf{f_\Pi}(\mathbf{y_0})$ is the translation vector.

Afterwards we define the affine mapping $\mathbf{f_T} : t \longrightarrow \tau$ as $\mathbf{f_T} = \mathbf{f_{A_\pi}} \circ \mathbf{f_{A_P}^{-1}}$, with the 3×2 Jacobian matrix $\mathbf{T}$,

$$\mathbf{T} = \mathbf{A}_\pi \mathbf{A_P^{-1}}. \tag{3.9}$$

Let π be the plane that contains $\tau \subset \Sigma$. We want to move q on P in order to find the ideal triangle $t_I \subset P$. As we have previously stated, the ideal element on P is defined as the triangle that is mapped by $\mathbf{f_T}$ into an equilateral triangle $\tau_E \subset \pi$. It is important to point out that each element of $M(p)$ defines a different t_I because π is not the same for all the triangles on Σ.

Note that $\mathbf{f}_{\mathbf{A}_\pi}$ maps $\mathbf{u_0} = (0,0)$, $\mathbf{u_1} = (1,0)$ and $\mathbf{u_2} = (0,1)$ on P to the nodes of triangle τ. If we call V_π the subspace spanned by the columns of $\mathbf{A}_\pi$, π can be defined as the plane spanned by V_π at the point $\mathbf{y_0}$.

Now we decompose $\mathbf{A}_\pi$ by a $\mathbf{QR}$ factorization that ensures unicity:

$$\mathbf{A}_\pi = \mathbf{QR}, \tag{3.10}$$

where $\mathbf{Q}$ is a 3×2 orthogonal matrix (with unitary columns) and $\mathbf{R}$ is a 2×2 upper triangular matrix with $R_{11}, R_{22} > 0$.

The columns of $\mathbf{Q}$ define then an orthonormal basis $\{\mathbf{q_1}, \mathbf{q_2}\}$ that spans V_π. In this sense, $\mathbf{Q}$ maps the 2D basis of $P \subset \mathbb{R}^2$ into the 2D basis of $\pi \subset \mathbb{R}^3$. We can define an affine mapping $\mathbf{f_Q} : t_R \longrightarrow \tau_R$ such that $\mathbf{Q}$ is its Jacobian matrix. In addition, we can consider an affine mapping $\tilde{\mathbf{f}}_\mathbf{R} : \tau_R \longrightarrow \tau$, defined on plane π (expressed on basis $\{\mathbf{q_1}, \mathbf{q_2}\}$), such that $\mathbf{R}$ is its Jacobian matrix.

In Appendix A.1 it is proved that $\mathbf{R}$ is indeed the Jacobian matrix of the affine mapping that maps τ_R into τ, and we can therefore write this mapping as $\tilde{\mathbf{f}}_\mathbf{R} : \tau_R \longrightarrow \tau$. In Appendix A.1 we also detail further analysis of the different affine mappings that we have derived from $\mathbf{f}_{\mathbf{A}_\pi}$.

From now on we will focus on the Jacobian matrices of the affine mappings and we will work out some matrix identities. Actually, the Jacobian matrices of the applications are what is truly interesting in order to develope for the optimization surface mesh algorithm.

It is important to point out that the same matrix can be the Jacobian matrix of different affine mappings. To illustrate this fact we notice that:

- Matrix $\mathbf{Q}$ has orthonormal columns and then it mantains angles and distances. Thus, it can be used to deffine the affine mappings

$$\begin{aligned} \mathbf{f}^\mathbf{1}_\mathbf{Q} &: t_R \longrightarrow \tau_R, \\ \mathbf{f}^\mathbf{2}_\mathbf{Q} &: t_E \longrightarrow \tau_E. \end{aligned} \tag{3.11}$$

  Note that these two affine mappings are indeed identical, but we distinguish them in order to emphasize that we will use matrix $\mathbf{Q}$ to send t_R and t_E to τ_R and τ_E, respectively.

- In the local basis of planes P and π, t_R and τ_R have respectively the same expressions. Moreover, the same happens between $t_E \subset P$ and $\tau_E \subset \pi$. Thus,

the Jacobian matrix $\mathbf{W_E}$ can be used to define the following affine mappings

$$\begin{aligned} \mathbf{f_{W_E}} &: t_R \longrightarrow t_E, \\ \tilde{\mathbf{f}}_{\mathbf{W_E}} &: \tau_R \longrightarrow \tau_E. \end{aligned} \tag{3.12}$$

Note that these two affine mappings have identical expressions, but they are expressed in different basis.

The target of this work is to optimize the surface mesh using equations (2.12) and (2.13). Thus, we have to find an expression for the shape matrix $\mathbf{S}$, that is defined on π. Our aim is to express $\mathbf{S}$ in terms of other matrices defined on P. The idea is then to have a key to move the nodes on P in order to improve the quality on π.

Consider the affine mapping $\mathbf{f_{QW_E}} : t_R \longrightarrow \tau_E$ ($\mathbf{f_{QW_E}} = \hat{\mathbf{f}}_{\mathbf{Q}} \circ \tilde{\mathbf{f}}_{\mathbf{W_E}}$), with $\mathbf{QW_E}$ its 3×2 Jacobian matrix. According to equation (3.10), $\mathbf{Q} = \mathbf{A}_\pi \mathbf{R^{-1}}$. Then we obtain the following identity for the Jacobian matrix

$$\mathbf{QW_E} = \mathbf{A}_\pi \mathbf{R^{-1}} \mathbf{W_E}, \tag{3.13}$$

that will be used later.

On plane π we define the affine mapping

$$\mathbf{f_S} : \tau_E \xrightarrow{\mathbf{f^{-1}_{W_E}}} \tau_R \xrightarrow{\mathbf{f_R}} \tau, \tag{3.14}$$

with

$$\mathbf{S} = \mathbf{R} \mathbf{W_E^{-1}} \tag{3.15}$$

the 2×2 Jacobian matrix of $\mathbf{f_S}$.

We choose the ideal triangle in π as the equilateral, $\tau_I = \tau_E$, and then we compute the Jacobian matrix $\mathbf{W_I}$ of the affine mapping $\mathbf{f_{W_I}} : t_R \longrightarrow t_I$ by imposing the condition

$$\mathbf{TW_I} = \mathbf{QW_E}. \tag{3.16}$$

Since $\mathbf{f_{TW_I}} : t_R \longrightarrow \tau_I$ and $\mathbf{f_{TW_E}} : t_R \longrightarrow \tau_E$, we are actually imposing to the affine mapping $\mathbf{f_{TW_I}}$ to be the exact same mapping than $\mathbf{f_{TW_E}}$.

Using the identity (3.13), equation (3.16) becomes

$$\mathbf{TW_I} = \mathbf{A}_\pi \mathbf{R^{-1}} \mathbf{W_E}, \tag{3.17}$$

and taking into account equation (3.9),

$$\mathbf{W_I} = \mathbf{A_P} \mathbf{R^{-1}} \mathbf{W_E}. \tag{3.18}$$

Now we define on P the affine map $\mathbf{f_{S_I}} : t_I \longrightarrow t$ ($\mathbf{f_{S_I}} = \mathbf{f_{A_P}} \circ \mathbf{f_{W_I^{-1}}}$), with Jacobian matrix

$$\mathbf{S_I} = \mathbf{A_P} \mathbf{W_I^{-1}}. \tag{3.19}$$

Using the matrix identities (3.18) and (3.15) in equation (3.19),

$$\begin{aligned} \mathbf{S_I} &\overset{(3.18)}{=} \mathbf{A_P W_E^{-1} R A_P^{-1}} \overset{(3.15)}{=} \mathbf{A_P W_E^{-1} S W_E A_P^{-1}} \\ &= \mathbf{A_P W_E^{-1} S (A_P W_E^{-1})^{-1} = S_E S S_E^{-1}}, \end{aligned} \tag{3.20}$$

where

$$\mathbf{S_E} = \mathbf{A_P W_E^{-1}} \tag{3.21}$$

is the Jacobian matrix of the affine map $\mathbf{f_{S_E}} : t_E \longrightarrow t$. To end with, using the identity (3.20) it follows that:

$$\mathbf{S} = \mathbf{S_E^{-1} S_I S_E}. \tag{3.22}$$

This final matrix identity points out that $\mathbf{S}$ and $\mathbf{S_I}$ are similar. Then we have been able to design a relation between the matrix of the affine mapping $\mathbf{f_S}$ defined on π, matrix $\mathbf{S}$, and the matrices of the affine mappings defined on P, $\mathbf{S_E}$ and $\mathbf{S_I}$.

Note that equation (3.22) is not a functional identity. It is a relation between the Jacobian matrices of different applications, and we must not identify it with a compositon of applications.

3.2 Objective function: a 2D function for a 3D problem

Expression (3.22) can be used together with (2.12) to construct the objective function. Thus, the minimization of (2.12) can be used to improve the quality of the triangle surface mesh. Though this is not that easy. On the one hand, despite being a 2×2 matrix, $\mathbf{S}$ depends on (x, y, z). On the other hand, each triangle of the local mesh defines a different plane π. Then, the optimization of $M(p)$ by an objective function defined directly from (2.12) using (3.22) would require to impose the constraint $p \in M(p)$, and this could be expensive from a computational point of view.

In order to avoid the problems derived from the fact that $\Sigma \subset \mathbb{R}^3$ we are going to carry out the minimization problem not in $M(p)$ but in $N(q)$. We will approximate (3.22) in order to restrict it to the two variables x and y of the plane P, and then working on P we will optimize the node q and finally we will update p.

Suppose that surface Σ is parametrized by equation (3.1). Consider that $\mathbf{x} = (x, y)^t$ is the position of the free node q. Then, the position of p is $\mathbf{y} = \mathbf{s}(\mathbf{x})^t = (\mathbf{x}, f(\mathbf{x}))^t = (x, y, f(x, y))^t$.

We are going to analyse one by one the matrices involved in the definition of $\mathbf{S}$, equation (2.4), and we are going to mantain constant in each step of the minimizing process those matrices that depend on z. Under this approach $\mathbf{S}(x, y, z)$ will become $\mathbf{S^0}(x, y)$, where $\mathbf{S^0}$ will be a two variable approximation of $\mathbf{S}$. Then, we will be able to carry out the optimization in P, updating in each step $\mathbf{S^0}$ taking into account the new position of the node, $\mathbf{y} = (x, y, z)$.

On the one hand, note that according to equation (3.21), $\mathbf{S_E} = \mathbf{A_P}\mathbf{W_E^{-1}}$, does not depend on $\mathbf{y}$. That is, $\mathbf{W_E}$ is a constant matrix that only depends on the coordinates of the equilateral triangle t_E, and $\mathbf{A_P} \equiv \mathbf{A_P}(\mathbf{x})$ is defined on P. Then,

$$\mathbf{S_E}(\mathbf{x}) = \mathbf{A_P}(\mathbf{x})\mathbf{W_E^{-1}}. \tag{3.23}$$

On the other hand, $\mathbf{S_I} = \mathbf{A_P}\mathbf{W_I^{-1}}$ does depend on $\mathbf{y}$. From (3.18), $\mathbf{W_I} = \mathbf{A_P}\mathbf{R^{-1}}\mathbf{W_E}$, and despite $\mathbf{A_P} \equiv \mathbf{A_P}(\mathbf{x})$ and $\mathbf{W_E}$ is constant, $\mathbf{R}$ is a function of $\mathbf{y}$ because it is derived from the $\mathbf{QR}$ factorization of $\mathbf{A}_\pi(\mathbf{y})$, see equation (3.10).

Then, in the iterative procedure we will optimize the local mesh $M(p)$ mantaining constant $\mathbf{W_I}(\mathbf{y})$ in each step. At the begining of each step, we fix $\mathbf{W_I}(\mathbf{y})$, as $\mathbf{W_I^0} = \mathbf{W_I}(\mathbf{y_0})$, where $\mathbf{y_0}$ is the initial position of p in this step.

Defining $\mathbf{S_I^0}(\mathbf{x})$ as

$$\mathbf{S_I^0}(\mathbf{x}) = \mathbf{A_P}(\mathbf{x})(\mathbf{W_I^0})^{-1}, \tag{3.24}$$

expression (3.22) can be approximed by

$$\mathbf{S^0}(\mathbf{x}) = \mathbf{S_E^{-1}}(\mathbf{x})\mathbf{S_I^0}(\mathbf{x})\mathbf{S_E}(\mathbf{x}). \tag{3.25}$$

This expression can be simplified taking into account equations (3.23), (3.24) and (3.25):

$$\begin{aligned}\mathbf{S^0}(\mathbf{x}) &= \mathbf{S_E^{-1}}(\mathbf{x})\mathbf{S_I^0}(\mathbf{x})\mathbf{S_E}(\mathbf{x}) \overset{(3.23),(3.24)}{=} \mathbf{W_E}\cancel{\mathbf{A_P^{-1}}(\mathbf{x})}\cancel{\mathbf{A_P}(\mathbf{x})}(\mathbf{W_I^0})^{-1}(\mathbf{x})\mathbf{S_E}(\mathbf{x}) \\ &\overset{(3.18)}{=} \cancel{\mathbf{W_E}\mathbf{W_E^{-1}}}\mathbf{R^0}(\mathbf{A_P^0})^{-1}\mathbf{S_E}(\mathbf{x}) = \mathbf{R^0}(\mathbf{A_P^0})^{-1}\mathbf{S_E}(\mathbf{x}).\end{aligned}$$

Therefore,

$$\mathbf{S^0}(\mathbf{x}) = \mathbf{R^0}(\mathbf{A_P^0})^{-1}\mathbf{S_E}(\mathbf{x}). \tag{3.26}$$

Equation (3.26) stablishes the new shape matrix. This shape matrix will be used to construct the objective function[4] as we have already done before in (2.12) and (2.13). For a given triangle in the local mesh, $\tau \subset \pi$, $\eta_k^0(\mathbf{x})$ will be computed as

$$\eta_k^0(\mathbf{x}) = \frac{|\mathbf{S_k^0}(\mathbf{x})|^2}{2h(\sigma_k^0(\mathbf{x}))}, \tag{3.27}$$

where $\sigma_k^0(\mathbf{x}) = \det(\mathbf{S_k^0}(\mathbf{x}))$. Then, the objective function will be

$$|K_\eta^0|_p(\mathbf{x}) = \left(\sum_{k=1}^{m} \left(\eta_k^0\right)^p(\mathbf{x})\right)^{1/p}. \tag{3.28}$$

[4]Note that for the final smoothing algorithm we are going to use the modified objective function (2.12) instead of the original one (2.10). We want a function able to untangle as well as smooth surface meshes.

3.3 Optimal projection plane

In Sections 3.1 and 3.2 we have assumed that the triangular surface mesh is projected into the $z = 0$ plane. However, this projection plane is not unique. Moreover, depending on the selected plane the optimization algorithm will need more or less steps in order to find a better configuration of the nodes. If the plane is well faced to $M(p)$ it will always take less steps, reducing the computational time.

The best faced plane P in which we can project is the one in which the area of the projected local sumbesh $N(q)$ is bigger. In [Escobar 06] the author proposes a method to find this plane based on a maximization problem of the area of $N(q)$ restricted to some constraints.

Here we present a different approach to the problem in which we avoid the maximization problem (therefore, improving the performance of the method). Our approach is based on the ideas developed in [Roca-Sarrate]. In this work it is proved that given a mesh node $\mathbf{x}$ and its local submesh $M(\mathbf{x})$ with external nodes $\mathbf{x_k},\ k = 0, \cdots, m-1$, the *pseudo-area* vector

$$\mathbf{a}^{\mathbf{M(x)}}_{\mathbf{pseudo}} := \frac{1}{2}\sum_{k=0}^{m-1}(\mathbf{x_k} - \mathbf{x}) \times (\mathbf{x_{k+1}} - \mathbf{x}), \tag{3.29}$$

is normal to the optimal projection plane. Note that in definition (3.29) we consider that $\mathbf{x_{m+1}} \equiv \mathbf{x_0}$.

Moreover, it is proved that the pseudo-area vector is independent of the central node $\mathbf{x}$, and that expression (3.29) can be computed more efficiently as

$$\mathbf{a}^{\mathbf{M(x)}}_{\mathbf{pseudo}} = \frac{1}{2}\sum_{k=0}^{m-1}\mathbf{x_k} \times \mathbf{x_{k+1}}. \tag{3.30}$$

The pseudo-area vector (3.30) defines the normal vector to the optimal plane, called the *pseudo-normal* vector,

$$\mathbf{n}^{\mathbf{M(x)}}_{\mathbf{pseudo}} := \frac{\mathbf{a}^{\mathbf{M(x)}}_{\mathbf{pseudo}}}{\|\mathbf{a}^{\mathbf{M(x)}}_{\mathbf{pseudo}}\|}. \tag{3.31}$$

In this work we denote $\mathbf{n} \equiv \mathbf{n}^{\mathbf{M(x)}}_{\mathbf{pseudo}}$.

Using the pseudo-normal we are able to define the optimal projection plane (see Figure 3.2). Suppose that $\{\mathbf{y_1}, \mathbf{y_2}, \mathbf{y_3}\}$ is a triangle of the submesh $M(p)$ with pseudo-normal $\mathbf{n}$ and that we want to compute its objective function through the optimal projection plane.

We consider a new $\mathbb{R}^3$ basis in which the two first coordinates are those of the plane P, and the third is the one in the direction $\mathbf{n}$. If $\mathbb{E} = \{\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}\}$ is the $\mathbb{R}^3$

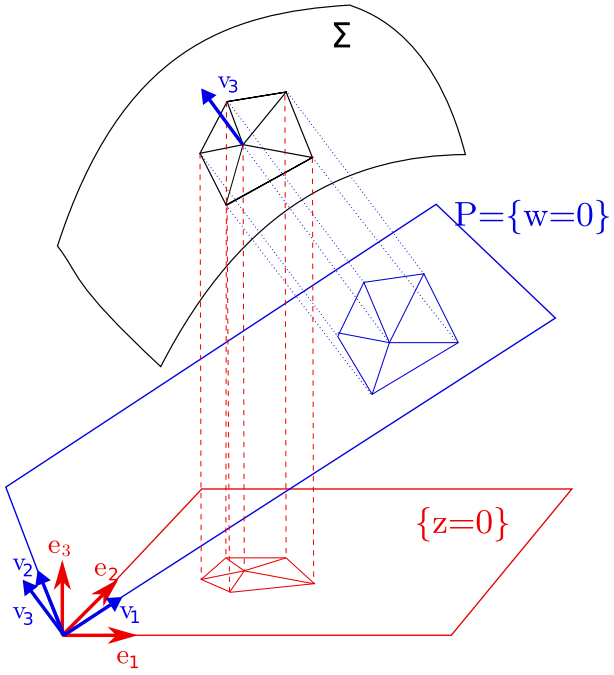

Figure 3.2: Pseudo-normal vector and optimal projection plane.

canonical basis, the new basis $\mathbb{V} = \{\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}\}$ can be expressed as

$$\begin{aligned} \mathbf{v_3} &= \mathbf{n}, \\ \mathbf{v_2} &= \frac{\mathbf{e_2} - (\mathbf{e_2}^t \cdot \mathbf{n})\mathbf{n}}{\|\mathbf{e_2} - (\mathbf{e_2}^t \cdot \mathbf{n})\mathbf{n}\|}, \\ \mathbf{v_1} &= \frac{\mathbf{e_2} \times \mathbf{n}}{\|\mathbf{e_2} \times \mathbf{n}\|}. \end{aligned}$$

Then we define the matrix that changes the basis from $\mathbb{E}$ to $\mathbb{V}$ as

$$\mathbf{M} = \begin{pmatrix} \mathbf{v_1^t} \\ \mathbf{v_2^t} \\ \mathbf{v_3^t} \end{pmatrix}. \tag{3.32}$$

If we denote by $\tilde{\mathbf{y}}_\mathbf{k}$ the position vectors of the nodes in basis $\mathbb{V}$ and by $\mathbf{y_k}$ their position on basis $\mathbb{E}$, using matrix (3.32) we can express the coordinates of the nodes of the mesh in this new basis:

$$\tilde{\mathbf{y}}_\mathbf{k} = \mathbf{M} \cdot \mathbf{y_k}, \ k = 1, 2, 3. \tag{3.33}$$

If we denote by (u, v, w) the coordinates of a point in basis $\mathbb{V}$, the projection plane P becomes the plane $w = 0$, because the normal of the plane coincides with the third vector of the new basis $\mathbb{V}$.

Now we just have to apply the already developed formulas in Sections 3.1 and 3.2 to the triangle $\{\tilde{\mathbf{y}}_\mathbf{k}\}_{k=1,2,3}$, working in coordinates (u, v, w) and no more in (x, y, z).

It is important to point out that the new matrices $\mathbf{A}_\pi$ and $\mathbf{A_P}$ in basis $\mathbb{V}$ (denoted by $\tilde{\mathbf{A}}_\pi$ and $\tilde{\mathbf{A}}_\mathbf{P}$) have to be computed as

$$\begin{aligned}
\tilde{\mathbf{A}}_\pi &= (\tilde{\mathbf{y}}_\mathbf{1} - \tilde{\mathbf{y}}_\mathbf{0} \;\; \tilde{\mathbf{y}}_\mathbf{2} - \tilde{\mathbf{y}}_\mathbf{0}) \\
&= \begin{pmatrix} u_1 - u_0 & u_2 - u_0 \\ v_1 - v_0 & v_2 - v_0 \\ w_1 - w_0 & w_2 - w_0 \end{pmatrix} \\
&= \mathbf{M} \cdot \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \\ z_1 - z_0 & z_2 - z_0 \end{pmatrix} \\
&= \mathbf{M} \cdot (\mathbf{y_1} - \mathbf{y_0} \;\; \mathbf{y_2} - \mathbf{y_0}) \\
&= \mathbf{M} \cdot \mathbf{A}_\pi
\end{aligned} \tag{3.34}$$

and

$$\begin{aligned}
\tilde{\mathbf{A}}_\mathbf{P} &= (\tilde{\mathbf{x}}_\mathbf{1} - \tilde{\mathbf{x}}_\mathbf{0} \;\; \tilde{\mathbf{x}}_\mathbf{2} - \tilde{\mathbf{x}}_\mathbf{0}) \\
&= \begin{pmatrix} u_1 - u_0 & u_2 - u_0 \\ v_1 - v_0 & v_2 - v_0 \end{pmatrix} \\
&= \mathbf{M} \cdot \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix} \\
&= \mathbf{M} \cdot (\mathbf{x_1} - \mathbf{x_0} \;\; \mathbf{x_2} - \mathbf{y_0}) \\
&= \mathbf{M} \cdot \mathbf{A_P}.
\end{aligned} \tag{3.35}$$

Then, the new objective function will be the result of the computation of expressions (3.26), (3.27) and (3.28), using matrices $\tilde{\mathbf{A}}_\pi$ and $\tilde{\mathbf{A}}_\mathbf{P}$ defined in (3.34) and (3.35).

Note that with this change of the projection plane, the only changes on the algorithm are the computation of matrices $\mathbf{A}_\pi$ and $\mathbf{A_P}$. The remaining of definitions and formulas are computed as it is detailed in Sections 3.1 and 3.2 but using the new expressions $\tilde{\mathbf{A}}_\pi$ and $\tilde{\mathbf{A}}_\mathbf{P}$ (equations (3.34) and (3.35)). The results of the optimization will be similar in both cases, but the convergence will increase using the new projection plane.

4 Mesh quality optimization

In Section 3 we have developed an objective function that does only depend on the two variables defined on the projection plane. This property avoids having to add to the optimization algorithm any surface constraints. However, at each step we will have to update the objective function in order to include the changes on the third coordinate.

It is also important to notice that the optimization will be based on a local smoothing [Escobar 03]. We smooth one by one all the submeshes defined by inner nodes in a Gauss-Seidel manner. Once you have optimized the position of one node, you use this new position to compute the new positions of the remaining nodes. Then we will have to keep repeating the procedure until we obtain global optimum configuration of the nodes.

4.1 Local optimization algorithm

Suppose that the initial position of the free node p is $\mathbf{y_0} = (\mathbf{x_0}, f(\mathbf{x_0}))^t$. In the first step of the optimization we compute $\mathbf{S^0}(\mathbf{x_0})$ (see equation (3.26)), keeping $\mathbf{y}$ on its initial value $\mathbf{y_0}$ (i.e. fixing the third coordinate at the initial value and mantaining constant the matrices that depend on it).

Then, we minimize the objective function that defines $\mathbf{S^0}(\mathbf{x})$ (see equations (3.27) and (3.28)). Suppose that $\mathbf{x_1}$ is the minimizing point. This new position is the first approximation to the minimum[5]. The result has to be improved updating the constant part of the objective function (see Section 3.2) at $\mathbf{y_1} = (\mathbf{x_1}, f(\mathbf{x_1}))^t$, and then repeating the procedure in order to find another minimizing point $\mathbf{x_2}$.

Repeating this procedure, we compute a minimizing sequence $\{\mathbf{x_k}\}_k$. We stop the procedure when we get a certain $\mathbf{x_{\tilde{k}}}$ that holds

$$\frac{K^{\tilde{k}+1} - K^{\tilde{k}}}{K^{\tilde{k}+1}} < tol, \tag{4.1}$$

where K^k is the computation of the ojective function (3.28) at the kth step and tol is a given tolerance. We select the optimal point as the approximation $\mathbf{x_{\tilde{k}}}$ that verifies condition (4.1). Then, we compute the optimal position of node p as $\mathbf{y_{\tilde{k}}} = (\mathbf{x_{\tilde{k}}}, f(\mathbf{x_{\tilde{k}}}))$.

We summarize this procedure in Algorithm 1.

In order to prevent a possible wierd case in which convergence of the algorithm could not be achieved, a maximum number of iterations has also been fixed.

Recall that Algorithm 1 optimizes the position of the central free node of a local submesh. Then it must be introduced in a loop over all the inner nodes in order to minimize the whole mesh.

[5]Note that $\mathbf{x_1}$ is not the exact minimum because we have approximated the shape matrix $\mathbf{S}$ by $\mathbf{S^0}$ fixing the third coordinate.

Algorithm 1 Local optimization

Return : Vector $\mathbf{y} \in \mathbb{R}^3$

function OPTIMIZESUBMESH(coordinates of the local submesh, connectivities of the elements of the local submesh, inner node of the submesh)
  Let *tol* be a given tolerance;
  $\mathbf{y_0} \leftarrow (\mathbf{x_0}, f(\mathbf{x_0}))$, initial position of the free node p on Σ;
  $\mathbf{y}$ fixed at value $\mathbf{y_0}$ ($z = z_0$);
  $\eta^0(\mathbf{x})$ and $\mathbf{S^0}(\mathbf{x})$ computed as (3.27) and (3.26) for all triangles of M(p);
  $|K^0_\eta(\mathbf{x})|$ computed as equation (3.28);
  $\tilde{K^0} = |K^0_\eta|(\tilde{\mathbf{x}}_\mathbf{0}) \leftarrow \min[|K^0_\eta|(\mathbf{x})]$;
  $\mathbf{x_1} \leftarrow \tilde{\mathbf{x}}_\mathbf{0}$;
  $\mathbf{y_1} \leftarrow \tilde{\mathbf{x}}_\mathbf{0}$;
  convergence←false;
  while convergence = false **do**
    $\mathbf{y}$ fixed at value $\mathbf{y_k}$ ($z = z_k$);
    $\eta^k(\mathbf{x})$ and $\mathbf{S^k}(\mathbf{x})$ are constructed for the all triangles of M(p);
    $|K^k_\eta|(\mathbf{x})$ is computed as (3.28);
    $\tilde{K^k} = |K^k_\eta|(\tilde{\mathbf{x}}_\mathbf{k}) \leftarrow \min[|K^k_\eta|(\mathbf{x})]$;
    if $|\frac{\tilde{K^k} - \tilde{K^{k-1}}}{\tilde{K^k}}| < tol$ **then**
      convergence←true;
    end if
    $\mathbf{x^{k+1}} \leftarrow \tilde{\mathbf{x}}^\mathbf{k}$;
    $\mathbf{y^{k+1}} \leftarrow (\mathbf{x^{k+1}}, f(\mathbf{x^{k+1}}))^t$;
    $k \leftarrow k + 1$;
  end while
  The optimal returned solution is $\mathbf{y^k}$;
end function

Algorithm 2 summarizes the global optimization of the mesh.

Algorithm 2 Global Optimization Algorithm

Return : Mesh M.

function GLOBALOPTIMIZATION(Mesh M)
  $d \leftarrow \infty$;
  $nnodes \leftarrow$ number of inner nodes of M;
  while $d > tol$ **do**
    $d \leftarrow 0$;
    for $k = 1 : nnodes$ **do**
      $y_k^0 \leftarrow$ initial coordinates of the kth node;
      $y_k \leftarrow$ OPTIMIZESUBMESH(Submesh $N(y_k)$);
      $d \leftarrow \max(d, \|y_k^0 - y_k\|)$;
    end for
  end while
end function

According to Algorithm 2, the optimization procedure is performed until the maximum desplacement prescribed over all the inner nodes is lower than a given tolerance *tol*.

4.2 Minization method

We have used a *line search* strategy to minimize the objective function. The line search strategies choose a direction $\mathbf{p_k}$ and search along this direction from the current point $\mathbf{x_k}$ in order to find a new position in which the function takes a lower value. We move in this direction a distance α that has to be determined. The new position is computed as

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \alpha \mathbf{p_k}. \tag{4.2}$$

We use the line-search strategy in the steepest descent direction:

$$\mathbf{p_k} = -\nabla \mathbf{f_k^t}. \tag{4.3}$$

This method is known as *steepest descent method* [Nocedal]. This is a quick method that does not require second derivatives. However, for a rough case, depending on the behavior of the objective function, the convergence of the method could be slow. Further discussion can be found in [Nocedal].

Once the advancing direction is fixed, it has to be decided the distance α that has to be covered from the initial position along the decided direction. According to [Nocedal], the *step length* α has to satisfy several conditions. We will use the *backtracking* approach to find α. This approach is indeed a simplification of Wolve conditions (see [Nocedal] for details). We summarize this procedure in Algorithm 3.

Algorithm 3 Backtracking Line Search

Return : Real α_k.

1: **function** BACKLINESEARCH(Vector $\mathbf{x_k}$, Vector p_k)
2: Fixed $\alpha > 0,\ \rho \in (0,1),\ c \in (0,1)$;
3: **while** $f(\mathbf{x_k} + \alpha \mathbf{p_k}) > f(\mathbf{x_k}) + c\alpha \nabla \mathbf{f_k^t} \mathbf{p_k}$ **do**
4: $\alpha \leftarrow \rho\alpha$;
5: **end while**
6: $\alpha_k \leftarrow \alpha$;
7: **end function**

Note that in this work function f in Algorithm 3 matches the objective function defined in equation (3.28) and the advancing direction is computed as $\mathbf{p_k} = -\nabla \mathbf{f_k^t}$. The other variables that determine the backtracking line search have also to be fixed. Due to the fact that in the Wolfe conditions the variable c has to be quite small, we have taken $c = 10^{-4}$ as it is proposed on [Nocedal]. In addition we fix $\rho = \frac{1}{2}$.

The steepest direction method only requires the computation of the first derivative of the minimizing function. The computation of the derivatives of the objective function can be found on Appendix A.2.

Recall that Algorithm 3 will be used in Algorithm 1 at each step in order to find $\min[|K_\eta^k|(\mathbf{x})]$.

4.3 Optimization on the projection plane

Recall that we compute the objective function through a orthogonal projection on a plane P that is in general different than the $z = 0$ plane.

The performance of the minimization algorithm 1 is directly affected by the selection of the projection plane, because it changes the expression of the required matrices involved in the objective function. Therefore, taking into account that this algorithm has to be applied several times to all the inner nodes, it is of the major importance to optimize its convergence.

According to the notation introduced in Section 3.3, let $\mathbb{E} = \{\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}\}$ be the canonical basis of $\mathbb{R}^3$ and $\mathbb{V} = \{\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}\}$ the orthonormal basis defined on P. Let $\mathbf{M}$ be the change of basis matrix from $\mathbb{E}$ to $\mathbb{V}$, see equation (3.32).

Recall that the surface Σ is parametrized by $(x, y, z)^t = s(x, y) = (x, y, f(x, y))^t$, equation (3.1), where (x, y) belongs to the $z = 0$ plane. Note that we do not have the parametrization of the surface in function of the variables (u, v) on P.

Given an inicial position $\mathbf{y_0} = (x^0, y^0, z^0)^t$ of the free node on Σ, we obtain its expression in basis $\mathbb{V}$, $(u^0, v^0, w^0)^t$, by applying the transformation (3.33). Then, we poject this point on plane P ($w = 0$), see Figure 3.2. This way we obtain the inicial position $(u^0, v^0)^t$ for the 2D optimization procedure on plane P.

During the optimization process we compute the advancing direction[6] $\mathbf{p_k} = (p_k^u, p_k^v)^t$ and the step length α using the backtracking line search method, see Algorithm 3. Then we update the position on P according to equation (4.3). That is

$$(u^n, v^n)^t = (u^0, v^0)^t + \alpha \mathbf{p_k}. \tag{4.4}$$

At this point, several strategies can be followed in order to obtain the new position of the node $(x^n, y^n, z^n)^t$ on the surface Σ. In this work we have applied the following scheme:

Note that according to equation (4.4) we have the new coordinates (u^n, v^n) of the point on the basis induced by $\mathbb{V}$ on P. Therefore we do not have the third coordinate w^n in order to apply the transformation (3.33) and express the point in canonical coordinates, in which we have the explicit parametrization of the surface $s(x, y)$ defined on the $z = 0$ plane.

We follow taking $w^n \approx w^0$. It's clear that this approximation $(u^n, v^n, w^0)^t$ is not on the surface. Though, for small changes in the smoothing process this approximation leads to a point that is near to the optimal point on the surface.

Then we undo the change of coordinates

$$(\tilde{x}^n, \tilde{y}^n, \tilde{z}^n)^t = \mathbf{M^{-1}}(u^n, v^n, w^n)^t. \tag{4.5}$$

Note that we have just changed the coordinates in which the point is expressed. The point is still the same, and it is not on the surface. Once in canonical reference coordinates, we project this point into the plane $z = 0$, $(\tilde{x}^n, \tilde{y}^n)^t$. Finally we take it back to the surface using the parametrization $s(x, y)$, obtaining a good approximation on the surface of the optimal point:

$$\begin{pmatrix} x^n \\ y^n \\ z^n \end{pmatrix} = \begin{pmatrix} \tilde{x}^n \\ \tilde{y}^n \\ f(\tilde{x}^n, \tilde{y}^n) \end{pmatrix}. \tag{4.6}$$

Note that inicially we may be applying big displacements, $\alpha \mathbf{p_k}$, in which case we would be committing some error on this approximation. Although, as we get near to the optimal point, the correction displacements $\alpha \mathbf{p_k}$ become really small, in which case this approximation is nearly exact. Thus the algorithm is going to converge to the optimal position.

[6]Recall that the advancing direction is minus the gradient of the objective function on P: $\mathbf{p_k} = (p_k^u, p_k^v)^t = -\left(\frac{\partial K_\eta^k}{\partial u}, \frac{\partial K_\eta^k}{\partial v}\right)^t$.

5 Examples: application to parametrized surfaces

In this section several examples are presented in order to assess the properties of the proposed method. In this work, the developed algorithms have been implemented using Matlab. Since Matlab does no support Nurbs surfaces, we have assumed that the parametrization of the surface is described according equation (3.1). Therefore, in all the examples the parametric space is the $z = 0$ plane.

To highlight the differences between the shape of the element on the surface mesh and its quality, we present six figures for all the examples. We first plot the initial triangular mesh both on the parametric plane $z = 0$ and on the surface. Second, to emphasize the capabilities of the proposed method to untangle meshes, the initial location of the nodes on the parametric space has been randomized. Therefore, almost all the elements of the inicial mesh are tangled, and two figures of this new mesh are shown (one on the parametric space and the other o the surface). Then the randomized mesh will be the input of our program. Finally we present the mesh after the optimization procedure (two figures are presented, one on the parametric space and one on the surface).

For all the examples we provide statistical information about the quality of the elements of the mesh. In particular we compute for both the initial (not randomized) mesh and for the optimized mesh, the following information:

- Minimum quality value over all the elements of the mesh.
- Maximum quality value over all the elements of the mesh.
- Mean value of the quality of the elements of the mesh.
- Standard deviation of the quality of the elements of the mesh.

Finally, after the presentation of all the examples, we also provide the following information for each example:

- Number of elements of the mesh.
- Number of iterations of the global algorithm.
- Elapsed time for minimization process.

In order to be able to compare the performance of the method for the surfaces presented in this section, all the parameters involved in the algorithm have been fixed at the same value. The most influential parameters even for the quality of the mesh, and for the computational time of the program are the tolerances that are used as stopping criteria in Algorithms 1 and 2.

- For Algorithm 1, with stopping criteria $|\frac{\tilde{K}^k - \tilde{K}^{k-1}}{\tilde{K}^k}| < tol$, the tolerance has been fixed as $tol = 10^{-7}$.

- For Algorithm 2, with stopping criteria $d < tol$ ($d \equiv$ maximum imposed displacement over all the inner nodes), the tolerance has been choosen as $tol = 10^{-4}$.

The stopping distance of Algorithm 2 influences in the same way all the examples. Since it measures the maximum imposed displacement, the value has to be choosen depending on the size of the elements. Then, as we are going to work with meshes that will have similar sizes it will make no difference between the analysed cases.

However, the tolerance for Algorithm 1 directly affects the number of iterations performed in each case. Taking $tol = 10^{-7}$ we ensure good results in all the examples, but we have to point out that in some of the examples it would be enough to take $tol = 10^{-5}$. In this case the computational time is reduced. Thus in some examples we could improve the computational statistics that are presented. However, we use the same values in all the examples in order to present a fair comparison.

5.1 Surface mesh 1

The first analysed surface is the plane $z = x$, with the parametrization defined by $f(x, y) = x, \quad (x, y) \in [0, 1] \times [0, 1]$. Figures 5.1 to 5.6 present the generated meshes.

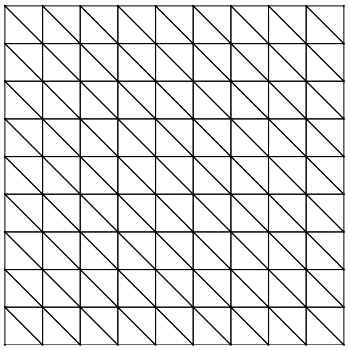

Figure 5.1: Planar mesh.

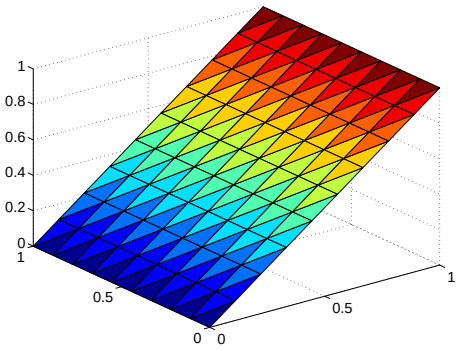

Figure 5.2: Planar mesh mapped into the surface.

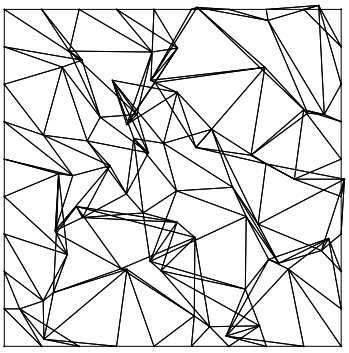

Figure 5.3: Randomized mesh.

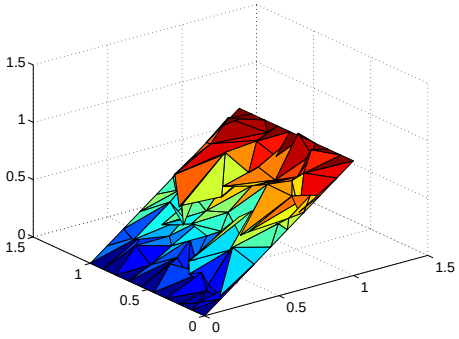

Figure 5.4: Randomized mesh mapped into the surface.

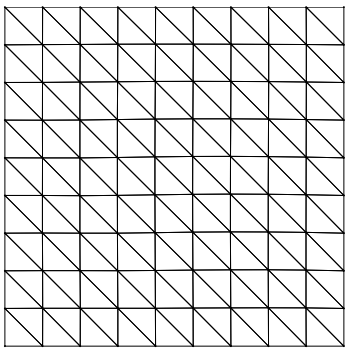

Figure 5.5: Projected final mesh after the application of the optimization.

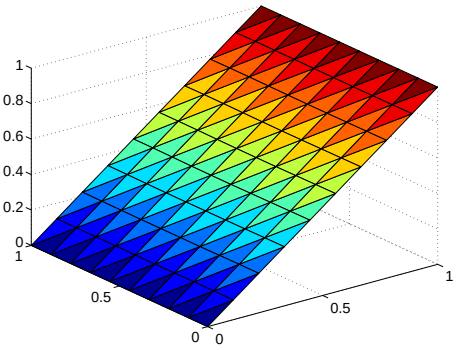

Figure 5.6: Final surface mesh after the application of the optimization.

Table 1 presents the statistical analysis of the mesh before and after the optimization process.

	Minimum	Maximum	Mean quality	Std. deviation
Initial mesh	0.816	0.816	0.816	0.000
Optimized mesh	0.809	0.824	0.816	0.003

Table 1: Statistical values for example 1.

For this surface, a plane with constant slope, the optimized mesh is the same that the original one. However, we have to emphasize that the algorithm has been able not even to untangle the randomized mesh, but also to reach almost the same original optimal configuration.

5.2 Surface mesh 2

The second example is the surface $z = x(x-1)y(y-1)$, with the parametrization defined by $f(x,y) = x(x-1)y(y-1), \quad (x,y) \in [0,1] \times [0,1]$.

Figures 5.7 to 5.12 present the generated meshes.

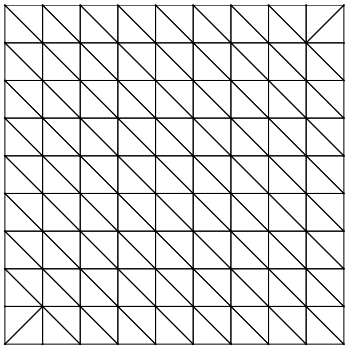

Figure 5.7: Planar mesh.

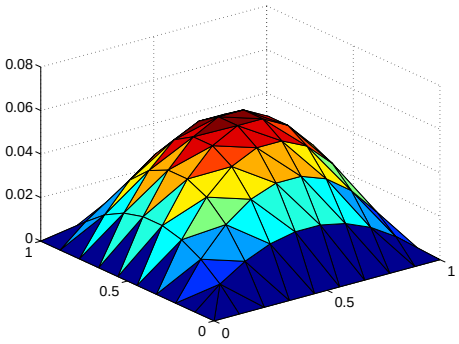

Figure 5.8: Planar mesh mapped into the surface.

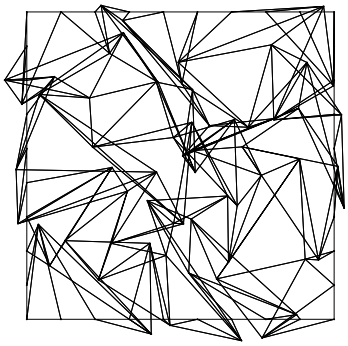

Figure 5.9: Randomized mesh.

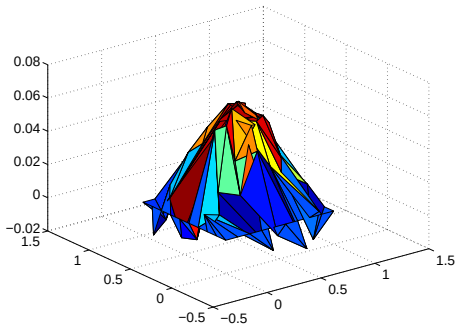

Figure 5.10: Randomized mesh mapped into the surface.

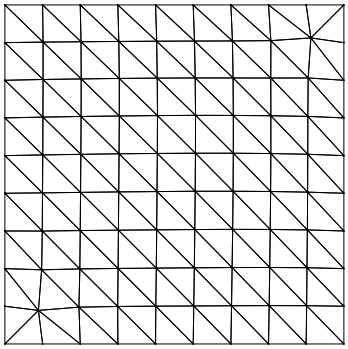

Figure 5.11: Projected final mesh after the application of the optimization.

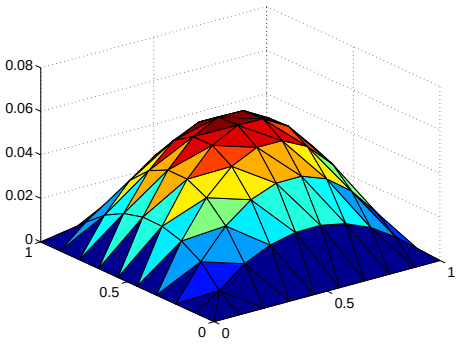

Figure 5.12: Final surface mesh after the application of the optimization.

Table 2 presents the statistical analysis of the initial and the optimized mesh.

	Minimum	Maximum	Mean quality	Std. deviation
Initial mesh	0.862	0.871	0.866	0.002
Optimized mesh	0.853	0.876	0.867	0.006

Table 2: Statistical values for example 2.

Once again, due to the simetry and the smoothness of the surface, the optimized mesh is similar to the equispaced mesh generated for the planar square domain.

5.3 Surface mesh 3

In this example the surface is defined by $f(x, y) = x^2y^2$, $(x, y) \in [0, 1] \times [0, 1]$. For this surface the values of the gradient are high near the point $(1, 1)$ and different results between the initial and the optimized mesh should be observed near this point. Figures 5.13 to 5.18 present the generated meshes.

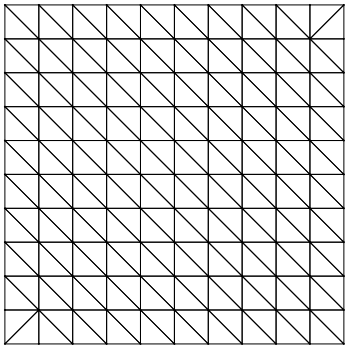

Figure 5.13: Planar mesh.

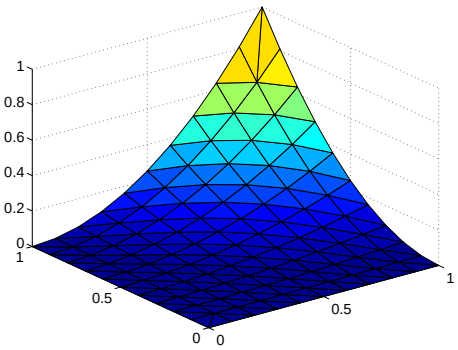

Figure 5.14: Planar mesh mapped into the surface.

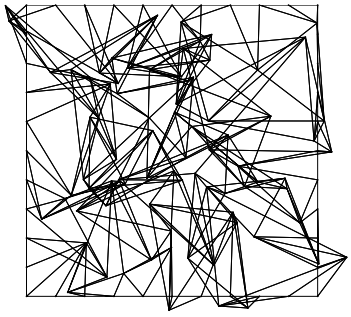

Figure 5.15: Randomized mesh.

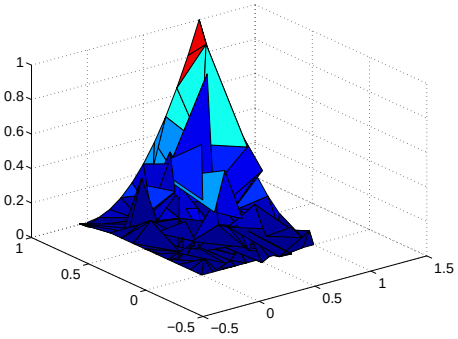

Figure 5.16: Randomized mesh mapped into the surface.

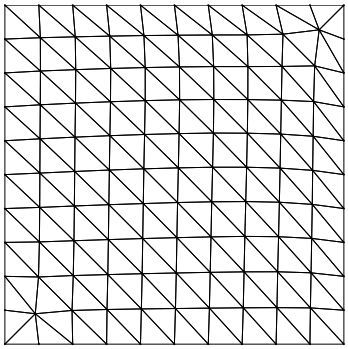

Figure 5.17: Projected final mesh after the application of the optimization.

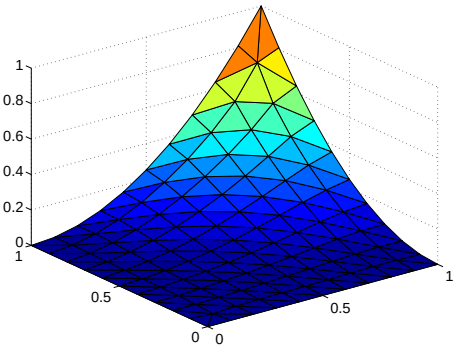

Figure 5.18: Final surface mesh after the application of the optimization.

Table 3 presents the statistical analysis of the mesh before and after the optimization process.

	Minimum	Maximum	Mean quality	Std. deviation
Initial mesh	0.866	0.999	0.899	0.042
Optimized mesh	0.829	1	0.900	0.037

Table 3: Statistical values for example 3.

Figures 5.13 and 5.17 show the initial and the optimized mesh. Although the qualities of both meshes are similar, and they seem nearly the same mesh, small differences appear. The optimization tries to approximate in the best possible way the surface, and this can lead to a few different mesh, even though the original one is good enough. The small difference between both meshes is in the upper-right corner, where the distribution of the elements has changed on the optimized mesh in order to capture properly the high gradient surface.

However the initial and the optimized mesh are nearly the same. If the smoothing algorithm was abel to move the boundary nodes, it would provide elements with better shape. Though, as the boundary nodes are fixed the algorithm is not able to move the inner nodes close to the boundary. Otherwise, the quailty of those elements will decrease. Despite this constraint, the method is able to untangle the mesh and achieve a final configuration with a quality similar to the original one.

5.4 Surface mesh 4

In the fourth example the surface defined by $f(x, y) = sin(\pi x)cos(\pi y), \quad (x, y) \in [0, 1] \times [0, 1]$. Figures 5.19 to 5.24 present the generated meshes.

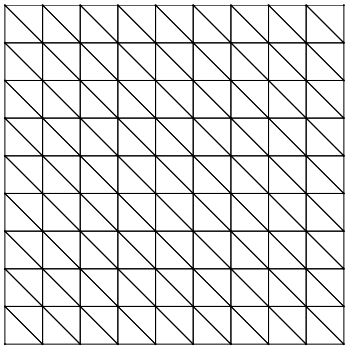

Figure 5.19: Planar mesh.

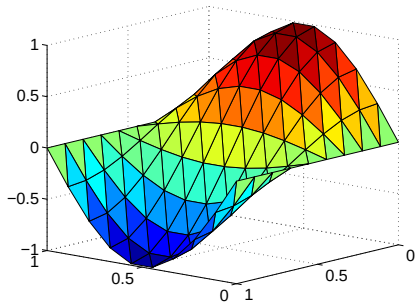

Figure 5.20: Planar mesh mapped into the surface.

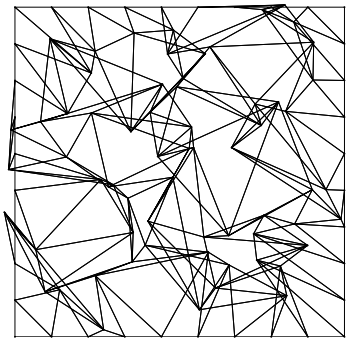

Figure 5.21: Randomized mesh.

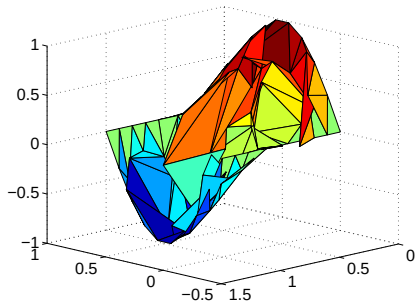

Figure 5.22: Randomized mesh mapped into the surface.

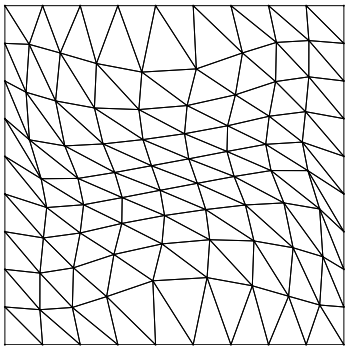

Figure 5.23: Projected final mesh after the application of the optimization.

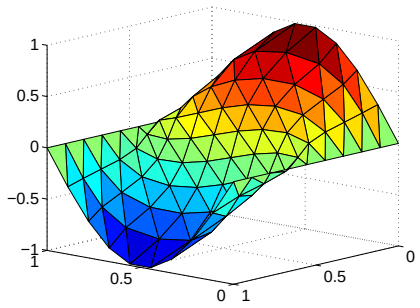

Figure 5.24: Final surface mesh after the application of the optimization.

Table 4 presents the statistical analysis of the mesh before and after the optimization process.

	Minimum	Maximum	Mean quality	Std. deviation
Initial mesh	0.455	0.955	0.656	0.164
Optimized mesh	0.440	0.990	0.742	0.148

Table 4: Statistical values for example 4.

In the previous examples, the differences between the initial and the final mesh were minimal. However, in this example the gradient of the surface reaches higher values, hence the final mesh differs from the initial one.

Figure 5.23 presents distorted elements on the parametric plane. Though, when the mesh is mapped to the surface (Figure 5.24), a good quality mesh is obtained (with almost equilateral triangles on the center of the domain). Recall that the optimization procedure modifies the elements on the projection plane in order to become ideal on the surface.

However, the generated elements close to the boundary are not as good as the ones from the inner domain. This is due to the fact that we cannot move the equispaced elements on the contour. Then, when the inner nodes are restructured, the elements on the boundary become a little more distorted than as they were originally. If the program was able to smooth the contour a better mesh would be obtained, with either a higher minimum quality and also a better mean quality of the resulting mesh.

5.5 Surface mesh 5

In the last example the surface is defined on the domain $[-2, 2] \times [-2, 2]$ by

$$f(x,y) = \begin{cases} a, & r \leq r_1 \\ (b-a)\frac{r-r_1}{r_2-r_1} + a, & r_1 < r < r_2 \\ b, & r >= r_2 \end{cases}$$

where $r = x^2 + y^2$, $r_1 = 0.75$, $r_2 = 1.75$, $a = 2$ and $b = 0$.

Figures 5.25 to 5.30 present the generated meshes.

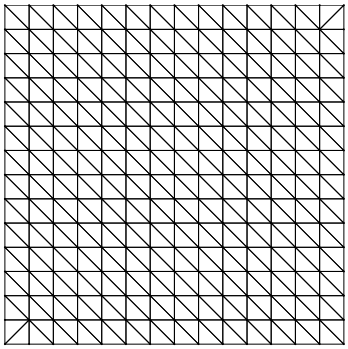

Figure 5.25: Planar mesh.

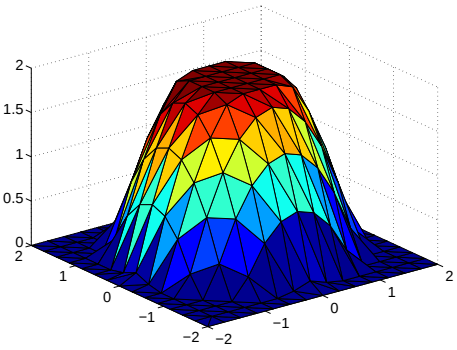

Figure 5.26: Planar mesh mapped into the surface.

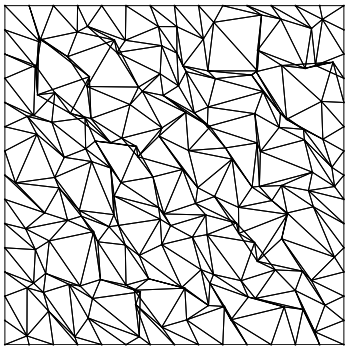

Figure 5.27: Randomized mesh.

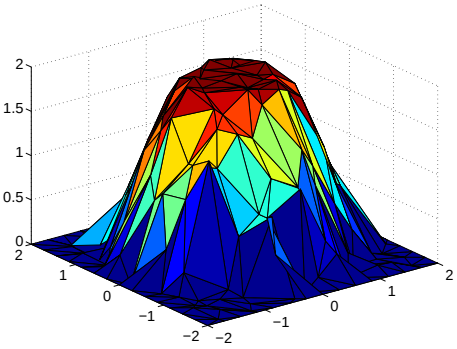

Figure 5.28: Randomized mesh mapped into the surface.

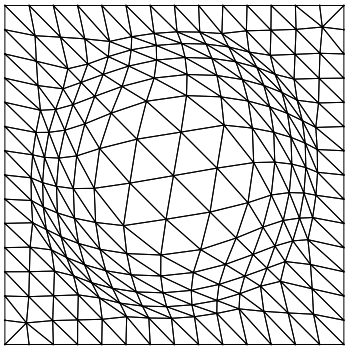

Figure 5.29: Projected final mesh after the application of the optimization.

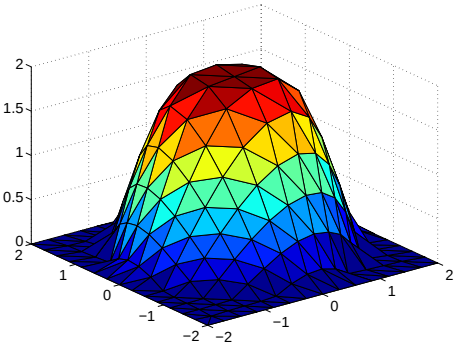

Figure 5.30: Final surface mesh after the application of the optimization.

Table 5 presents the statistical analysis of the mesh before and after the optimization process.

	Minimum	Maximum	Mean quality	Std. deviation
Initial mesh	0.442	0.995	0.780	0.148
Optimized mesh	0.681	0.998	0.861	0.082

Table 5: Statistical values for example 5.

From table 5 we realize that the optimized mesh differs from the initial one. The obtained mesh has relocated the nodes in order to approximate in a better way the big slope of the surface. Thus, the procedure results in a better mesh composed by triangles being almost equilateral.

5.6 Summary of computational aspects of the examples

Statistics \ Example	1	2	3	4	5
Number of elements	162	162	200	162	392
Number of iterations	20	35	25	36	107
Elapsed time (sec.)	303.17	809.54	812.15	430.65	1168.78

Table 6: Computational statistics for all the examples.

Table 6 presents several computational features for all the examples presented in the previous sections. First, note that the elapsed time of the procedure depends on the type of surface and also on the number of elements of the mesh. As expected, the computational cost of the proposed method is high. However, we have to emphasize that the initial configuration of the nodes is tangled, and so, the algorithm has not only to smooth but also to untangle an extremelly low quality mesh.

Several better results are obtained if the initial mesh is not randomized. One of the main capabilities of the algorithm is that is able to untangle the initial mesh, but note that there's a second important application of the developed algorithm. The algorithm can be used to improve the quality of a non tangled initial mesh. In that case the elapsed time will decrease. In Table 7 we present the same statistics that we have previously displayed taking as input of the optimization algorithm the initial not randomized mesh.

Statistics \ Example	1	2	3	4	5
Number of elements	162	162	200	162	392
Number of iterations	1	4	26	33	102
Elapsed time (sec.)	0.37	13.82	339.43	362.41	1109.97

Table 7: Computational statistics for all the examples taking as initial mesh, the not randomized one.

Then we can conclude this section extracting that altough the elapsed time needed to untangle and smooth a mesh is high, we have to be aware that the algorithm has been able to untangle a really bad configuration of the mesh. This illustrates the robustness of the method. If it is only used to smooth an untangled mesh, the computational cost is reduced drastically. And then, this shows the efficency of the method.

However, note that for the fifth example the computation time has not decreased when we have carried out the optimization with the initial not randomized mesh. On the one hand, the higher elapsed time is due to the fact that the mesh and the domain are bigger in this example than in the others. On the other hand, this is also due to the fact that the algorithm has to reach a new configuration of the nodes that differs from the initial one. Note that although the initial mesh was not tangled, the algorithm still has to apply many corrections to the original position of the nodes in order to improve the shape of the elements.

6 Conclusions and future research

The procedure that has been developed in this work is able to simultaneously smooth and untangle surface triangular meshes. Moreover the smoothing is carried out obtaining a mesh with better surface quality than the initial not randomized mesh. The procedure is based on the algorithm developed by Escobar in [Escobar 03]. We have increased the computational efficency of the original method by adding a new algorithm for the search of the optimal projection plane based on [Roca-Sarrate]. Moreover we have carried out a more robust discussion of the theoretical construction of the objective function for this new algorithm (Section 3 and Appendix A.1).

However future research has to be developed in order to improve the efficiency of the developed algorithm. Here we present future research lines in order to improve the performance of the proposed method:

- It is important to point out that no effort has been made in this work in order to speed up the matlab application. Moreover, a simple data structure has been used in the current implementation. Therefore, special attention will have to be paid on this issues when the algorithm will be translated to the `EZ4U` environment.

- Optimization of the developed codes (including data structure) in order improve the performance of the algorithms.

- Boundary optimization: the developed algorithm only relocates the inner nodes of the mesh. Thus we have seen in some proposed examples that although the mean quality has increased, the minimum quality has sometimes slightly decreased. The worse elements are placed over the boundary. Therefore, the quality of these elements can be improved if the algorithm can move boundary nodes.

- Change of programming language: we want to translate the codes from Matlab to C++ in order to optimize the efficency of the method and also enlarge the range of examples in which we can test the algorithms and the range of applications in which the method can be used.

- Update the C++ codes in the mesh generator environment `EZ4U` developed by *Laboratori de Càlcul Numèric*.

- Extension to quadrilateral surface meshes of the developed method for triangular elements.

A Appendices

A.1 Relation between f_Q and f_R affine mappings

In this appendix we deduce the relation between the applications derived from $\mathbf{f}_{\mathbf{A}_\pi}$. We will show why the affine mapping that maps τ to τ_R is exactly $\tilde{\mathbf{f}}_{\mathbf{R}}$, with Jacobian matrix $\mathbf{R}$. Moreover we analyse the differences between $\mathbf{f_Q}$ and $\mathbf{f_R}$. Similarly to Section 3.1 in this appendix we conside the projection plane P as the $\{z = 0\}$ plane. Extension to an arbitrary plane P can be performed following Section 3.3.

Figure 3.1 presents a graphical representation of the affine mapping involved in the smoothing method. We reproduce if here again for completness.

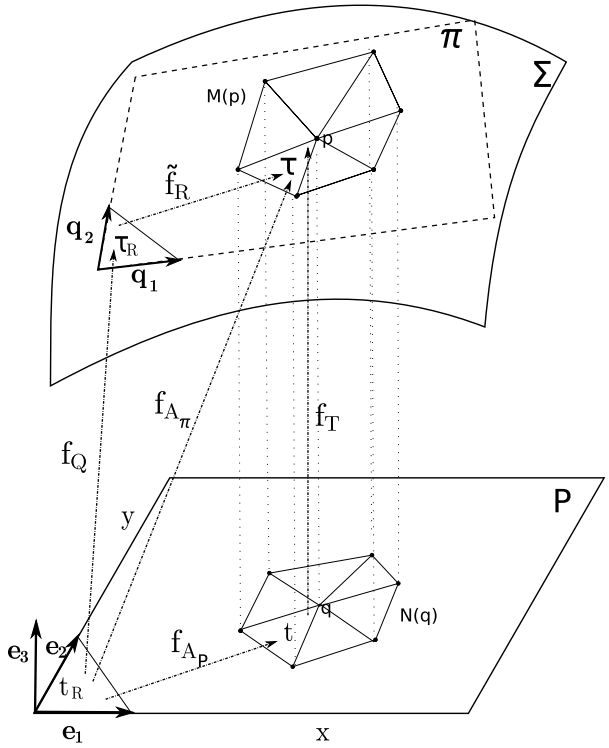

Figure A.1: Applications between the local surface mesh $M(p)$ and the associated parametric mesh $N(q)$.

Figure A.1 can be misunderstood and can bring to confusion when trying to analyse the applications that appear on it. The confusion comes from the fact that $\mathbf{f}_{\mathbf{A}_\pi} = \mathbf{f_Q} \circ \mathbf{f_R}$, but in Figure 3.1 they are represented in a different order and then it seems that we state that $\mathbf{f}_{\mathbf{A}_\pi}$ also holds that $\mathbf{f}_{\mathbf{A}_\pi} = \tilde{\mathbf{f}}_{\mathbf{R}} \circ \mathbf{f_Q}$. It is clear that this second expression is not true. In order to justify the developed ideas lets analyse the troublesome functions of Section 3.1.

First, recall that the affine mapping $\tilde{\mathbf{f}}_{\mathbf{R}}$ that appears on Figure 3.1 is considered on plane π, with coordinates on basis $\{\mathbf{q_1}, \mathbf{q_2}\}$.

Then, as we have already done for mappings $\mathbf{Q}$ and $\mathbf{W_E}$ in equations (3.11) and (3.12), we must distinguish that $\mathbf{R}$ can be understood as the Jacobian matrix of:

- $\mathbf{f_R} : t_R \longrightarrow t_u$, $t_u = \mathbf{f_R}(t_R)$, that is expressed on the local basis of plane P. This mapping holds that
$$\mathbf{f_{A_\pi}} = \mathbf{f_Q} \circ \mathbf{f_R}.$$
- $\tilde{\mathbf{f}}_\mathbf{R} : \tau_R \longrightarrow \tau$, that is expressed on the local basis of plane π, $\{\mathbf{q_1}, \mathbf{q_2}\}$. Then it cannot even be considerated the composition $\tilde{\mathbf{f}}_\mathbf{R} \circ \mathbf{f_Q}$ because each application is considered in a different coordinate system.

Note that we are working in two planes and that locally each plane generates a 2D basis in which we can work with no need of the third coordinate. However, in the analysis presented in this appendix, all the computatios will be performed considering the 3D representation of each node (even when they are projected on the selected plane).

The Jacobian matrix $\mathbf{A}_\pi$ is then a 3×3 dimensional matrix in which the two first columns are defined as in (3.4) and the third is just the normalized vectorial product of the two first columns. Then if we denote by $\mathbf{a_1} = \mathbf{y_1} - \mathbf{y_0}$ and $\mathbf{a_2} = \mathbf{y_2} - \mathbf{y_0}$, the definition of $\mathbf{A}_\pi$ changes into:

$$\mathbf{A}_\pi = \left(\mathbf{a_1} \;\; \mathbf{a_2} \;\; \frac{\mathbf{a_1} \times \mathbf{a_2}}{\|\mathbf{a_1} \times \mathbf{a_2}\|} \right).$$

This Jacobian matrix defines exactly the same application $\mathbf{f_{A_\pi}} : P \longrightarrow \pi$ that we have been considering, but defined on the canonical basis $\mathbb{E} = \{\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}\}$.

Imposing the same $\mathbf{QR}$ factorization, we obtain a 3×3 orthogonal matrix $\mathbf{Q}$ and a 3×3 matrix $\mathbf{R}$ such that $R_{ii} > 0,\ i = 1, 2, 3$.

As happened before, $\mathbf{Q} = (\mathbf{q_1}\ \mathbf{q_2}\ \mathbf{q_3})$ maps the vectors of the canonical basis $\mathbb{E}$ to the vectors of the basis defined on π, $\mathbb{M} = \{\mathbf{q_1}\ \mathbf{q_2}\ \mathbf{q_3}\}$. Using the subscripts $_\mathbb{E}$ and $_\mathbb{M}$ to distinguish if a vector is expressed on basis $\mathbb{E}$ or $\mathbb{M}$ respectively we can then write

$$\mathbf{q_{i,\mathbb{E}}} = \mathbf{Q} \cdot \mathbf{e_{i,\mathbb{E}}}.$$

Then, $\mathbf{Q}^t$ is the change of basis matrix from $\mathbb{E}$ to $\mathbb{M}$:

$$\mathbf{e_{i,\mathbb{M}}} = \mathbf{Q^t} \cdot \mathbf{e_{i,\mathbb{E}}} = (\mathbf{q_1}_{,\mathbb{E}}{}^t \cdot \mathbf{e_{i,\mathbb{E}}} \quad \mathbf{q_2}_{,\mathbb{E}}{}^t \cdot \mathbf{e_{i,\mathbb{E}}} \quad \mathbf{q_3}_{,\mathbb{E}}{}^t \cdot \mathbf{e_{i,\mathbb{E}}}).$$

Note that as $\mathbf{Q}$ is orthogonal, $\mathbf{Q}^t = \mathbf{Q^{-1}}$.

Now,

$$\mathbf{A}_\pi = \mathbf{QR} = \mathbf{QR} \cdot \mathbf{Q}^t\mathbf{Q} = (\mathbf{Q}^t)^{-1}\mathbf{R}\mathbf{Q}^t \cdot \mathbf{Q},$$

and defining the change of basis function $\varphi(\mathbf{y}) = \mathbf{Q}^t\mathbf{y}$, we can finally stablish the different application identities that could be initially confusing (see Figure A.2)

$$\begin{aligned} \mathbf{f_{A_\pi}} &= \mathbf{f_Q} \circ \tilde{\mathbf{f}}_\mathbf{R} \\ \mathbf{f_{A_\pi}} &= \varphi^{-1} \circ \tilde{\mathbf{f}}_\mathbf{R} \circ \varphi \circ \mathbf{f_Q}. \end{aligned}$$

Figure A.2 shows the relations that we have developed. All the mappings that appear on this figure are expressed in the same reference coordinates (basis $\mathbb{E}$).

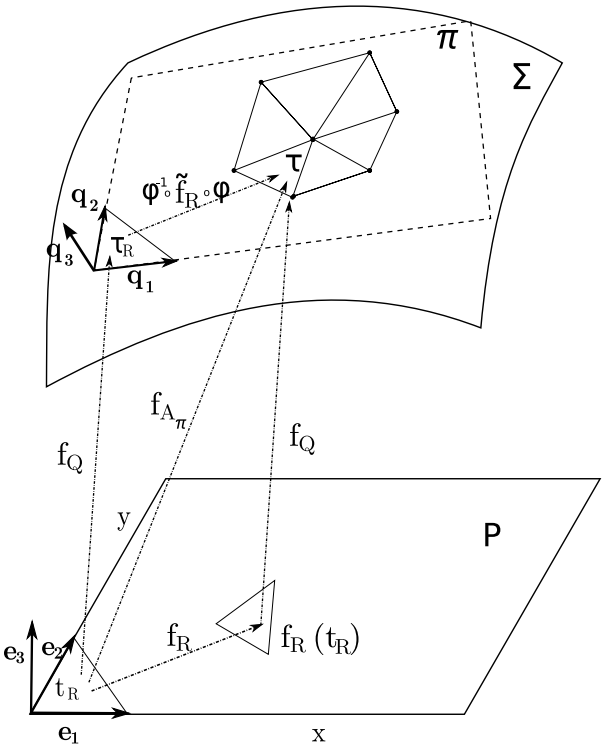

Figure A.2: Applications derived from the **QR** factorization of $\mathbf{A}_\pi$, expressed on the canonical basis.

Figure A.3 shows the relationship between the applications that are defined on plane P using basis $\mathbb{E}$.

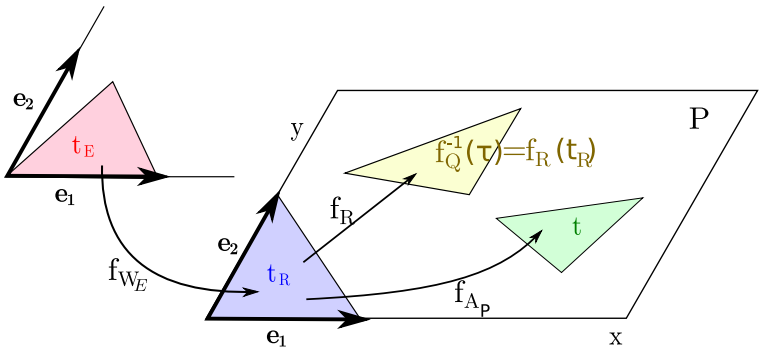

Figure A.3: Applications involved in plane P expressed in basis $\mathbb{E}$.

Figure A.4 shows the relationship between the mappings that are defined on plane π using basis $\mathbb{M}$.

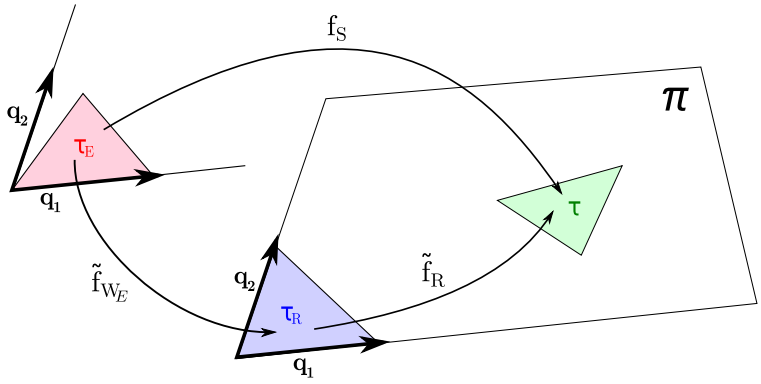

Figure A.4: Applications involved in plane π expressed in basis $\mathbb{M}$.

Figures A.3 and A.4 clearly state the applications that are involved in planes P and π. Therefore, they allow a deep understanding of the developed algotithm.

A.2 Computation of the derivatives of the objective function

In the minimization procedure that we develop in order to find the optimal configuration of the mesh we need to compute the derivatives of the objective function. According (3.28), the objective function that we have used for the surface mesh optimization is

$$K_\eta^0(\mathbf{x}) = \left(\sum_{k=1}^{m} \left(\eta_k^0\right)^p (\mathbf{x}) \right)^{1/p}, \tag{A.1}$$

where η_k^0 is defined in (3.27) by

$$\eta_k^0(\mathbf{x}) = \frac{|\mathbf{S_k^0}(\mathbf{x})|^2}{2h(\sigma_k^0(\mathbf{x}))}, \tag{A.2}$$

being $\sigma_k^0(\mathbf{x}) = \det\left(\mathbf{S_k^0}(\mathbf{x})\right)$.

Moreover $\mathbf{S_k^0}(\mathbf{x})$ was defined in (3.26) by

$$\mathbf{S^0}(\mathbf{x}) = \mathbf{R^0}(\mathbf{A_P^0})^{-1}\mathbf{S_E}(\mathbf{x}) = \mathbf{R^0}(\mathbf{A_P^0})^{-1}\mathbf{A_P}(\mathbf{x})\mathbf{W_E^{-1}}, \tag{A.3}$$

Recall that $\mathbf{R^0}$, $\mathbf{A_P^0}$ and $\mathbf{W_E}$ are constant.

Then, to compute the derivatives of the objective function we must apply the chain rule to these expressions. In order to simplify the notation, lets denote by α the variable respect to which we are computing the derivative (since $\mathbf{x} = (x, y)^t$, then α can be equal to x or y).

- First we deduce the expression corresponding to the derivative of equation (A.1):

$$\begin{aligned} \frac{\partial K_\eta^0(\mathbf{x})}{\partial \alpha} &= \frac{1}{p}\left(\sum_{k=1}^{m} \left(\eta_k^0\right)^p (\mathbf{x}) \right)^{\frac{1}{p}-1} \cdot \sum_{k=1}^{m} \left(p\left(\eta_k^0\right)^{p-1}(\mathbf{x}) \cdot \frac{\partial \eta_k^0}{\partial \alpha}(\mathbf{x}) \right) \\ &= \left(\sum_{k=1}^{m} \left(\eta_k^0\right)^p (\mathbf{x}) \right)^{\frac{1}{p}-1} \cdot \sum_{k=1}^{m} \left(\left(\eta_k^0\right)^{p-1}(\mathbf{x}) \cdot \frac{\partial \eta_k^0}{\partial \alpha}(\mathbf{x}) \right). \end{aligned}$$

- Second, we deduce the expression corresponding to the derivative of equation (A.2). Recall that $|\mathbf{S}| = \sqrt{(\mathbf{S}, \mathbf{S})}$, where $(\mathbf{A}, \mathbf{B}) = \mathrm{tr}(\mathbf{A}^t\mathbf{B})$. To this end we also need the following derivatives:

  - □ Derivative of the Frobenius norm of matrix $\mathbf{S}$:

$$\frac{\partial |\mathbf{S}(\mathbf{x})|}{\partial \alpha} = \frac{\partial \sqrt{(\mathbf{S}, \mathbf{S})}}{\partial \alpha} = \frac{1}{2}\frac{1}{\sqrt{(\mathbf{S}, \mathbf{S})}}\frac{\partial (\mathbf{S}, \mathbf{S})}{\partial \alpha} = \frac{1}{2}\frac{1}{|\mathbf{S}|}\frac{\partial (\mathbf{S}, \mathbf{S})}{\partial \alpha} = \frac{\left(\frac{\partial \mathbf{S}}{\partial \alpha}, \mathbf{S}\right)}{|\mathbf{S}|}.$$

□ Derivative of function $h(\sigma)$ (see equation (2.11)):

$$\frac{\partial h(\sigma(\mathbf{x}))}{\partial \alpha} = \frac{\partial}{\partial \alpha}\left(\frac{1}{2}\left(\sigma + \sqrt{\sigma^2 + 4\delta^2}\right)\right) = \frac{1}{2}\left(1 + \frac{\sigma}{\sqrt{\sigma^2 + 4\delta^2}}\right)\frac{\partial \sigma}{\partial \alpha}.$$

□ To compute the derivative of σ we just have to apply the formula for the derivative of a determinant of a matrix, see reference [Golberg]. Then,

$$\frac{\partial \sigma(\mathbf{x})}{\partial \alpha} = \frac{\partial \det(\mathbf{S})}{\partial \alpha} = \det(\mathbf{S})\mathrm{tr}\left(\mathbf{S^{-1}}\frac{\partial \mathbf{S}}{\partial \alpha}\right) = \sigma \mathrm{tr}\left(\mathbf{S^{-1}}\frac{\partial \mathbf{S}}{\partial \alpha}\right).$$

With these previous calculations, now lets proceed with the derivative of $\eta_k^0(\mathbf{x})$:

$$\begin{aligned}
\frac{\partial \eta}{\partial \alpha}(\mathbf{x}) &= \frac{\partial |\mathbf{S}|^2}{\partial \alpha}\frac{1}{2h(\sigma)} + |\mathbf{S}|^2\frac{\partial}{\partial \alpha}\left(\frac{1}{2h(\sigma)}\right) \\
&= \frac{1}{2h(\sigma)}2|\mathbf{S}|\frac{\partial |\mathbf{S}|}{\partial \alpha} - |\mathbf{S}|^2\frac{1}{2(h(\sigma))^2}\frac{\partial h(\sigma)}{\partial \alpha} \\
&= \frac{\left(\frac{\partial \mathbf{S}}{\partial \alpha}, \mathbf{S}\right)}{h(\sigma)} - \frac{|\mathbf{S}|^2}{2(h(\sigma))^2}\left(\frac{1}{2} + \frac{\sigma}{2\sqrt{\sigma^2 + 4\delta^2}}\right)\frac{\partial \sigma}{\partial \alpha} \\
&= 2\eta\frac{\left(\frac{\partial \mathbf{S}}{\partial \alpha}, \mathbf{S}\right)}{|\mathbf{S}|^2} - \frac{\eta}{h(\sigma)}\frac{\sigma + \sqrt{\sigma^2 + 4\delta^2}}{2\sqrt{\sigma^2 + 4\delta^2}}\frac{\partial \sigma}{\partial \alpha} \\
&= 2\eta\frac{\left(\frac{\partial \mathbf{S}}{\partial \alpha}, \mathbf{S}\right)}{|\mathbf{S}|^2} - \frac{\eta}{h(\sigma)}\frac{h(\sigma)}{\sqrt{\sigma^2 + 4\delta^2}}\frac{\partial \sigma}{\partial \alpha} \\
&= 2\eta\left(\frac{\left(\frac{\partial \mathbf{S}}{\partial \alpha}, \mathbf{S}\right)}{|\mathbf{S}|^2} - \frac{\frac{\partial \sigma}{\partial \alpha}}{2\sqrt{\sigma^2 + 4\delta^2}}\right).
\end{aligned}$$

- Third we deduce the expression corresponding to the partial derivative of equation (A.3):

$$\frac{\partial \mathbf{S^0}}{\partial \alpha}(\mathbf{x}) = \mathbf{R^0}(\mathbf{A_P^0})^{-1}\frac{\partial \mathbf{A_P}}{\partial \alpha}(\mathbf{x})\mathbf{W_E^{-1}}, \tag{A.4}$$

where $\mathbf{A_P}$ is defined in (3.8) as

$$\mathbf{A_P}(x, y) = \begin{pmatrix} x_1 - x & x_2 - x \\ y_1 - y & y_2 - y \end{pmatrix}. \tag{A.5}$$

Then the partial derivatives are:

$$\frac{\partial \mathbf{A_P}}{\partial x}(x, y) = \begin{pmatrix} -1 & -1 \\ 0 & 0 \end{pmatrix}, \tag{A.6}$$

$$\frac{\partial \mathbf{A_P}}{\partial y}(x, y) = \begin{pmatrix} 0 & 0 \\ -1 & -1 \end{pmatrix}. \tag{A.7}$$

References

[Escobar 03] Escobar, J. M., E. Rodríguez, R. Montenegro, G. Montero, and J. M. González-Yuste, *Simultaneous untangling and smoothing of tetrahedral meshes*, Computer Methods in Applied Mechanics and Engineering (2003); 192 (25), 2775-2787.

[Escobar 06] Escobar, J. M., G. Montero, R. Montenegro, and E. Rodríguez, *An algebraic method for smoothing surface triangulations on a local parametric space*, Int. J. Numer. Meth. Engng (2006); 66:740-760.

[Field] Field, David A., *Quality measures for initial meshes*, Int. J. Numer. Meth. Engng. 47, 887-906 (2000).

[Giuliani] Giuliani, S., *An algorithm for continous rezoning of the hydrodynamic grid in arbitrary Lagrangian-Eulerian computer codes*, Nuclear Engineering and Design, Volume 72, Num. 2 (1982).

[Golberg] M. A. Golberg, *The Derivative of a Determinant*, Mathematical Association of America, The American Mathematical Monthly, Vol. 79, No. 10 (Dec., 1972), pp. 1124-1126.

[Herrman] L. R. Herrmann, *Laplacian-Isoparametric Grid generation scheme*, J. Engng. Mech. Div. 102, 749-756 (1976)

[Knupp 01] Knupp, Patrik M., *Algebraic mesh quality metrics*, SIAM J. Sci. Comput. (2001); Vol.23 N1,pp. 193-218.

[Knupp 03] Knupp, Patrik M., *Algebraic mesh quality metrics for unstructured initial meshes*, Finite Elements in Analysis and Design 39 (2003) 217-241.

[Knupp 03b] Knupp, Patrik M., *A method for hexaedral mesh shape optimization*, Int. J. Numer. Meth. Engng. 58:3119-332 (2003).

[Nocedal] Jorge Nocedal, Stephen J. Wright, *Numerical optimization*, Springer (2006).

[Roca-Sarrate] Xevi Roca, Josep Sarrate, *An automatic and general least-squares projection procedure for sweep meshing*, Engineering with Computers, DOI: 10.1007/s00366-009-0172-z.