POLITECNICO DI MILANO
Dipartimento di Matematica "F. Brioschi"
Ph.D. Course in Mathematical Models and Methods for Engineering
XXVI Cycle

Advanced Techniques for the Generation and the Adaptation of Complex Surface Meshes

Supervisor: Prof. Simona Perotto
Tutor: Prof. Luca Formaggia
Coordinator: Prof. Roberto Lucchetti

Ph.D. candidate:
Franco Dassi, 769134

Milan, June 30th, 2014

Ai miei nonni.

Contents

Introduction

The main goal of this thesis is to develop some new tools for and generating and optimizing computational grids in various scientific research fields, with a particular emphasis on surface grids.

The generation of the "best" mesh is not an easy task and, very often, it still represents an open issue. Indeed, the concept of "best mesh" cannot be easily generalized and it usually depends on the field of interest. For instance, in the numerical solution of partial differential equations (PDEs), it can be considered as "optimal" a mesh that minimizes the number of elements for a target accuracy, or, vice-versa, the mesh that, for a given number of elements, provides the most accurate approximations. In this context, mesh optimization is usually driven by either a-priori or a-posteriori error estimators. When dealing with surface grids, which may provide the boundary of a three-dimensional domain or the actual computational domain, we need also to preserve a sufficiently accurate representation of the geometry. It means that the error estimator should take into account the geometry of the surface itself. Thus, from the implementation point of view, we have also to ensure that the grid modifications induced by the mesh optimization procedure are consistent with the geometrical representation of the surface. For instance, a node movement procedure is demanded to preserve the points on the surface.

In the context of Computer Aided Drafting (CAD) softwares, a geometry is often provided via parametric patches. This kind of representation introduces other technical difficulties that we have tried to deal with. Moreover, in some peculiar settings, the surfaces may also represent internal interfaces. For example, in geological applications, surfaces represent the so-called horizons. They are generally obtained from seismic imaging data and they may be present holes or be only partially defined.

We have focused on surface meshes composed by triangular elements and we have essentially considered the following applications: statistical analysis of

large data set on complex geometries, mesh generation of CAD models and the generation of domains associated with geological seismic data set. Since we tackle different topics associated with surface triangular meshes, we devote the first two chapters to provide the essential background to better understand the theoretical results and the algorithms developed in the remaining part of the thesis. Then, each of the next chapters focuses on a specific content of interest and provides both the theory and a large variety of numerical examples.

All the proposed algorithms have been developed in a C++ library called MESHDOCTOR, that is a proprietary code of ENI company, developed during the project GeoMod. Now we detail the contents of each chapter.

Chapter 1 provides some basic concepts about mesh generation and adaptation. More precisely, we give an overview on the main relations between mesh and piecewise interpolation as well as between mesh and the solution of PDEs. Moreover, we introduce the basic mesh operations required to modify a generic triangular mesh, [22, 27], and some concepts about mesh metric [60, 62] for both a planar and non-planar configuration.

Chapter 2 gives an overview of the theory of PDEs on surfaces embedded in $\mathbb{R}^3$ and it introduces some error estimates,[31], and gradient recovery strategies for both planar and non-planar grids, [114, 115, 110].

In **Chapter 3**, we provide the theory driving the anisotropic surface mesh adaptation. We start from the proposal of an anisotropic interpolation error estimator for functions defined on surfaces. Moving from this new error estimator, we provide a novel anisotropic a-posteriori error estimator to control the energy norm for both the Laplace-Beltrami problem and a convection-diffusion problem defined on implicit surface. Finally, we extend the well-known Zienkiewicz and Zhu error estimator [114, 115] to PDEs defined on surfaces. Via these error estimators, we define a metric based adaptation procedure. The reliability of this mesh adaptation strategy is numerically checked on several test cases.

In **Chapter 4**, we propose a new method for re-meshing three-dimensional surfaces based on the idea of a higher dimensional embedding proposed in [14, 71, 75] to get an accurate approximation of the actual surface. In [75], B. Lévy and N. Bonneel propose a re-meshing procedure based on the computation of the Voronoi diagram in the embedded space, but, unfortunately, this method does not preserve sharp features. To overcome this drawback, we propose a new way to exploit the higher dimensional embedding. In more details, we directly optimize the triangular mesh in the embedded space in such a way that the triangles are as uniform as possible in $\mathbb{R}^6$. As provided in [14, 71, 75] the image of the resulting mesh uniform mesh in $\mathbb{R}^6$ will yield a curvature-adapted anisotropic surface mesh

in $\mathbb{R}^3$. The reliability and the robustness of the proposed re-meshing strategy is investigated with several complex examples.

In the first part of **Chapter 5**, we introduce the issue of simplifying a mesh, i.e., how to build a simpler mesh starting from an initial very complex one. In the second part of this chapter, we introduce a new simplification strategy that reduces the number of triangular elements taking into account both the geometrical approximation of the surface and the statistical properties associated with it. In particular, we focus on a medical application by analysing the thickness data of the brain cortex. The basic idea behind this method is to properly couple the proposed mesh simplification algorithm with the spatial regression analysis provided by B. Ettinger et al. in [35].

In the geological framework, quadrilateral and hexahedral meshes are basically exploited. In **Chapter 6**, we extend all the basic mesh operations tailored on quadrilateral and hexahedral meshes to triangular and tetrahedral meshes, to build a suitable geological domain moving from seismic imaging data. Since the intersection of surface meshes represents one of the most recurrent operations to get this computational domain, we present a new method to reduce the computational effort associated with such an operation. Moreover, we extend to surface triangular meshes some specific geological operations, e.g., to fill the data lack and we introduce the so-called "hard and soft" operation to deal with overlapping horizons.

Chapter 1

Mesh Generation and Adaptation

In this chapter we give an overview about mesh generation and adaptation. In particular, we focus on the relations between the mesh and the piecewise linear interpolation as well as the mesh and the solution of a Partial Differential Equation.

1.1 Mesh Classification

A mesh is a finite union of elements that covers an arbitrary computational domain. The elements may have different shapes: for instance quadrilaterals, triangles for a two-dimensional domains. In the three-dimensional space the volume domains are usually composed by hexahedra or tetrahedra.

We distinguish two different types of meshes: isotropic and anisotropic. In the former case, each mesh element has a regular shape, i.e., we require that the ratio of the element diameter and the diameter of the inscribed circle are bounded from above, for instance, a triangular isotropic mesh is composed by equilateral triangles. In the latter case, the elements may have an arbitrary shape and orientation and they may be arbitrarily elongated.

In this thesis we are dealing with triangular meshes composed by simplices that approximate a surface embedded in the three dimensional space.

Definition 1.1.1 *Given a positive integer k, a k-**simplex** is a k-dimensional polytope which is the convex hull of its $(k+1)$ vertices. Suppose the $(k+1)$ points $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^k$ to be linearly independent, which means that the vectors $\mathbf{x}_1 - \mathbf{x}_0, \ldots, \mathbf{x}_k - \mathbf{x}_0$ are linearly independent. Then, the simplex associated with*

the points $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k$ *is determined by the set*

$$C := \left\{ \theta_0 \mathbf{x}_0 + \theta_1 \mathbf{x}_1 + \ldots \theta_k \mathbf{x}_k \, : \, \theta_k \geq 0 \, , \, 0 \leq i \leq k \, , \sum_{i=0}^{k} \theta_i = 1 \right\} . \qquad (1.1)$$

Remark 1.1.1 *A **simplex** is a generalization of the notion of triangle or of tetrahedron to an arbitrary dimension.*

We consider a **2-simplex**, whose vertices $\mathbf{x}_0, \, \mathbf{x}_1, \, \mathbf{x}_2 \in \mathbb{R}^3$. In more in details, we will consider triangular surface anisotropic meshes.

1.1.1 Isotropy vs Anisotropy

Creating "good quality" mesh is the main issue of mesh generation and mesh improvement algorithms. In fact, interpolation procedures and finite element methods rely on meshes whose elements have right shapes and sizes. The accuracy or the speed of a numerical simulation may be corrupted by just a few "bad elements". In the last decades, understanding what "bad" and "good" elements means and how to get a "good quality" mesh has become a crucial issue. However, the knowledge of the actual relationship between mesh geometry and numerical accuracy remains very often incomplete, even in the simplest cases.

It is important to say that the "quality" of the mesh is strictly related to the application field. For instance, the interpolation accuracy is important in surface fitting. While the condition number of the global stiffness matrix is really important in a finite element analysis.

In both cases, moving from a mesh and from an unknown function f, we are usually interested in finding a piecewise approximation f_h of f. We define the **discretization error** as the difference between f and f_h, measured in an appropriate norm. This quantity will converge to zero as the discrete approximation f_h converges to the continuous function f. In [107], it is shown that the discretization error is affected **not only** by the size of the triangles: the shape and, in particular, the internal angles of the elements could drastically change the approximation of f_h.

When we interpolate a continuous function with a piecewise function f_h, there are two different types of interpolation error:

- $f - f_h$, the difference between the interpolated function and the exact function;

- $\nabla(f - f_h)$, the difference between the gradient of the interpolated function and the gradient of the exact function.

There is a large variety of applications, such as rendering, map-making, that require a good approximation also of the gradient of the interpolation function. If f is smooth enough, the interpolation error may be reduced by diminishing the size of the triangles. On the other hand, if the elements are not properly aligned with the gradient of the interpolating function f or if they have large angles, the error on the gradient may drastically increase, see [107].

In a finite element analysis, we are interested in solving a Partial Differential Equation (PDE) on a certain mesh. The discretization of the PDE leads to a linear system; in particular, the condition number of the stiffness matrix plays a key role in the reliability of the solution. Since the stiffness matrix depends on the discretization of the computational domain, the element shape has a strong influence on the condition number. Unlike the interpolation framework, in this case small angles should be avoided, while large angles represent a good choice.

In conclusion, we may state that:

- when we are dealing with interpolation, we should avoid large angles;
- the condition number of the stiffness matrix increases, when the mesh triangles have small angles.

Starting from these two observations, we may infer that a triangular mesh with not too big or too small angles could be the optimal for both the interpolation and the finite element framework. This conjecture is numerically proved in [107] and statistics based on experimental data show that meshes composed by nearly equilateral triangles provide better results.

Triangle Quality

To evaluate the quality of a triangular mesh, it is necessary to define some precise criteria. A mesh generation or a mesh optimization algorithm usually selects a single and easy-computable index to evaluate the "goodness" of a single element, i.e., to estimate how far is the triangle shape from the equilateral shape.

Definition 1.1.2 *Consider a triangle T, and let R, r be the radius of the circumscribed and the inscribed circle, respectively. We define the* ***aspect ratio****, Q, as*

$$Q(T) := \frac{2r}{R}\,. \tag{1.2}$$

Remark 1.1.2 *It is possible to compute the area of a triangle T moving from the lengths of its sides, a, b and c, as:*

$$A := \sqrt{p(p-a)(p-b)(p-c)}, \tag{1.3}$$

where p is the semi-perimeter of the triangle T. This is a well-known result of elementary geometry called ***Heron's Formula****.*

Proof. A corresponding proof may be found in [83].

□

Remark 1.1.3 *We may compute the radius of the circumscribed and of the inscribed circle by resorting to the measure of the triangle sides, of the semi-perimeter p and of the area A as:*

$$R := \frac{abc}{4A} \quad \textit{and} \quad r := \frac{A}{p}\,, \tag{1.4}$$

respectively.

Proof. This is a classical result of elementary geometry and the proof may be found in [83].

□

Remark 1.1.4 *The quantity Q can be computed via the lengths of the sides of the triangle T and the semi-perimeter p, as:*

$$Q = \frac{8(p-a)(p-b)(p-c)}{abc}\,. \tag{1.5}$$

Proof. Via Remark 1.1.2 and 1.1.3 we have the following chain of equalities:

$$\begin{aligned} Q &= \frac{2r}{R} = \frac{8A^2}{pabc} \\ &= \frac{8(p-a)(p-b)(p-c)}{abc}\,. \end{aligned}$$

□

For an equilateral triangle the index Q is equal to 1. Values of Q close to zero correspond to very distorted triangle that leads to a **not** accurate approximation of the interpolating function or to a very high condition number for the associated stiffness matrix. The particular case of $Q = 0$, corresponds to a **degenerate triangle**, i.e., a zero-area triangle.

Anisotropy

Despite the previous considerations, in some circumstances an **anisotropic** element, i.e., an element stretched and oriented in an appropriate direction, may be advantageous from the computational point of view. When the stretching and the orientation of the triangles is determined by the function we are interpolating, or by the solution of partial differential equation we are solving, it has been observed that the results are astonishingly better than the ones provided by an **isotropic** mesh.

This fact is well-established in finite element analysis and, in particular, it becomes more evident in fluid dynamics applications, when the meshes employed for practical computations are strongly anisotropic [103, 88].

Here we show an example to clarify this fact. Consider a function

$$f(x, y) = \tanh(40y), \tag{1.6}$$

defined on the domain $\Omega = (-1, 1) \times (-1, 1)$, see Figure 1.1. Consider $\epsilon > 0$ and the sub-domain $\Omega_1 = \{(x, y) \in \Omega : |y| < \epsilon\}$. In this region f presents a great variation along the $y-$axis direction, while it is constant along the $x-$axis.

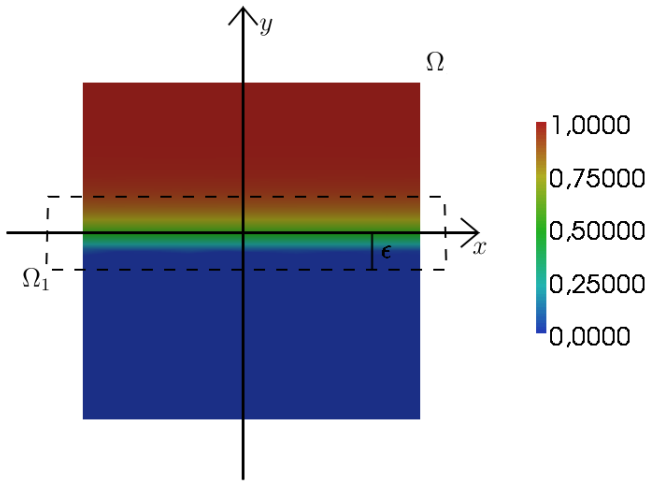


Figure 1.1: Function f, with the region Ω_1 highlighted.

Thus, we need a very high detail along the $y-$axis and a low detail along the $x-$axis to capture the jump of the function f in Ω_1, i.e., we need a very small mesh size along the $y-$axis, but on the $x-$direction it could be arbitrarily large.

When we are dealing with an isotropic mesh adaptation, if we need a small mesh size along one particular direction, we are forced to have a mesh with very small elements to recover equilateral triangles. So we have to fill Ω_1 with a lot of very small elements to capture the jump of f, see Figure 1.2(a).

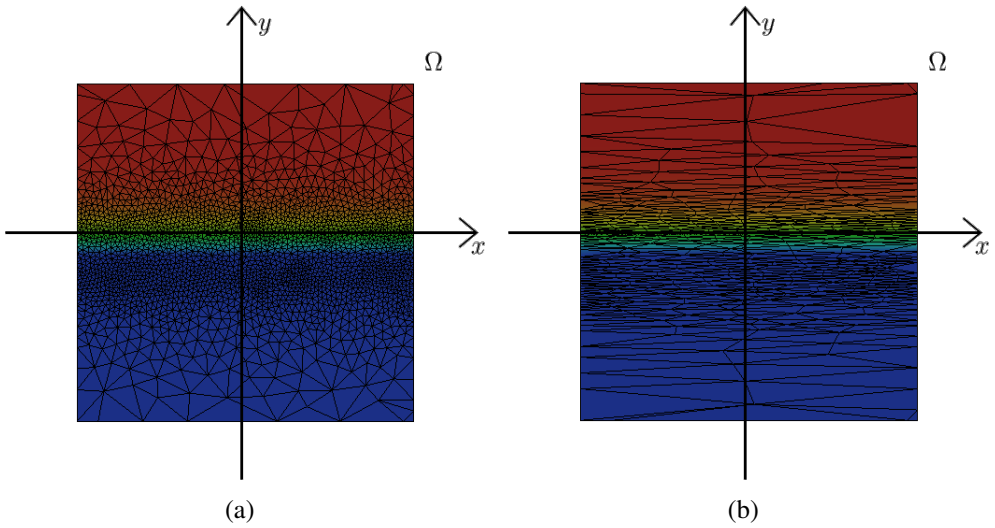


Figure 1.2: Interpolation of the function f with an isotropic mesh, (a) and with an anisotropic mesh, (b).

However, if we consider an anisotropic mesh adaptation, i.e., if we allow the triangles to stretch, we may set a very small size of the triangles in the $y-$direction and we are allowed to stretch them along the $x-$direction, see Figure 1.2(b).

Stretched triangles cover more area than equilateral elements, see Figure 1.3. This is the key concept about anisotropic mesh adaptation and this is the reason why this kind of meshes offers a better **error-vs-number of elements** behaviour than the isotropic one. In fact, if we fix about the same number of elements, we get a lower error, when we consider an anisotropic mesh. On the contrary, an anisotropic mesh provides about the same error of an isotropic mesh, but with less triangles.

Now, we numerically show this behaviour. Consider the exact function defined in Equation (1.6) and the piecewise linear approximation f_h. Then, we consider the L^2-norm of the discretization error defined as:

$$e_h := ||f - f_h||_{L^2(\Omega)} = \sqrt{\int_\Omega |f - f_h|^2}\,.$$

In Table 1.1 and 1.2 we compare the numerical results associated with an isotropic and an anisotropic mesh. Here, we appreciate that, if we fix about the same number of elements, we get a lower discretization error in the case of anisotropic mesh, see Table 1.1. Likewise, we get a similar accuracy with less elements in the anisotropic case, see Table 1.2.

	number of Elements	e_h
anisotropic	204	2.541e-02
isotropic	264	2.978e-01

Table 1.1: Numerical result fixed the number of elements.

	number of Elements	e_h
anisotropic	204	2.541e-02
isotropic	1574	2.784e-02

Table 1.2: Numerical result fixed the accuracy.

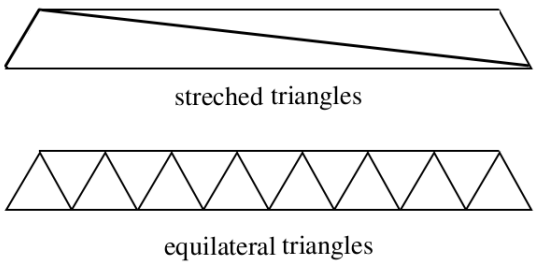


Figure 1.3: A couple of stretched triangles cover more area than a set of equilateral triangles

The better performances of anisotropic meshes are shown in a lot of works, see, e.g., [40, 41, 38, 36, 103, 88]

1.2 Operations on Triangular Meshes

The most well-known and commonly used local mesh operations for a triangular mesh are: edge flipping, edge splitting, edge contraction and node smoothing. These operations are widely discussed in the literature, see, e.g., [22, 27, 60, 62].

They modify the triangular elements to achieve a desired goal. For example, in [34], an edge flip algorithm is presented to recover the Delaunay criteria for a generic two dimensional planar mesh. In [47], the edge contraction operation is used to reduce the number of nodes in a surface mesh, and in [43] all these operations are applied to fit the geometry of a surface. Finally, in [38] the authors use these local operations to improve the accuracy of the solution of a Navier-Stokes equation.

1.2.1 Edge Flipping

Edge Flipping is the most efficient and effective local operation to modify a generic triangular mesh. An **edge-flip** on the edge **ab** removes the triangles **abc** and **bad**, and replace them with the triangles **cdb** and **dca**, see Figure 1.4. As a result, the edge **ab** is replaced by the edge **cd**.

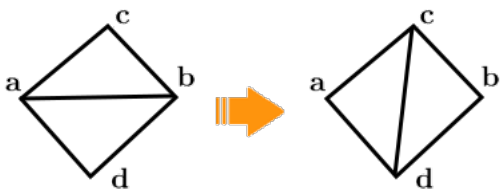


Figure 1.4: Flipping of the edge **ab**.

On a flat triangular mesh, it is not always possible to do an edge-flip. An edge **ab** is flippable if both these two conditions are verified:

(i) the edge **ab** belongs to the mesh;

(ii) any of the angles adjacent to the edge **ab** has to be obtuse, see Figure 1.5.

Remark 1.2.1 *Condition (i) will be clearer when we describe the Lawson's flip algorithm. This is a diffusive process. At the beginning, we put all the mesh edges in a stack and, once we flip an edge, we put all the edges linked to this edge in the*

*same stack, i.e., if we flip the edge **ab**, we successively put in the stack the edges **ac**, **cb**, **bd** and **ad**, see Figure 1.4. So, when we "pop" an edge from this stack, it is not a priori guaranteed that it still exists in the mesh.*

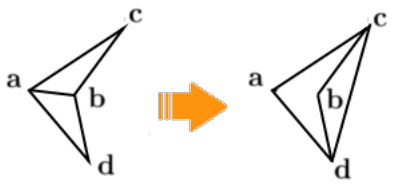


Figure 1.5: Example of an unflippable edge **ab** due to condition (ii).

On a triangular surface mesh the flipping operation becomes more complex due to the curvature of the surface. In particular, in the zones where the mesh presents ridges, the edge-flipping could bring to an incorrect approximation of the surface, see Figure 1.6. To overcome this issue, we decide to add another condition for the edge-flip validity:

(iii) the angle between the normals to triangles **abc** and **bad** has to be lower than a threshold value θ_{min}, in this work we set $\theta_{\text{min}} = 15°$.

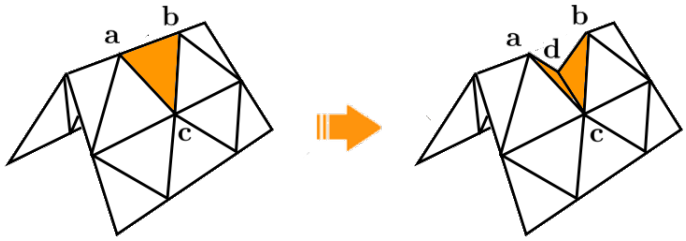


Figure 1.6: An example of edge-flip that creates an undesired approximation of a crest.

After checking the validity of the flipping of the edge **ab**, we proceed with the flip if the new edge **cd** matches a required criterion better than **ab**.

Lawson's Flip Algorithm

We have developed a novel edge-flip routine suited to deal with triangular surface meshes. It is the generalization for surface triangular meshes of the well-known Lawson's flip algorithm [74] for the construction of planar two-dimensional Delaunay triangulations. This new flip algorithm is provided in Algorithm 1. It uses two stacks S and S_1:

- S keeps all the edges to be checked and eventually flipped;
- S_1 is empty and it will keep edges which are not flippable.

Once a flip is done, the algorithm propagates to the neighbouring edges of the new edge **cd**, lines 8-10. The edges in S_1 are tried again if any flip has been done in the inner loop, lines 3-16. A key issue for the termination of this algorithm is to show that, once an edge is flipped, it will never be created again.

This routine is very efficient and it can be applied to all the edges of the surface mesh or even to a small subset of edges.

1.2.2 Edge Splitting

Mesh refinement has a major importance in mesh adaptation, due to its key role in increasing the resolution of the mesh in the zones of interest. In this kind of process a criterion is defined to locate a too long edge; then, the splitting procedure replaces the faces **abc** and **bad** with four faces **avc**,**vbc**,**bvd** and **vad**, where **v** is the middle point of **ab**, see Figure 1.7.

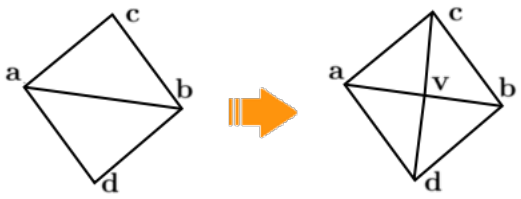


Figure 1.7: Splitting of the edge **ab**.

When we are dealing with a flat triangular mesh, the location of the new point **v** at the middle of the segment **ab** is always allowed. However, when we are dealing with a surface triangular mesh, the middle point of **ab** is not necessarily a good choice.

Algorithm 1 The edge-flip algorithm

FLIPEDGES(S, S_1)

Data: S is a stack of edges to be checked and flipped, S_1 is the stack empty on input.

while S is non-empty **do**
  $t = 0$;
  while S is non-empty **do**
    pop $\mathbf{ab}$ from S;
    if $\mathbf{cd}$ matches the criterion better than $\mathbf{ab}$ **then**
      if $\mathbf{ab}$ meets conditions (i), (ii) and (iii) **then**
        flip $\mathbf{ab}$ to $\mathbf{cd}$;
        for $\mathbf{xy} \in \{\mathbf{ac}, \mathbf{cb}, \mathbf{bd}, \mathbf{da}\}$ **do**
          push $\mathbf{xy}$ on S;
        end for
        $t = t + 1$;
      else
        push $\mathbf{ab}$ on S_1;
      end if
    end if
  end while
  if S_1 is non-empty and $t > 0$ **then**
    swap S and S_1;
  end if
end while

Consider a discrete surface Γ_h, a triangular approximation of a surface Γ, and suppose that we want to split the edge **ab**, see Figure 1.8 left. If we insert the point **v** in the middle of the edge **ab**, the resulting mesh does **not** fit the actual surface Γ, see Figure 1.8 middle. Thus, to improve the accuracy of the discretization of the surface Γ, we have to project **v** on Γ, as shown in Figure 1.8 right.

1.2.3 Edge Contraction

Mesh coarsening is another important aspect of a mesh adaptation procedure, aimed at decreasing the resolution of the triangular mesh in the zones where it is not necessary. This is the inverse operation of the edge splitting.

Also in this procedure a criterion is defined to locate a too short edge, **ab**, then,

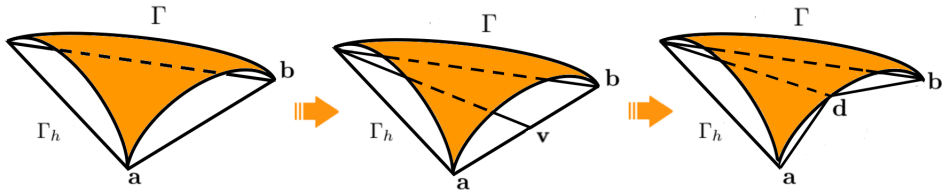


Figure 1.8: Splitting of the edge **ab** via the point **v** and projection of the added point on the surface Γ, point **d**.

the contraction of the edge **ab** removes **ab** together with the two triangles **abx** and **bay** and it mends the hole by gluing **xa** to **xb** and **ya** to **yb** as shown in Figure 1.9. Vertices **a** and **b** are made coincident to form a new vertex **c**. All the triangles that share the vertex **c** are new, while the other triangles stay the same.

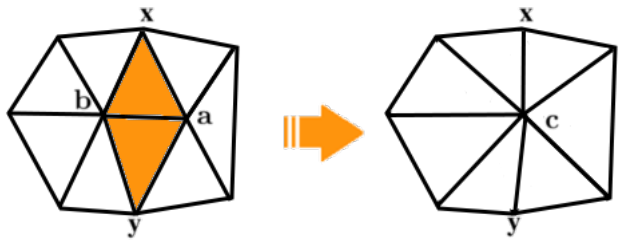


Figure 1.9: Collapsing of the edge **ab**.

This operation may lead to invalid topological configurations, such as overlapped triangles see the highlighted triangles in Figure 1.10 on the right. To avoid such situations, we may control the nodes connected to the end-points of the segment **ab**, [34]. Consider the segment **ab**, the nodes connected to **a** and the ones connected to **b**, separately. If these two sets have in common other points than the nodes of the triangles that share the segment **ab**, the contraction of this segment brings to a topological invalid configuration, see [34] for a rigorous demonstration. In Figure 1.10 we show an example: the sets of the nodes connected to **a** and **b** have in common **x**, **y** and **c**; nevertheless only **x** and **y** are nodes of the triangles that share the segment **ab**. So the contraction of the edge **ab** leads to an undesired topological configuration of the mesh.

There are different locations for the vertex **c**. We could place it at one of

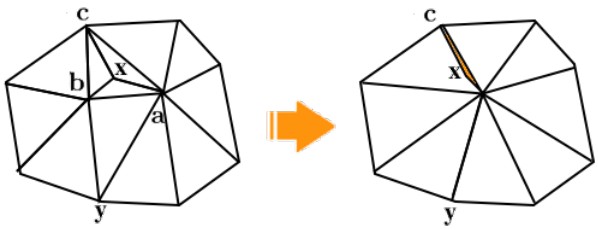


Figure 1.10: Example of topological invalid configuration due to the contraction of the edge **ab**. On the left we highlight two overlapped triangles.

the end-points of the segment **ab**, or at the middle point. When we are dealing with a flat triangular mesh, all these locations are reasonable. However, when the triangular mesh represents a surface embedded in the three-dimensional space, if we contract the edge into the middle point, we need to project this new vertex onto the real surface.

1.2.4 Node Smoothing

Smoothing is one of the classical methods to modify a mesh. This algorithm is in contrast with the ones described before, in fact it does **not** modify the topology of the grid, but it simply moves the nodes of the mesh in a new position.

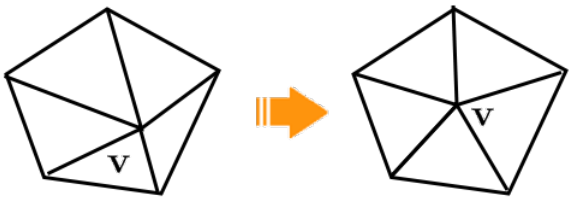


Figure 1.11: Smoothing of the node **v**.

The new position of the point **v** may lead to invalid configurations such as overlapped triangles, see Figure 1.12, so even in this case a check on the validity of the new mesh has to be performed.

Indeed, when we are dealing with a surface triangular meshes, once we have moved the point, we have to project it on the real surface, in order to fit the surface

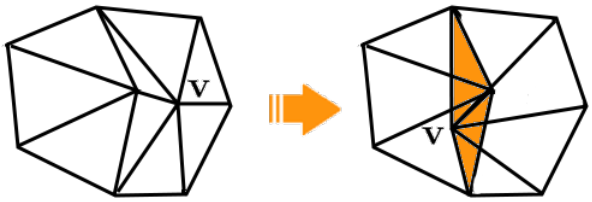


Figure 1.12: Example of a topological invalid configuration due to the smoothing of the node **v**. On the right it is highlighted the inverted triangles.

as well as possible.

1.3 Mesh Generation

Mesh generation is the practice of generating a polygonal or polyhedral mesh to approximate a domain.

In literature there are many algorithms to generate different kinds of meshes, we refer the reader to [100] for a deep survey. Since we are interested in triangular and tetrahedral meshes, we briefly give an overview of triangular and tetrahedral mesh generators. These two types of mesh generators are usually classified into two main categories:

- Delaunay mesh generators;
- advancing front mesh generators.

Delaunay Mesh Generators

This mesh generation technique produces a planar triangular or a volume tetrahedral mesh that satisfies the so-called **Delaunay Criterion**: all the circumcises of the triangular elements, or the circumsphere of the tetrahedral elements, do not contain any node of the mesh, [34]. In Figure 1.13 we show an example of a triangular planar mesh that satisfies such criterion.

There are different algorithms to get this kind of meshes and we refer the reader to [100] for a detailed overview of them. Here we recall “Triangle”, [106], and “Tetgen”, [108], that are the most popular triangular and tetrahedral Delaunay mesh generator.

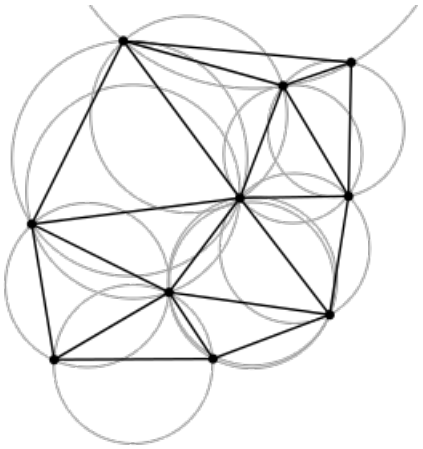


Figure 1.13: Example of Delaunay triangular mesh, we highlight in gray the circumcircles of each triangles and we observe that they do **not** contain the mesh nodes.

Advancing Front Mesh Generators

An advancing front mesh generation technique is quite different from a Delaunay mesh generation. In this case, the elements are constructed moving from the boundary of the domain. In Figure 1.14 we highlight with a dashed line the advancing front and we provide some steps of this mesh generation technique. One of the most popular advancing front mesh generation softwares is "Netgen" for both triangular and tetrahedral meshes, [101].

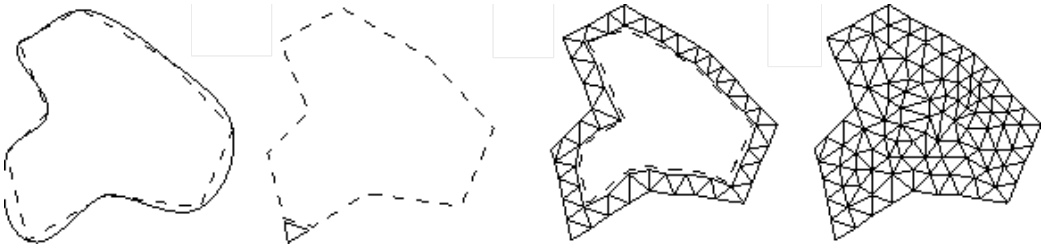


Figure 1.14: Example advancing front mesh generation, the solid line represents the boundary of the real domain, the dashed line is the advancing front.

1.4 Mesh Adaptation

Let us consider an initial mesh Ω_h. A mesh adaptation algorithm is a sequence of procedures applied to Ω_h to get a new mesh, Ω'_h, that satisfies a precise criterion. There are different ways to adapt a mesh. The difference among them consists in the operations that are applied and the criteria chosen to proceed with the adaptation.

In Section 1.2, we have already explained the main operations that could be applied to a triangular mesh. Here we focus on the different criteria to proceed with a generic mesh adaptation:

- adaptation driven by heuristic criteria;
- adaptation driven by a rigorous error estimator.

Adaptation driven by Heuristic Criteria

The idea behind these error estimators is to have a proper criterion to evaluate the accuracy of the discretization. This kind of techniques has been devised in the past in many papers, see, e.g., [15, 103, 55]. In the finite element framework, a typical methodology consists in estimating the Hessian or the gradient of the numerical solution and using this information to drive the mesh adaptation procedure.

A typical example of adaptation driven by heuristic criteria is the so-called **Zienkiewicz and Zhu** adaptation procedure, see, e.g., [89, 113]. In this case, the grid is adapted moving from the error associated with the finite element discretization, i.e., the H^1-semi-norm of the difference between the exact solution u and the discrete solution u_h. We refer to [113, 89, 114, 115] for an example of such error estimators in the case of planar triangular meshes.

Adaptation driven by Error Estimators

In this case adaptation is strongly linked to the problem at hand, typically the discretization of a PDE. Consider the discretization error, e_h. We aim at guaranteeing that the adapted grid Ω'_h is such that

$$e_h < \tau \,,$$

where τ is a suitable tolerance.

The basic idea behind this kind of adaptation is to find an estimate of e_h, i.e., a computable quantity S such that

$$e_h \leq S\,. \tag{1.7}$$

S is called **global error estimator** of the discretization error. To proceed with a mesh adaptation, the global error estimator S should satisfy the following properties:

- **computable**: we need to evaluate this quantity in a fast and straightforward way;
- **locality**: the estimate should be computable moving from quantities that depend only on a single element T of the mesh, or on a limited number of the elements around T, i.e,

  $$\eta_T = \sum_{T_i \in \omega_T} \eta_{T_i}\,, \tag{1.8}$$

  where ω_T is a set of triangular elements in the neighbourhood of T, η_T is called **local error estimator** that estimates the discretization error of the element T. Via Equation (1.8), we may define the the **total error estimator** as:

  $$S = \sum_{T \in \Omega_h} \eta_T\,, \tag{1.9}$$

  where Ω_h is the mesh.
- **reliability**: there exists a constant $C_1 \simeq 1$, such that

  $$C_1 e_h \leq S\,,$$

  this fact guarantees that, if $S \leq \tau$, then also $e_h \leq \tau$;
- **efficiency**: there exists a constant $C_2 \simeq 1$, such that

  $$S \leq C_2 e_h\,,$$

  this fact guarantees that, if $S \simeq \tau$, then also $e_h \simeq \tau$.

Remark 1.4.1 *The reliability and the efficiency are key properties of an error estimator, because they ensure the equivalence between the error estimator and the actual discretization error, i.e.,*

$$C_1 e_h \leq S \leq C_2 e_h\,.$$

Looking at the possible expression for the error estimator S, we may distinguish two different categories of error estimators:

- **a-priori error estimators**: in this case, S depends on the quantity we are approximating; for instance, in a finite element framework, when we approximate an exact function u with a piecewise polynomial function u_h, the quantities S and, consequently, η_T depend on the unknown solution u.
- **a-posteriori error estimators**: in this case, the quantity S depends on the discretization of the exact quantity we are approximating; for example, in a finite element framework the quantities S and η_T depend on the finite element approximation u_h which discretizes the exact solution u.

Moving from these estimates or, more precisely, considering the value of the local error estimator, Equation (1.8), it is possible to refine the mesh where η_T is greater than a fixed threshold and vice-versa. This kind of procedure is widely used in literature, and in [70, 4, 6, 79] we find examples of such error estimators.

In a finite element framework, these kinds of error estimators have received a lot of interest and different types of a-posteriori error estimators have been developed. We distinguish the following classes:

- residual-based error estimators;
- goal-oriented error estimators.

Residual-based error estimators

This is a typical kind of error estimators related to the approximation of a partial differential equation problem. Let us consider a standard Laplace problem defined in a two-dimensional domain, Ω,

$$\begin{cases} -\Delta u &= f & \text{in } \Omega\,, \\ &+ \text{ B.C.} & \end{cases}\,, \tag{1.10}$$

In such error estimators, we may distinguish two different contributions:

- the **internal residual**: the error related to the internal of an element mesh;
- the **jump residual**: the error due to the jump of the normal derivative of the discrete solution across a generic edge e of the mesh.

An example of this kind of error estimators may be found in [69, 7].

Goal-Oriented error estimators

In some applications, we might be interested in some quantities related to the physics of the problem at hand. For instance, one might be interested in quantities such as the concentrations around critical areas of the computational domain, or to control fluxes across sections of interest, etc.. In this case, the mesh is properly refined in order to reduce **only** the approximation of this particular quantity of interest and not to decrease the discretization error. In [91, 84], we have examples of this kind of error estimators.

1.5 Metric Based Adaptation

If we consider the error estimators described in Section 1.4, we may only proceed with a mark-refine adaptation, i.e., we may find only the triangular elements associated with a high discretization error and refine them. In general, the error estimator is often not enough to properly exploit all the mesh operations described in Section 1.2. To achieve this goal, one of the possible strategies is to build a suitable metric field that rigorously employs the information provided by the error estimator.

Before dealing with metric mesh adaptation, it can be useful to remind some basic aspects about the concepts of metric, [44].

Definition 1.5.1 *A **metric** on a set X is a function, called the distance function or simply distance, $d : X \times X \to \mathbb{R}^+ \cup \{0\}$. For all x, y and z in X, this function has to satisfy the following properties:*

- $d(x,\, y) \geq 0\,;$
- $d(x,\, y) = 0 \iff x = y\,;$
- $d(x,\, y) = d(y,\, x)\,;$
- $d(x,\, y) \leq d(x,\, z) + d(z,\, y)\,.$

Definition 1.5.2 *A **metric space** is an ordered pair (X, d), where X is a set and d is a metric.*

Remark 1.5.1 *In $\mathbb{R}^2$ the specification of a metric is equivalent to define two strictly positive scalar functions $\sigma_1 = \sigma_1(\mathbf{x})$, $\sigma_2 = \sigma_2(\mathbf{x})$ and two orthogonal unitary vector functions $\mathbf{u}_1 = \mathbf{u}_1(\mathbf{x})$, $\mathbf{u}_2 = \mathbf{u}_2(\mathbf{x})$. Then, we may define a metric d via the following metric tensor*

$$\boldsymbol{M}(\mathbf{x}) = \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 \end{pmatrix} \begin{pmatrix} 1/\sigma_1^2 & 0 \\ 0 & 1/\sigma_2^2 \end{pmatrix} \begin{pmatrix} \mathbf{u}_1^t \\ \mathbf{u}_2^t \end{pmatrix} . \tag{1.11}$$

The definition of such metric tensor may be extended to the generic space $\mathbb{R}^n$.

Once we have defined a metric space, we can compute the length of a generic segment. In $\mathbb{R}^2$, since the metric $\mathbf{M}(\mathbf{x})$ could vary in space, given two points $\mathbf{a}, \mathbf{b} \in \mathbb{R}^2$, the length of the segment $\mathbf{ab}$ depends on the selected metric field $\mathbf{M}(\mathbf{x})$. In particular, it can be expressed by the following integral form:

$$||\mathbf{ab}||_{\mathbf{M}} := \frac{1}{||\mathbf{ab}||} \int_{\mathbf{a}}^{\mathbf{b}} \sqrt{\mathbf{ab}^t \mathbf{M}(\mathbf{x}) \mathbf{ab}}\, d\mathbf{x} \tag{1.12}$$

where $\int_{\mathbf{a}}^{\mathbf{b}}$ is the line integral along the segment $\mathbf{ab}$ and $||\cdot||$ denotes the standard Euclidean norm.

Nevertheless, the length defined in Equation (1.12) is computationally expansive, so not practical. To approximate this value, we use the following expression:

$$||\mathbf{ab}||_{\mathbf{M}} := \max\left(\sqrt{\mathbf{ab}^t \mathbf{M}(\mathbf{a}) \mathbf{ab}}, \sqrt{\mathbf{ab}^t \mathbf{M}(\mathbf{b}) \mathbf{ab}} \right) , \tag{1.13}$$

where $\mathbf{M}(\mathbf{a})$ and $\mathbf{M}(\mathbf{b})$ are the metric matrices associated with the points $\mathbf{a}$ and $\mathbf{b}$, respectively. This hypothesis is conservative if the metric does not show great variations on $\mathbf{ab}$. This means that it tends to privilege the metric where the segment length is greater.

Remark 1.5.2 *Via the metric $\boldsymbol{M}(\mathbf{x})$ that varies with the position, it is possible to get an anisotropic mesh. In particular, consider a triangle T, whose vertexes are the points $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$. If we have*

$$||\boldsymbol{ab}||_{\boldsymbol{M}} = ||\boldsymbol{bc}||_{\boldsymbol{M}} = ||\boldsymbol{ca}||_{\boldsymbol{M}} = l\,,$$

the triangle is equilateral according to the metric, $\boldsymbol{M}(\mathbf{x})$, but since the metric field $\boldsymbol{M}$ distorts the distance, it will result stretched in the standard Euclidean space.

This remark explains one of the main ideas to get an anisotropic element. However, the principal issue is how to create a metric field $\mathbf{M}(\mathbf{x})$ that provides the appropriate orientation, shape and size of the triangles.

Once you get a metric field $\mathbf{M}(\mathbf{x})$, it is possible to proceed with two different kinds of mesh adaptation:

- re-meshing;
- local mesh adaptation.

Re-meshing

In this procedure the mesh is completely reconstructed. More precisely, consider a domain Ω and its discretization Ω_1; suppose that, on Ω_1, we are able to define a metric field $\mathbf{M}_1(\mathbf{x})$. Moving from this metric field, we can get a new discretization of Ω, where the elements are equilateral according to $\mathbf{M}_1(\mathbf{x})$, i.e., each element has its sides of the same length.

This could be an iterative procedure, i.e., we can get a finite sequence of adapted mesh Ω_1, Ω_2, Ω_3 ... Ω_n. In fact, once we get the new adapted mesh Ω_1, we may build a new metric field $\mathbf{M}_2(\mathbf{x})$ and, consequently, a new mesh Ω_2, and so on. This iterative procedure usually ends when a suitable criterion on the solution or on the number of elements of the mesh, is met.

In a finite element framework, this procedure is widely used to get an adapted anisotropic mesh. We refer to [41, 38] for examples on the two-dimensional planar domain meshes.

Local Mesh Adaptation

In this type of procedure, an initial mesh Ω_h is modified with the local mesh operations, introduced in Section 1.2, to get a new mesh Ω_h' such that all the elements are equilateral according to a metric field $\mathbf{M}(\mathbf{x})$.

There are many works in the literature that exploit this mesh adaptation procedure, for instance [76, 45] for the planar two-dimensional case.

Remark 1.5.3 *In this thesis we are dealing with surface meshes, so we have to consider a metric field $\boldsymbol{M}(\mathbf{x})$ defined on a surface. This metric is defined through two strictly positive scalar functions $\sigma_1 = \sigma_1(\mathbf{x})$, $\sigma_2 = \sigma_2(\mathbf{x})$ and three vector functions $\mathbf{u_1} = \mathbf{u_1}(\mathbf{x})$, $\mathbf{u_2} = \mathbf{u_2}(\mathbf{x})$ and $\mathbf{n} = \mathbf{n}(\mathbf{x})$, where*

$$\mathbf{u_1} \cdot \mathbf{u_2} = \mathbf{u_1} \cdot \mathbf{n} = \mathbf{u_2} \cdot \mathbf{n} = 0$$

and $\mathbf{n}$ *is the outward unit normal to the surface* Γ*. Then, we define a metric tensor* $\boldsymbol{M}(\mathbf{x})$ *at a generic point* $\mathbf{x} \in \Gamma$*, as*

$$\boldsymbol{M}(\mathbf{x}) = \begin{pmatrix} \mathbf{u_1} & \mathbf{u_2} & \mathbf{n} \end{pmatrix} \begin{pmatrix} 1/\sigma_1^2 & 0 & 0 \\ 0 & 1/\sigma_2^2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u_1}^t \\ \mathbf{u_2}^t \\ \mathbf{n}^t \end{pmatrix} . \tag{1.14}$$

Chapter 2

PDE on Surfaces

In this chapter we give a brief overview about Partial Differential Equations (PDEs) defined on surfaces embedded in $\mathbb{R}^3$. We highlight the principal issues on the definition of the domain. Then, we focus on the main partial differential operators used to define such PDEs. Finally, we recall some error estimates and gradient recovery strategies useful for the approximation of such PDEs.

2.1 Introduction on PDEs defined on Surfaces

The Partial Differential Equations defined on Surfaces have received a lot of interest in the last decades. There is a large variety of methods to define and solve this kind of PDEs. Among the first contributions, we cite the paper of G. Dziuik, [31]. In this work the author considers a piecewise polygonal surface and introduces a finite element space defined on this triangular surface mesh. Then, in [32, 33], the authors further analyse and extend this theory.

Moving from the idea in [9], another approach has been introduced by K. Deckelnick et al. in [23]. In this case, the surface is given by the zero level set of a signed distance function and the basic idea is to solve the partial differential equation on a narrow band around the surface.

In [86], Olshanskii et al. provided a new method to discretize a PDE on a surface based on an outer mesh. The main idea is to use the finite element space that is induced by triangulation of an outer domain to discretize the partial differential equation. In this case, the authors restrict the outer partial differential equation, instead of extending the PDE out off the surface, as in [31, 9]. This theory has been further studied and analysed in [85, 26] and it is particularly suitable for

problems assigned on a surface implicitly defined, i.e., coinciding with the zero level set of a signed distance function and when there exists a coupling between the PDE defined on the surface and the PDE defined on a fixed outer domain. In this framework, M. A. Olshanskii et al. use a finite element technique for the discretization of the outer domain. This setting immediately gives an easy to implement discretization method for the surface equation, so it does not require any additional surface elements.

In this thesis we use the theory provided by G. Dziuk in [31].

2.2 Definition of a Surface

A **surface** is defined as follows, [11]:

Definition 2.2.1 *A subset $\Gamma \subset \mathbb{R}^3$ is a differentiable surface if, for each point $\mathbf{x} \in \Gamma$ there exists an open subset $U \subset \mathbb{R}^3$ and a differentiable function $F : U \to \mathbb{R}$ such that $U \cap \Gamma = F^{-1}(0)$ and $\nabla F(\mathbf{x}) \neq 0\,, \forall \mathbf{x} \in U$.*

Remark 2.2.1 *The set Γ can be considered as the set of points such that $F(\mathbf{x}) = 0$, where F is a differentiable function.*

Remark 2.2.2 *The condition on the gradient, $\nabla F(\mathbf{x}) \neq 0\,, \forall \mathbf{x} \in U$, guarantees that the surface Γ is smooth at each point $\mathbf{x} \in \Gamma$.*

There are different ways to describe a surface Γ embedded in $\mathbb{R}^3$:

- as the graph of a function;
- in parametric form;
- via a global implicit form.

Graph of a Function

Consider an open set $A \subset \mathbb{R}^2$ and a differentiable function $f : A \to \mathbb{R}$. Via the function f, we define the surface Γ as follows:

$$\Gamma = \{(x, y, z) \in \mathbb{R}^3 \,:\, z = f(x, y)\}\,. \tag{2.1}$$

This way of representing a surface is very simple and straightforward, but it is not so general. In fact, there are surfaces that cannot be represented as a graph of a function f, for example a sphere or a torus.

Parametric Form

A surface can be built as the image of a injective differentiable function of two real variables in $\mathbb{R}^2$. Consider a function $\varphi : A \to \mathbb{R}^3$. Here A is a open set of $\mathbb{R}^2$, called parametric space. Moving from the function φ, we can get the coordinates $(x,\, y,\, z)$ of a generic point on the surface Γ as

$$\begin{aligned} x &= \varphi_1(u,\, v)\,, \\ y &= \varphi_2(u,\, v)\,, \\ z &= \varphi_3(u,\, v)\,, \end{aligned} \tag{2.2}$$

where $(u,\, v) \in A$.

This is the most natural way to define a surface and it provides simple expression to evaluate the length of arc on the surface, or the area of particular regions on the surface [93].

Unfortunately, given a generic surface Γ, it is not easy to find a function $\varphi(u,\, v)$ that satisfies Equation (2.2), [80]. To get a parametrization of a surface, it is sometimes necessary to split the surface Γ in to a finite set of sub-surfaces usually called patches, such that.

$$\Gamma = \bigcup_{i=1}^{n} \Gamma_i \quad \text{and} \quad \mathring{\Gamma}_i \cap \mathring{\Gamma}_j = \emptyset\,,$$

where $\mathring{\Gamma}_i$ denotes the internal part of the sub-surface Γ_i. Then, we parametrize each of these patches, separately, [8].

Global Implicit Form

A differentiable surface Γ can be represented as the zero level set of a **single** differentiable function $F : \mathbb{R}^3 \to \mathbb{R}$, i.e.,

$$\Gamma = \{\mathbf{x} \in \mathbb{R}^3 \,:\, F(\mathbf{x}) = 0\}\,. \tag{2.3}$$

This representation is really general and it allows us to represent a large variety of surfaces.

Moreover, suppose that we know only few points of an unknown surface Γ together with the normals to the surface at these points. In such a case, one of the possible ways to construct a function F, whose zero level set interpolates is to employ the so-called **radial basis functions**, [12, 78, 29].

In the framework of Partial Differential Equations defined on surfaces, the domain is represented via a global implicit form, [31]. In fact, this representation provides a unique and complete definition of the surface and offers a series of advantages in terms of the geometrical and the mesh adaptation point of view.

2.2.1 Geometric Considerations

We assume that F is a function of class C^2. Moving from F and its derivatives, it is possible to compute some useful geometrical quantities related to the geometry of the surface. In this subsection, we describe how to compute these geometrical quantities via its global implicit form, [105].

Given a point $\mathbf{x} \in \Gamma$, we are able to compute the outward unit normal as

$$\mathbf{n}(\mathbf{x}) := \frac{\nabla F(\mathbf{x})}{||\nabla F(\mathbf{x})||} \,. \tag{2.4}$$

Then, from the Hessian of F, $\mathbf{H}$, we can also obtain the principal curvatures of Γ. The problem of finding the principal curvatures at $\mathbf{x}$ can be formulated as follows: find the maximum and the minimum of the quadratic form

$$Q(\mathbf{w}) = \mathbf{w}^t \mathbf{K} \mathbf{w} \,, \quad \mathbf{w} \in S,$$

where

$$\mathbf{K} := -\frac{\mathbf{H}}{||\nabla F||} \,, \tag{2.5}$$

and

$$S := \{\mathbf{w} : \, ||\mathbf{w}|| = 1, \, \mathbf{w}^t \mathbf{n} = 0\} \,.$$

This problem can be reduced to find the eigenvalues of $\mathbf{K}$. In fact, it is possible to prove that the principal curvatures, k_{max} and k_{min}, are related to the maximum and the minimum eigenvalues of $\mathbf{K}$. Since the eigenvectors of $\mathbf{K}$ might not lie on the tangent plane to the surface Γ at $\mathbf{x}$, this result cannot be directly applied to find the curvatures. To overcome this problem, we work with instead of the matrix $\mathbf{K}$ with a matrix $\mathbf{G} \in \mathbb{R}^{3\times 3}$, that has the following properties:

i) $\mathbf{n}$ is the eigenvector of $\mathbf{G}$ with null eigenvalue;

ii) $\mathbf{w}^t \mathbf{K} \mathbf{w} = \mathbf{w}^t \mathbf{G} \mathbf{w} \quad \forall \mathbf{w} \in S$.

In this way, the searching of the maximum of the quadratic form $\mathbf{G}$ permits to find the maximum of $\mathbf{K}$ on the tangent plane to Γ at $\mathbf{x}$.

Proposition 2.2.1 *A matrix* $\boldsymbol{G}$ *that satisfies the properties, i) and ii) is*

$$\boldsymbol{G} := \boldsymbol{P}\boldsymbol{K}\,, \tag{2.6}$$

with

$$\boldsymbol{P} := \boldsymbol{I} - \mathbf{n} \otimes \mathbf{n}\,, \tag{2.7}$$

$\boldsymbol{K}$ *defined in Equation (2.5),* $\boldsymbol{I}$ *the identity matrix,* $\mathbf{n}$ *is the outward unit normal to the surface* Γ *at* $\mathbf{x}$*, and* $\otimes$ *the standard tensor product between vectors.*

Proof. Before dealing with the proof, we recall that

$$\mathbf{n} \otimes \mathbf{n} = \mathbf{n}\mathbf{n}^t \quad \text{and} \quad \mathbf{n}^t\mathbf{n} = 1\,.$$

Now we can proceed with the proof of property i):

$$\begin{aligned} \mathbf{n}^t\mathbf{G}\mathbf{n} &= \mathbf{n}^t\mathbf{P}\mathbf{K}\mathbf{n} \\ &= \mathbf{n}^t(\mathbf{I} - \mathbf{n} \otimes \mathbf{n})\mathbf{K}\mathbf{n} \\ &= \mathbf{n}^t\mathbf{K}\mathbf{n} - \mathbf{n}^t\mathbf{n}\mathbf{n}^t\mathbf{K}\mathbf{n} \\ &= \mathbf{n}^t\mathbf{K}\mathbf{n} - \mathbf{n}^t\mathbf{K}\mathbf{n} = 0\,. \end{aligned}$$

The proof of ii) can be obtained by the following chain of equalities. We consider a vector $\mathbf{w} \in S$, i.e. such that $\mathbf{w}^t\mathbf{n} = 0$, then, we have

$$\begin{aligned} \mathbf{w}^t\mathbf{G}\mathbf{w} &= \mathbf{w}^t\mathbf{P}\mathbf{K}\mathbf{w} \\ &= \mathbf{w}^t(\mathbf{I} - \mathbf{n} \otimes \mathbf{n})\mathbf{K}\mathbf{w} \\ &= \mathbf{w}^t\mathbf{K}\mathbf{w} - \mathbf{w}^t\mathbf{n}\mathbf{n}^t\mathbf{K}\mathbf{w} \\ &= \mathbf{w}^t\mathbf{K}\mathbf{w} - 0 = \mathbf{w}^t\mathbf{K}\mathbf{w}\,. \end{aligned}$$

□

Once we get the eigenvalues of $\mathbf{G}$, we can compute the principal curvatures of Γ at the point $\mathbf{x}$. More precisely, let λ_1, λ_2 and 0 be the eigenvalues of $\mathbf{G}$ and $|\lambda_1| > |\lambda_2|$; we have:

$$k_{\text{max}} = |\lambda_1| \quad \text{and} \quad k_{\text{min}} = |\lambda_2|\,.$$

2.2.2 Considerations on Mesh Adaptation

As seen in the previous chapter, there are some mesh operations that move or add points on the mesh. In such cases, it is crucial to locate the point on the actual domain. On the contrary, we are dealing with mesh adaptation of flat two-dimensional domains, this is not an issue. In fact, moving from the points of the mesh, we can easily find a new point that lies on the same plane. However, when the computational domain is **not** planar, we need a way to obtain a point that lies on the actual curved domain.

Fortunately, the global implicit form offers robust algorithms able to project a point onto the surface that the implicit form represents. One of the most popular is the one proposed by E. Hartmann [58]. The basic idea of this algorithm is the combination of calculating foot points on tangent planes and approximating foot points on tangent parabolas. To achieve this goal, the algorithm requires only the first order derivatives and the employment of the gradient method to move the point onto the surface along the steepest direction.

2.3 Finite Elements on Surfaces

In the framework of Partial Differential Equations defined on surfaces, we have to evaluate the derivatives of a function defined on a surface. As a consequence, to get the gradient of such a function, we **cannot** directly apply the standard differential operators, see [31].

Here, we recall the theory provided by G. Dziuk in [31] to overcome this issue. Let us consider a connected C^2 compact two-dimensional surface Γ embedded in $\mathbb{R}^3$. In particular, we assume that Γ coincides with the zero level set of a signed distance function $d : \mathbb{R}^3 \to \mathbb{R}$, such that

$$|d(\mathbf{x})| = dist(\mathbf{x}, \Gamma)\,, \quad \forall \mathbf{x} \in \mathbb{R}^3\,.$$

Moreover, if Γ is a closed surface, we assume $d < 0$ inside Γ, while $d > 0$ outside Γ. On the contrary, if Γ is an open surface, we assume $\partial\Gamma$ to be piecewise regular. Then, we consider a shell of width $\delta > 0$ about Γ,

$$U_\delta = \{\mathbf{x} \in \mathbb{R}^3 : |d(\mathbf{x})| < \delta\}\,.$$

The thickness δ has to be sufficiently small to guarantee the uniqueness of the decomposition

$$\mathbf{x} = a(\mathbf{x}) + d(\mathbf{x})\mathbf{n}(\mathbf{x}) \quad \forall \mathbf{x} \in U_\delta, \tag{2.8}$$

with $a : U_\delta \to \Gamma$ the orthogonal projection operator onto Γ and $\mathbf{n}(\mathbf{x}) = \nabla d/||\nabla d||$. The admissible values for δ depend on the principal curvatures of the surface. In particular, if we choose

$$\delta \leq \frac{1}{\max_{i=1,2} ||k_i||_{L^\infty(\Gamma)}}$$

where k_i are the principal curvatures of the surface Γ, the uniqueness of a is guaranteed, [25, 51].

The projection a is instrumental to extend the definition of a generic function f assigned on Γ to the whole shell U_δ. In fact, we define the extended function $f^E : U_\delta \to \mathbb{R}$ as

$$f^E(\mathbf{x}) := f(a(\mathbf{x})) \quad \forall \mathbf{x} \in U_\delta \,, \tag{2.9}$$

see Figure 2.1.

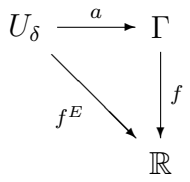


Figure 2.1: Diagram of the function f^E.

Remark 2.3.1 *f^E can be viewed as an extension along wires of the function f.*

Remark 2.3.2 *Since the function f^E is defined in the shell U_δ, the classical differential operators are well-defined on f^E.*

Moving from this projection operator, we can evaluate some useful quantities defined on surfaces parallel to Γ, i.e., Γ_1, Γ_2,... Γ_n, all included in U_δ, see Figure 2.2.

Remark 2.3.3 *For a generic point $\mathbf{x} \in U_\delta$, Equation (2.4) provides the outward unit normal to any surfaces parallel to Γ.*

Remark 2.3.4 *For a generic point $\mathbf{x} \in U_\delta$, it is possible to compute the curvature of the parallel surfaces, Γ_i, via the formula*

$$k_i(\mathbf{x}) = \frac{k_i(a(\mathbf{x}))}{1 + d(\mathbf{x})k_i(a(\mathbf{x}))} \,, \tag{2.10}$$

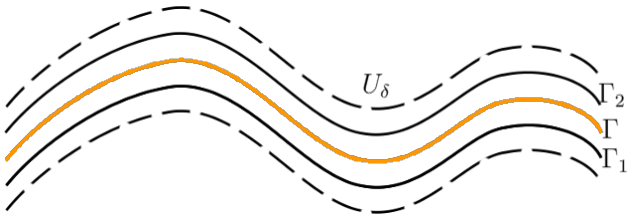


Figure 2.2: Surfaces parallel to Γ included in U_δ.

where $i = 1, 2$, while k_1 and k_2 denote the maximum and minimum curvature, respectively.

Proof. This is a well-known result and it is proved in [51].

Now, we show how we can recover the derivative of $f : \Gamma \to \mathbb{R}$ moving from $f^E : \Gamma_h \to \mathbb{R}$ and the classical differential operators.

Definition 2.3.1 *Given a function $f : \Gamma \to \mathbb{R}$, we define the **tangential gradient** $\nabla_\Gamma f$ as*

$$\nabla_\Gamma f := \boldsymbol{P} \nabla f^E \,, \tag{2.11}$$

with ∇ the standard gradient operator in $\mathbb{R}^3$,

$$\boldsymbol{P} = \boldsymbol{I} - \mathbf{n} \otimes \mathbf{n} \,, \tag{2.12}$$

and where $\boldsymbol{I}$ is the identity matrix, $\mathbf{n}$ is the outward unit normal to the surface Γ at $\mathbf{x}$ and $\otimes$ is the standard tensor product between vectors, as in Proposition 2.2.1.

Remark 2.3.5 *Thanks to definition (2.9), the tangential gradient $\nabla_\Gamma f$ depends only on the values of f on Γ even if Equation (2.11) involves the shell U_δ. In fact, we evaluate the usual gradient operator on f^E, but, via the matrix $\boldsymbol{P}$, we lose the component along the normal direction $\mathbf{n}$.*

Proof. We consider a function $f : \Gamma \to \mathbb{R}$ and its extension f^E defined by Equation (2.9). We consider the component $\nabla_\Gamma f$ along the normal $\mathbf{n}$ via the following scalar product

$$\begin{aligned} \nabla_\Gamma f^t \mathbf{n} &= (\mathbf{P} \nabla f^E)^t \mathbf{n} \\ &= ((\mathbf{I} - \mathbf{n} \otimes \mathbf{n}) \nabla f^E)^t \mathbf{n} \end{aligned}$$

$$
\begin{aligned}
&= (\nabla f^E - \mathbf{n}\mathbf{n}^t \nabla f^E)^t \mathbf{n} \\
&= (\nabla f^E)^t \mathbf{n} - (\nabla f^E)^t \mathbf{n}^t \mathbf{n}\mathbf{n} \\
&= (\nabla f^E)^t \mathbf{n} - (\nabla f^E)^t \mathbf{n} = 0
\end{aligned}
$$

□

Definition 2.3.2 *Given a function $\Psi : \Gamma \to \mathbb{R}^3$, we define the **tangential divergence** as*

$$\nabla_\Gamma \cdot \Psi := \boldsymbol{P} : \nabla \Psi^E \,, \tag{2.13}$$

where $:$ is the classical scalar product between matrices.

Definition 2.3.3 *Given a function $u : \Gamma \to \mathbb{R}$, we define the **Laplace-Beltrami operator** as*

$$\nabla_\Gamma u := \nabla_\Gamma \cdot (\nabla_\Gamma u) \,. \tag{2.14}$$

Remark 2.3.6 *The Laplace-Beltrami operator is the extension to non-planar domains of the classical Laplace operator defined for planar two-dimensional domain.*

Remark 2.3.7 *It is possible to generalize this theory by removing the assumption that Γ coincides with the zero level set of a signed distance function d. In particular, we can replace d with a sufficiently regular function F, see Definition 2.2.1, that represents the surface Γ as a zero level set, i.e.,*

$$\Gamma = \{\mathbf{x} \in \mathbb{R}^3 \,:\, F(\mathbf{x}) = 0\} \,.$$

2.3.1 The Laplace-Beltrami Problem

Continuous Formulation

Moving from the previous definitions, we formulate the so-called Laplace-Beltrami problem. Given a connected C^2 compact two-dimensional surface Γ embedded in $\mathbb{R}^3$, with $|\partial\Gamma| \neq \emptyset$, and a function $f : \Gamma \to \mathbb{R}$, the Laplace-Beltrami problem is: find u such that

$$\begin{cases} -\Delta_\Gamma u &= f \quad \text{on } \Gamma \,, \\ u &= 0 \quad \text{on } \partial\Gamma \,. \end{cases} \tag{2.15}$$

Now we introduce the weak formulation of this problem. We consider these functional spaces:

$$L^1(\Gamma) \;:=\; \left\{ \psi : \int_\Gamma |\psi| \, d\sigma < +\infty \right\},$$

$$
\begin{aligned}
L^2(\Gamma) &:= \left\{ \psi : \int_\Gamma |\psi|^2 \, d\sigma < +\infty \right\}, \\
H^1(\Gamma) &:= \left\{ \psi : \ \psi \in L^2(\Gamma), \nabla_\Gamma \psi \in [L^2(\Gamma)]^3 \right\}, \\
H_0^1(\Gamma) &:= \left\{ \psi \in H^1(\Gamma) : \ \psi|_{\partial\Gamma} = 0 \right\}.
\end{aligned}
$$

In these spaces we define the following norms

$$
\begin{aligned}
||\psi||_{L^1(\Gamma)} &:= \int_\Gamma |\psi| d\sigma \,, \\
||\psi||^2_{L^2(\Gamma)} &:= \int_\Gamma |\psi|^2 d\sigma \,, \\
||\psi||^2_{H^1(\Gamma)} &:= ||\psi||^2_{L^2(\Gamma)} + ||\nabla_\Gamma \psi||^2_{L^2(\Gamma)} \,.
\end{aligned}
$$

Remark 2.3.8 *For the functional Sobolev spaces defined before, the same embedding rules of the classical Sobolev spaces hold.*

Theorem 2.3.1 *Given a $n-$dimensional surface Γ and a function $\boldsymbol{v} : \Gamma \to \mathbb{R}^{n+1}$, the following equality holds*

$$
\int_\Gamma \nabla_\Gamma \cdot \boldsymbol{v} \, d\sigma = \int_{\partial\Gamma} \boldsymbol{v} \cdot \mathbf{n} \, d\gamma - \int_\Gamma \mathcal{K} \boldsymbol{v} \cdot \mathbf{n} \, d\sigma \,, \tag{2.16}
$$

where $\mathbf{n}$ is the outward unit normal to $\partial\Gamma$, tangent to Γ and $\mathcal{K} := \nabla_\Gamma \mathbf{n}$, is the mean curvature of Γ. This result is known as ***divergence theorem****.*

Then, a weak formulation of Problem (2.15) is formally obtained by multiplying the first equation in Equation (2.15) by $\phi \in H_0^1(\Gamma)$ and by applying Theorem 2.3.1.

$$
\begin{aligned}
\int_\Gamma f \, \phi \, d\sigma &= -\int_\Gamma \Delta_\Gamma u \, \phi \, d\sigma \\
&= \int_\Gamma \nabla_\Gamma u \cdot \nabla_\Gamma \phi \, d\sigma - \int_{\partial\Gamma} \nabla_\Gamma u \cdot \mathbf{n} \, \phi \, d\sigma + \int_\Gamma \mathcal{K} \nabla_\Gamma u \cdot \mathbf{n} \, \phi \, d\sigma \\
&= \int_\Gamma \nabla_\Gamma u \cdot \nabla_\Gamma \phi \, d\sigma \,,
\end{aligned}
$$

to simplify the previous expression we have used both the condition, $\phi = 0$ on $\partial\Gamma$, and $\nabla_\Gamma u \cdot \mathbf{n} = 0$. So, the weak formulation of the problem defined by Equation (2.15) is

Problem 2.3.1 *Given a function $f \in L^2(\Gamma)$, find $u \in H_0^1(\Gamma)$ such that:*

$$\mathcal{A}(u\,,\phi) = \mathcal{B}(\phi)\,, \qquad \forall \phi \in H_0^1(\Gamma)\,, \tag{2.17}$$

where $\mathcal{A} : H_0^1(\Gamma) \times H_0^1(\Gamma) \to \mathbb{R}$ and $\mathcal{B} : H_0^1(\Gamma) \to \mathbb{R}$ are a bilinear and linear form, respectively, defined as

$$\mathcal{A}(u\,,\phi) := \int_\Gamma \nabla_\Gamma u \cdot \nabla_\Gamma \phi \, d\sigma \quad \text{and} \quad \mathcal{B}(\phi) := \int_\Gamma f\phi \, d\sigma\,. \tag{2.18}$$

Now, we give an important result about the existence, uniqueness and stability of Problem 2.3.1. The proof of this result can be found in [31].

Theorem 2.3.2 *For each function $f \in L^2(\Gamma)$, there exists a unique weak solution of Problem 2.3.1, with $u \in H_0^1(\Gamma)$. Then, the following estimate holds*

$$||u||_{H^2(\Gamma)} \leq C||f||_{L^2(\Gamma)} \tag{2.19}$$

Remark 2.3.9 *A similar result holds in the case of an open surface Γ without boundary, i.e., for $|\partial\Gamma| = 0$, but, in this particular case we have to consider only the function $f \in L^2(\Gamma)$ such that*

$$\int_\Gamma f \, d\sigma = 0\,. \tag{2.20}$$

Discrete Formulation

The principal issue for the numerical approximation of this type of Partial Differential Equations is that the involved functions are defined on a continuous surface Γ. Unlike the classical finite element framework for a planar domain, in this case the computational domain $\Gamma_h \not\subset \Gamma$, so the most common finite element property are not verified, e.g., the well-known Galerkin orthogonality does not hold.

We follow the theory provided by G. Dziuk in [31] to discretize Problem 2.3.1. We approximate the smooth surface Γ by a triangular mesh Γ_h whose vertices are on Γ and such that $\Gamma_h \subset U_\delta$, see Figure 2.3.

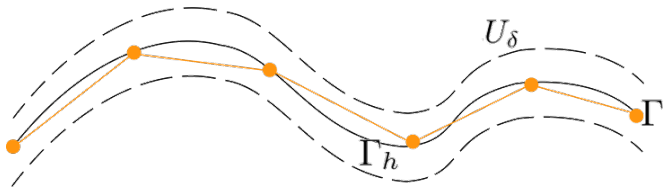


Figure 2.3: Section view of the surface Γ (solid line), of the shell U_δ (dashed line) and of the polyhedron surface Γ_h (line with markers).

We define the following finite element space over the domain Γ_h

$$V_h := \{v_h \in C^0(\Gamma_h) : v_h|_T \in \mathbb{P}^1(T) \quad \forall T \in \Gamma_h\}\,, \tag{2.21}$$

where we have denoted by $\mathbb{P}^1$ the space of the polynomials less than or equal to 1. On this discrete domain we define the following functional spaces

$$\begin{aligned}
L^1(\Gamma_h) &:= \left\{\psi_h : \int_{\Gamma_h} |\psi_h|\, d\sigma_h < +\infty\right\}, \\
L^2(\Gamma_h) &:= \left\{\psi_h : \int_{\Gamma_h} |\psi_h|^2\, d\sigma_h < +\infty\right\}, \\
H^1(\Gamma_h) &:= \left\{\psi_h : \ \psi \in L^2(\Gamma_h), \nabla_{\Gamma_h}\psi \in [L^2(\Gamma_h)]^3\right\}, \\
H_0^1(\Gamma_h) &:= \left\{\psi_h \in H^1(\Gamma_h) : \ \psi_h|_{\partial\Gamma_h} = 0\right\}.
\end{aligned}$$

To discretize the function f, we use the function $f_h : \Gamma_h \to \mathbb{R}$ defined as

$$f_h(\mathbf{x}) := f(a(\mathbf{x})) \quad \forall \mathbf{x} \in \Gamma_h\,. \tag{2.22}$$

Now we have all the ingredients to state the discrete formulation of Problem 2.3.1:

Problem 2.3.2 *Given a function $f_h \in L^2(\Gamma_h)$, find $u_h \in H_0^1(\Gamma_h)$ such that:*

$$\mathcal{A}_h(u_h\,, \phi_h) = \mathcal{B}_h(\phi_h)\,, \qquad \forall \phi \in H_0^1(\Gamma_h)\,, \tag{2.23}$$

where $\mathcal{A}_h : H_0^1(\Gamma_h) \times H_0^1(\Gamma_h) \to \mathbb{R}$ and $\mathcal{B}_h : H_0^1(\Gamma_h) \to \mathbb{R}$ are a bilinear and linear form, respectively, defined in such a way:

$$\mathcal{A}_h(u_h\,, \phi_h) := \int_{\Gamma_h} \nabla_{\Gamma_h} u_h \cdot \nabla_{\Gamma_h} \phi_h \, d\sigma_h \quad \textit{and} \quad \mathcal{B}_h(\phi_h) := \int_{\Gamma_h} f_h \phi_h \, d\sigma_h\,. \tag{2.24}$$

Remark 2.3.10 *In the weak formulation of Problem 2.3.1, we have replaced the linear and bilinear form, i.e. in Problem 2.3.1 we have $\mathcal{A}$ and $\mathcal{B}$ defined on the actual domain Γ, see Equation (2.18), while in Problem 2.3.2 we have $\mathcal{A}_h$ and $\mathcal{B}_h$ defined on the discrete domain Γ_h, see Equation (2.24).*

Remark 2.3.11 *The existence and uniqueness results of Problem 2.3.1 may be extended to Problem 2.3.2. Even in the discrete problem, when we are dealing with a surface Γ without boundary, we require*

$$\int_{\Gamma_h} f_h \, d\sigma_h = 0 \,, \tag{2.25}$$

to guarantee the existence of the solution.

The finite element scheme defined in Equation (2.23), is similar to the one of the classical finite element analysis. In fact, since a generic function $v_h \in V_h$ is piecewise linear, it can be uniquely defined via the values that assumes at the nodes of the mesh Γ_h. Thus, a basis for the discrete space V_h is provided by the set of functions $\varphi_j(\mathbf{x})$ for $j = 1, 2, \ldots, N$, such that:

$$\varphi_j(\mathbf{x}_i) = \begin{cases} 1 & \text{if} \quad j = i \\ 0 & \text{if} \quad j \neq i \end{cases} \qquad i, j = 1, 2, \ldots N, \tag{2.26}$$

where N is the number of nodes of the mesh Γ_h and essential boundary conditions are not assigned. Thus, a generic function $u_h \in V_h$ can be written as a linear combination of the basis functions $\varphi_j(\mathbf{x})$ as

$$u_h(\mathbf{x}) = \sum_{j=1}^{N} u_j \, \varphi_j(\mathbf{x}) \,, \tag{2.27}$$

where $u_j \; \forall j = 1, 2, \ldots, N$, is the value of the function u_h at the node $\mathbf{x}_j$. At this level, we remark that the finite element scheme has exactly the same structure of a classical finite element method for a planar problem. In fact, if we choose the test function $\varphi_i(\mathbf{x})$ in Equation (2.23) coinciding with the function $\varphi_i(\mathbf{x})$, we have

$$\sum_{j=1}^{N} u_j \int_{\Gamma_h} \nabla_{\Gamma_h} \varphi_j(\mathbf{x}) \cdot \nabla_{\Gamma_h} \varphi_i(\mathbf{x}) \, d\sigma_h = \int_{\Gamma_h} f_h \, \varphi_i(\mathbf{x}) \, d\sigma_h \quad i = 1, 2, \ldots N.$$

Thus, we can write the previous set of equations in a matrix form, as

$$\mathbf{A}\mathbf{u} = \mathbf{f} \,, \tag{2.28}$$

where we recognize the classical stiffness matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$, whose elements are

$$\mathbf{A}_{ij} := \int_{\Gamma_h} \nabla_{\Gamma_h}\varphi_j(\mathbf{x}) \cdot \nabla_{\Gamma_h}\varphi_i(\mathbf{x})\, d\sigma_h \,,$$

the vector of the unknown $\mathbf{u}$ and the right hand side $\mathbf{f}$, defined as

$$\mathbf{u}_j := u_h(\mathbf{x}_j) \quad \text{and} \quad \mathbf{f}_j := \int_{\Gamma_h} f_h\, \varphi_j(\mathbf{x})\, d\sigma_h \,,$$

respectively.

2.4 Convergence Estimate

In this section we formalize the comparison between the solution of Problem 2.3.1, u, with its finite element approximation, u_h, i.e., the solution of Problem 2.3.2. The principal issue of this comparison is that these two functions are defined on a different domain, since $u : \Gamma \to \mathbb{R}$ and $u_h : \Gamma_h \to \mathbb{R}$. To compare these two solutions, we use the projection operator a defined in Equation (2.8).

Consider a generic function v_h defined on Γ_h, we define its projection on Γ, $\tilde{v}_h : \Gamma \to \mathbb{R}$ as

$$v_h(\mathbf{x}) =: \tilde{v}_h(a(\mathbf{x})) \quad \forall \mathbf{x} \in \Gamma_h \,. \tag{2.29}$$

Then we can define $\tilde{v}_h^E(\mathbf{x}) := \tilde{v}_h(a(\mathbf{x}))$. The final result of this series of extension is to have extended the function v_h defined on Γ_h to the function $\tilde{v}_h^E$ defined on whole shell U_δ, see Figure 2.4.

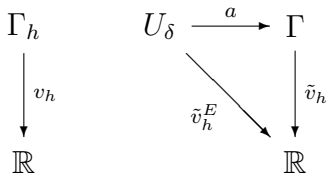


Figure 2.4: Diagram of the function v_h and $\tilde{v}_h^E$.

To simplify the notations, we refer to $\tilde{v}_h^E(\mathbf{x})$ as $v_h^E(\mathbf{x})$. Now, we estimate the convergence of the discrete solution for this kind of partial differential equations, Lemma 2.4.5. Before dealing with the proof of such an estimate, we need some useful properties of $\mathbf{P}$ and $\mathbf{H}$, Proposition 2.4.1. Moreover, we also need a relation between the operators ∇_Γ and ∇_{Γ_h}, Proposition 2.4.2 and an equivalence between the norm on Γ and Γ_h, Lemma 2.4.1.

Proposition 2.4.1 *We consider a signed distance function $d : \mathbb{R}^3 \to \mathbb{R}$, the normal $\mathbf{n}$ defined as in Equation (2.4), the Hessian $\boldsymbol{H}$ of d and the operator $\boldsymbol{P}$ defined as Equation (2.12). Then, the following relations holds:*

i) $\boldsymbol{H}\mathbf{n} = 0\,,$

ii) $\boldsymbol{P}\boldsymbol{P} = \boldsymbol{P}\,,$

iii) $\boldsymbol{H}\boldsymbol{P} = \boldsymbol{P}\boldsymbol{H} = \boldsymbol{H}\,.$

Proof. We recall that the Hessian, $\mathbf{H}$, of the signed distance function d that define the surface Γ is given by

$$\mathbf{H} = \begin{pmatrix} \frac{\partial^2 d}{\partial^2 x} & \frac{\partial^2 d}{\partial y \partial x} & \frac{\partial^2 d}{\partial z \partial x} \\ \frac{\partial^2 d}{\partial x \partial y} & \frac{\partial^2 d}{\partial^2 y} & \frac{\partial^2 d}{\partial z \partial y} \\ \frac{\partial^2 d}{\partial x \partial z} & \frac{\partial^2 d}{\partial y \partial z} & \frac{\partial^2 d}{\partial^2 z} \end{pmatrix}.$$

The first component of the product between $\mathbf{H}$ and $\mathbf{n}$ is

$$\frac{\partial^2 d}{\partial^2 x} n_x + \frac{\partial^2 d}{\partial y \partial x} n_y + \frac{\partial^2 d}{\partial z \partial x} n_z\,,$$

where n_x, n_y and n_z are the components of $\mathbf{n}$. We can rewrite this equation in such a way

$$\frac{\partial}{\partial x}\left(\frac{\partial d}{\partial x} n_x\right) + \frac{\partial}{\partial x}\left(\frac{\partial d}{\partial y} n_y\right) + \frac{\partial}{\partial x}\left(\frac{\partial d}{\partial z} n_z\right).$$

But if we use Equation (2.4), we have

$$\frac{\partial}{\partial x}\left(c\, n_x n_x\right) + \frac{\partial}{\partial x}\left(c\, n_y n_y\right) + \frac{\partial}{\partial x}\left(c\, n_z n_z\right),$$

where $c = ||\nabla d||$. Finally, we get

$$\frac{\partial}{\partial x}(c\,(\mathbf{n}_x \mathbf{n}_x + \mathbf{n}_y \mathbf{n}_y + \mathbf{n}_z \mathbf{n}_z)) = \frac{\partial}{\partial x}\left(c\,\right) = 0\,.$$

These computations can be repeated for each component of the vector $\mathbf{Hn}$, so we prove *i)*.

Now, we consider *ii)*. We recall that

$$\mathbf{P} \;=\; \mathbf{I} - \mathbf{n} \otimes \mathbf{n}$$

$$= \mathbf{I} - \mathbf{n}\mathbf{n}^t .$$

Thus we have

$$\begin{aligned}
\mathbf{PP} &= (\mathbf{I} - \mathbf{n} \otimes \mathbf{n})(\mathbf{I} - \mathbf{n} \otimes \mathbf{n}) \\
&= (\mathbf{I} - \mathbf{n}\mathbf{n}^t)(\mathbf{I} - \mathbf{n}\mathbf{n}^t) \\
&= \mathbf{I} - \mathbf{n}\mathbf{n}^t - \mathbf{n}\mathbf{n}^t + \mathbf{n}\mathbf{n}^t\mathbf{n}\mathbf{n}^t \\
&= \mathbf{I} - \mathbf{n}\mathbf{n}^t - \mathbf{n}\mathbf{n}^t + \mathbf{n}\mathbf{n}^t \\
&= \mathbf{I} - \mathbf{n}\mathbf{n}^t \\
&= \mathbf{P} .
\end{aligned}$$

Finally, via *i)*, we can prove the equalities *iii)*, i.e., we have

$$\begin{aligned}
\mathbf{HP} &= \mathbf{H}(\mathbf{I} - \mathbf{n} \otimes \mathbf{n}) \\
&= \mathbf{H}(\mathbf{I} - \mathbf{n}\mathbf{n}^t) \\
&= \mathbf{H} - \mathbf{H}\mathbf{n}\mathbf{n}^t \\
&= \mathbf{H} ,
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{PH} &= (\mathbf{I} - \mathbf{n} \otimes \mathbf{n})\mathbf{H} \\
&= (\mathbf{I} - \mathbf{n} \cdot \mathbf{n}^t)\mathbf{H} \\
&= \mathbf{H} - \mathbf{n}\mathbf{n}^t\mathbf{H} \\
&= \mathbf{H} - \mathbf{n}(\mathbf{H}\mathbf{n})^t \\
&= \mathbf{H} ,
\end{aligned}$$

respectively.

□

Proposition 2.4.2 *Consider a function $v_h : \Gamma_h \to \mathbb{R}$ and its projection on the surface Γ, $v_h^E : \Gamma \to \mathbb{R}$. Then, the following relation holds:*

$$\nabla_{\Gamma_h} v_h(\mathbf{x}) = \boldsymbol{P}_h(\mathbf{x})(\boldsymbol{I} - d\boldsymbol{H})(\mathbf{x})\nabla_\Gamma v_h^E(a(\mathbf{x})) , \tag{2.30}$$

where d is the signed distance function that defines the surface Γ, $\boldsymbol{H}$ is the Hessian of the function d, a is the projection operator, $\boldsymbol{I}$ is the identity matrix and we have defined

$$\boldsymbol{P}_h := \boldsymbol{I} - \mathbf{n}_h \otimes \mathbf{n}_h , \tag{2.31}$$

where $\mathbf{n}_h$ is the unit outward normal to the surface Γ_h.

Proof. If we resort to the chain rule of derivatives, we get

$$\nabla v_h^E(\mathbf{x}) = (\mathbf{P} - d\mathbf{H})(\mathbf{x})\nabla v_h^E(a(\mathbf{x}))\,, \tag{2.32}$$

Then, via the Proposition 2.4.1, we have

$$\begin{aligned}(\mathbf{P} - d\mathbf{H})(\mathbf{x})\nabla v_h^E(a(\mathbf{x})) &= (\mathbf{P} - d\mathbf{H})(\mathbf{x})\nabla v_h^E(a(\mathbf{x}))\\ &= (\mathbf{P} - d\mathbf{HP})(\mathbf{x})\nabla v_h^E(a(\mathbf{x}))\\ &= (\mathbf{I} - d\mathbf{H})(\mathbf{x})\mathbf{P}(\mathbf{x})\nabla v_h^E(a(\mathbf{x}))\\ &= (\mathbf{I} - d\mathbf{H})(\mathbf{x})\nabla_\Gamma v_h^E(a(\mathbf{x}))\,.\end{aligned}$$

Then if we consider $\mathbf{P}_h$ defined in Equation (2.31), we have

$$\begin{aligned}\nabla_{\Gamma_h} v_h(\mathbf{x}) &= \mathbf{P}_h(\mathbf{x})\nabla v_h^E(\mathbf{x})\\ &= \mathbf{P}_h(\mathbf{x})(\mathbf{I} - d\mathbf{H})(\mathbf{x})\nabla_\Gamma v_h^E(a(\mathbf{x}))\,,\end{aligned}$$

and this completes the proof.

□

Lemma 2.4.1 *Given a function $v_h : \Gamma_h \to \mathbb{R}$ and its projection on Γ, v_h^E, there exists a constant C such that, $\forall \mathbf{x} \in \Gamma_h$, the following relations hold:*

$$\begin{aligned}\frac{1}{C}||v_h||_{L^2(T)} \leq \quad ||v_h^E||_{L^2(T^E)} \quad &\leq C||v_h||_{L^2(T)}\,,\\ \frac{1}{C}||\nabla_{\Gamma_h} v_h||_{L^2(T)} \leq \; ||\nabla_\Gamma v_h^E||_{L^2(T^E)} \; &\leq C||\nabla_{\Gamma_h} v_h||_{L^2(T)}\,,\end{aligned}$$

here T is a triangle of the mesh Γ_h and

$$T^E = \left\{a(\mathbf{x}) \,:\, \mathbf{x} \in T\right\}. \tag{2.33}$$

Proof. The proof of these relations can be found in [31].

□

Remark 2.4.1 *The relations in Lemma 2.4.1 can be read as an equivalence relations between the norms defined on the actual surface Γ and its discrete approximation Γ_h.*

Lemma 2.4.2 *Consider a surface Γ defined as the zero level set of a signed distance function d and its discretization Γ_h. Given a point $\mathbf{x} \in \Gamma_h$, then the following relation holds*

$$\mu_h := \frac{d\sigma}{d\sigma_h} = (1 - d(\mathbf{x}))k_1(\mathbf{x})(1 - d(\mathbf{x}))k_2(\mathbf{x})(\mathbf{n}(\mathbf{x}))^t\mathbf{n}_h(\mathbf{x})\,, \tag{2.34}$$

where $\mathbf{n}_h(\mathbf{x})$ is the outward unit normal to the surface Γ_h, k_1 and k_2 are the principal curvatures defined in Remark 2.3.4, and $d\sigma$, $d\sigma_h$ are the infinitesimal portions of the surfaces Γ and Γ_h, respectively.

Proof. The proof of such relation can be found in [25].

□

Lemma 2.4.3 *We consider a continuous surface Γ and its discretization Γ_h. We suppose that Γ is the zero level set of a signed distance function d. Then, there exists a constant C such that:*

$$||d||_{L^\infty(\Gamma_h)} \leq Ch^2\,, \tag{2.35}$$

where h is the discretization size. Moreover, we have

$$||1 - \mu_h||_{L^\infty(\Gamma_h)} \leq Ch^2\,, \tag{2.36}$$

$$||(\boldsymbol{I} - \boldsymbol{A}_h)\boldsymbol{P}||_{L^\infty(\Gamma)} \leq Ch^2\,, \tag{2.37}$$

where

$$\boldsymbol{A}_h := \frac{1}{\mu_h}\boldsymbol{P}(\boldsymbol{I} - d\boldsymbol{H})\boldsymbol{P}_h(\boldsymbol{I} - d\boldsymbol{H})\boldsymbol{P}\,. \tag{2.38}$$

Proof. The proof of such relations can be found in [31].

□

Lemma 2.4.4 *Consider a function $f : \Gamma \to \mathbb{R}$ and its piecewise polynomial approximation $f_h : \Gamma_h \to \mathbb{R}$. Then, the following estimates hold*

$$||f_h||_{L^2(\Gamma_h)} \leq C||f||_{L^2(\Gamma)}\,, \tag{2.39}$$

$$||f - F_h||_{L^2(\Gamma)} \leq Ch^2||f||_{L^2(\Gamma)}\,, \tag{2.40}$$

where C is a constant, h is the discretization size and

$$F_h := \frac{1}{\mu_h}f_h\,.$$

Proof. The proof of such relations can be found in [31].

□

Definition 2.4.1 *Consider a function $u : \Gamma \to \mathbb{R}$ and its discrete approximation $u_h : \Gamma_h \to \mathbb{R}$, we define the discretization error as*

$$e_h := |u - u_h^E| \,. \tag{2.41}$$

where u_h^E is the extension of the discrete approximation u_h to U_δ, see Equation (2.9).

Now we have all the results needed to prove the energy estimate for the solution of Problem 2.3.2. We present the estimate introduced by G. Dziuk in [31], that treats the case $\partial\Gamma = \emptyset$. This proof can be easily extended to an open surface.

Lemma 2.4.5 *Let $|\partial\Gamma| = \emptyset$ and let u be the solution of Problem 2.3.1, u_h the solution of Problem 2.3.2 and u_h^E its extension to Γ given by*

$$u_h(\mathbf{x}) =: u_h^E(a(\mathbf{x})) \quad \mathbf{x} \in \Gamma_h \,.$$

Then, the following energy estimate holds

$$||\nabla_\Gamma(u - u_h^E)||_{L^2(\Gamma)} \leq \left(\inf_{\phi \in V} |u - \phi_h^E|_{H^1(\Gamma)} + h^2 ||f||_{L^2(\Gamma)} \right). \tag{2.42}$$

Proof. We consider a function $u_h \in V_h$, $f \in L^2(\Gamma)$ and $f_h \in L^2(\Gamma_h)$ such that

$$\int_\Gamma f \, d\sigma = 0 \quad \text{and} \quad \int_{\Gamma_h} f_h \, d\sigma_h = 0 \,.$$

see Remark 2.3.9 and 2.3.11. From Equation (2.17) and (2.23), we have

$$\int_\Gamma \nabla_\Gamma u \nabla_\Gamma \phi \, d\sigma = \int_\Gamma f\phi \, d\sigma \quad \forall \phi \in H^1(\Gamma) \,, \tag{2.43}$$

$$\int_{\Gamma_h} \nabla_{\Gamma_h} u_h \nabla_{\Gamma_h} \phi_h \, d\sigma_h = \int_{\Gamma_h} f_h \phi_h \, d\sigma_h \quad \forall \phi_h \in H^1(\Gamma_h) \,. \tag{2.44}$$

Then, according to Equation (2.29), we define

$$u_h(\mathbf{x}) =: u_h^E(a(\mathbf{x})) \quad \mathbf{x} \in \Gamma_h \,,$$

$$\phi_h(\mathbf{x}) \quad =: \quad \phi_h^E(a(\mathbf{x})) \quad \mathbf{x} \in \Gamma_h \,.$$

These functions belong to the following functional space:

$$V_h^E = \left\{ \phi_h(\mathbf{x}) = \phi_h^E(a(\mathbf{x})) \,, \mathbf{x} \in \Gamma_h \,, \phi_h(\mathbf{x}) \in V_h \right\} .$$

Via these transformations, Proposition 2.4.2 and the definition of $\mathbf{A}_h$, see Equation (2.38), we get, from Equation (2.44),

$$\int_\Gamma \mathbf{A}_h \nabla_\Gamma u_h^E \cdot \nabla_\Gamma \phi_h^E \, d\sigma = \int_\Gamma F_h \phi_h^E \, d\sigma \,.$$

Then, we have

$$\int_\Gamma \nabla_\Gamma u_h^E \cdot \nabla_\Gamma \phi_h^E \, d\sigma = \int_\Gamma F_h \phi_h^E \, d\sigma + \int_\Gamma (\mathbf{A}_h - \mathbf{I}) \nabla_\Gamma u_h^E \cdot \nabla_\Gamma \phi_h^E \, d\sigma$$

and, if we subtract this equation form Equation (2.43), where we choose $\phi = \phi_h^E$, we obtain

$$\int_\Gamma \nabla_\Gamma (u - u_h^E) \cdot \nabla_\Gamma \phi_h^E \, d\sigma = \int_\Gamma (\mathbf{I} - \mathbf{A}_h) \nabla_\Gamma u_h^E \cdot \nabla_\Gamma \phi_h^E \, d\sigma + \int_\Gamma (f - F_h) \phi_h^E \, d\sigma \,. \tag{2.45}$$

This relation is valid $\forall \phi_h^E \in H^1(\Gamma)$. Now, we look for an estimate of an error with respect to the $H^1(\Gamma)$-semi norm,

$$|u - u_h^E|^2_{H^1(\Gamma)} := ||\nabla_\Gamma (u - u_h^E)||^2_{L^2(\Gamma)} = \int_\Gamma \nabla_\Gamma (u - u_h^E) \nabla_\Gamma (u - u_h^E) \, d\sigma \,.$$

We sum and subtract the quantity

$$\int_\Gamma \nabla_\Gamma (u - u_h^E) \cdot \nabla_\Gamma \phi_h^E \, d\sigma \,,$$

and we use Equation (2.45), to get

$$\begin{aligned} |u - u_h^E|^2_{H^1(\Gamma)} \quad &= \quad \int_\Gamma \nabla_\Gamma (u - u_h^E) \cdot \nabla_\Gamma (u - \phi_h^E) \, d\sigma \\ &+ \quad \int_\Gamma (\mathbf{I} - \mathbf{A}_h) \nabla_\Gamma u_h^E \cdot \nabla_\Gamma (u_h^E - \phi_h^E) \, d\sigma \\ &- \quad \int_\Gamma (f - F_h)(u_h^E - \phi_h^E) \, d\sigma \,. \end{aligned}$$

Since $\mathbf{PP} = \mathbf{P}$, see Proposition 2.4.1, we have

$$\int_\Gamma (\mathbf{I} - \mathbf{A}_h)\nabla_\Gamma u_h^E \cdot \nabla_\Gamma (u_h^E - \phi_h^E)\, d\sigma = \int_\Gamma (\mathbf{I} - \mathbf{A}_h)\mathbf{P}\nabla_\Gamma u_h^E \cdot \nabla_\Gamma (u_h^E - \phi_h^E)\, d\sigma\,.$$

Then, via the Schwarz inequality, we have

$$\begin{aligned} |u - u_h^E|^2_{H^1(\Gamma)} \quad &\leq \quad ||\nabla_\Gamma (u - u_h^E)||_{L^2(\Gamma)} ||\nabla_\Gamma (u - \phi_h^E)||_{L^2(\Gamma)} + \\ &+ \quad ||(\mathbf{I} - \mathbf{A}_h)\mathbf{P}||_{L^\infty(\Gamma)} ||\nabla_\Gamma u_h^E||_{L^2(\Gamma)} ||\nabla_\Gamma (u_h^E - \phi_h^E)||_{L^2(\Gamma)} + \\ &+ \quad ||f - F_h||_{L^2(\Gamma)} ||u_h^E - \phi_h^E||_{L^2(\Gamma)}\,. \end{aligned}$$

Using the estimates of Lemmas 2.4.3, 2.4.4, we get

$$||(\mathbf{I} - \mathbf{A}_h)\mathbf{P}||_{L^\infty(\Gamma)} ||\nabla_\Gamma u_h^E||_{L^2(\Gamma)} \leq Ch^2 ||f||_{L^2(\Gamma)}\,.$$

Then, moving from the relations described in Lemma 2.4.4, we observe that

$$||f - F_h||_{L^2(\Gamma)} ||u_h^E - \phi_h^E||_{L^2(\Gamma)} \leq Ch^2 ||f||_{L^2(\Gamma)} ||u_h^E - \phi_h^E||_{L^2(\Gamma)}\,.$$

Without loss of generality, we can suppose that

$$\int_\Gamma (u_h^E - \phi_h^E) = 0\,,$$

and using the Poincaré inequality, we get the desired estimate, Equation (2.42), see [31] for more details.

□

Now we analyse more in details the error estimator proposed in Lemma 2.4.5:

$$||\nabla_\Gamma (u - u_h^E)||_{L^2(\Gamma)} \leq \Bigg(\underbrace{\inf_{\phi_h \in V_h} |u - \phi_h^E|_{H^1(\Gamma)}}_{(I)} + \underbrace{h^2 ||f||_{L^2(\Gamma)}}_{(II)} \Bigg)\,. \tag{2.46}$$

We notice that the error is composed by two different parts.

(I) represents the classical error due to the finite element approximation, i.e., the "best approximation error";

(II) represents the error due to the discretization of Γ with Γ_h.

In this framework the last term plays an important role. In fact, if we increase the order of the finite elements, i.e., if we increase the degree of the polynomial space, we do not get a convergence rate greater than two in the semi norm H^1. To achieve a higher convergence rate, it is necessary to approximate the surface Γ with curved elements, e.g., piecewise parabolic elements. In [24] it is proved a convergence result for curved elements. In particular, if the discrete surface is approximated by piecewise elements of order k, the error estimate defined in Equation (2.42) becomes:

$$||\nabla_\Gamma(u - u_h^E)||_{L^2(\Gamma)} \leq \left(\inf_{\phi_h \in V_h} |u - \phi_h^E|_{H^1(\Gamma)} + h^{k+1}||f||_{L^2(\Gamma)} \right). \tag{2.47}$$

2.5 A Convection-Diffusion Problem Defined on a Surface

There are a lot of mathematical models that involve the resolution of partial differential equations defined on surfaces. Sometimes these equations are coupled with other equations that are formulated in a fixed domain which contains the surface. This could happen in a multiphase fluids dynamics, see, e.g., [53]. The surface transport of such surface is driven by a Convection-Diffusion problem defined on a surface:

$$\begin{cases} -\Delta_\Gamma u + \mathbf{v} \cdot \nabla_\Gamma u &= f \quad \text{on } \Gamma\,, \\ u &= 0 \quad \text{on } \partial\Gamma\,, \end{cases} \tag{2.48}$$

where Γ is an arbitrary two-dimensional surface embedded in $\mathbb{R}^3$ and $\mathbf{v}$ is a vector field defined on Γ.

In this section we analyse in more detail this kind of problems. In particular, we focus on the corresponding discretization.

To get the weak form of (2.48), we introduce the same functional spaces used for the Laplace-Beltrami problem, see Subsection 2.3.1. We multiply the first equation of system defined in Equation (2.48) by $\phi \in H_0^1(\Gamma)$ and we integrate by parts. Finally, we get the following problem.

Problem 2.5.1 *Given a function $f \in L^2(\Gamma)$, find $u \in H_0^1(\Gamma)$ such that:*

$$\mathcal{A}^{CV}(u\,, \phi) = \mathcal{B}(\phi)\,, \quad \forall \phi \in H_0^1(\Gamma)\,. \tag{2.49}$$

Here $\mathcal{B}$ is the same linear form as in Problem 2.3.1, while we define a new bilinear form $\mathcal{A}^{CV} : H_0^1(\Gamma) \times H_0^1(\Gamma) \to \mathbb{R}$ as

$$\mathcal{A}^{CV}(u\,,\phi) := \int_\Gamma (\nabla_\Gamma u \cdot \nabla_\Gamma \phi + \boldsymbol{v} \cdot \nabla_\Gamma u\, \phi)\, d\sigma\,. \tag{2.50}$$

To discretize Equation (2.48), we follow the theory provided in [31]. We consider a discretization Γ_h of Γ, as defined in Section 2.3, and f_h, $\mathbf{v}_h$ are suitable extension of the function f and the vector field $\mathbf{v}$, i.e., $f_h : \Gamma_h \to \mathbb{R}$ such that $f_h|_\Gamma = f$ and $\mathbf{v}_h : \Gamma_h \to \mathbb{R}^3$ such that $\mathbf{v}_h|_\Gamma = \mathbf{v}$. The discrete version of the system defined in Equation (2.48) becomes

$$\begin{cases} -\Delta_{\Gamma_h} u_h + v_h \nabla_{\Gamma_h} u_h &= f_h \quad \text{on } \Gamma_h \\ u_h &= 0 \quad \text{on } \partial\Gamma_h \end{cases}. \tag{2.51}$$

As for the continuous formulation, we introduce proper functional spaces and then we are able to state the weak form of (2.51).

Problem 2.5.2 *Given a function $f_h \in L^2(\Gamma_h)$, find $u_h \in H_0^1(\Gamma_h)$, such that*

$$\mathcal{A}_h^{CV}(u_h\,,\phi_h) = \mathcal{B}_h(\phi_h)\,, \qquad \forall \phi_h \in H_0^1(\Gamma_h)\,, \tag{2.52}$$

Here $\mathcal{B}_h$ is the same linear form as in Problem 2.3.2 and we define the bilinear form $\mathcal{A}_h^{CV} : H_0^1(\Gamma_h) \times H_0^1(\Gamma_h) \to \mathbb{R}$ as

$$\mathcal{A}_h^{CV}(u_h\,,\phi_h) := \int_\Gamma \nabla_{\Gamma_h} u_h \cdot \nabla_{\Gamma_h} \phi_h + \boldsymbol{v}_h \cdot \nabla_{\Gamma_h} u\, \phi_h\, d\sigma_h\,. \tag{2.53}$$

Remark 2.5.1 *For simplicity, we assume that the vector field $\boldsymbol{v}$, defined on the surface Γ, is extended to a vector field $\mathbf{v}^E$ such that $\mathbf{v}^E|_\Gamma = \boldsymbol{v}$ and $\mathbf{v}^E|_{\Gamma_h} = \boldsymbol{v}_h$. Moreover, we are not taking into account the error due to the discretization of this vector field.*

2.6 Gradient Recovery Techniques

Consider a piecewise linear approximation, u_h, of a function u. In a finite element analysis, one could be interested in the gradient of u_h. For instance, when we are

dealing with an elasticity problem, we might be more interested in computing the stresses and the strains rather than the displacements of the elastic body at hand. Furthermore, the normal component of the gradient of a discrete function u_h is generally discontinuous across the edges of a mesh element and, consequently, we obtain a discontinuous approximation of the quantity of interest. For this reason, it could be useful a **gradient recovery**, i.e., a post-processing procedure that smooths the gradient of a discrete function u_h.

In [113, 114, 115], it is shown that, under certain mesh configurations, the recovered gradient is more accurate than the exact gradient of the original finite element function, u_h.

In this section we show how to generalize the standard local gradient recovery schemes in the case of non-planar meshes. Firstly, we recall the local averaging schemes in the planar case, then we move to the non-planar framework.

2.6.1 The Planar Case

Consider a planar triangular mesh Ω_h and a node $\mathbf{z} \in \omega_\mathbf{z}$, where $\omega_\mathbf{z}$ is the set of triangles that share the node $\mathbf{z}$, i.e., such that $\mathbf{z} \in T_j$ for at least one j, see Figure 2.5. We consider a standard finite element space of piece-wise linear functions on Ω_h, W_h, and a finite element function $w_h \in W_h$.

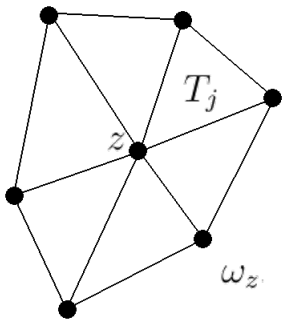


Figure 2.5: Patch $\omega_\mathbf{z}$ in the case of planar domain, Ω_h.

Given the finite element approximation w_h of a generic function $w \in W$, where W is a proper functional space defined in Ω, we may use one of the follow-

ing **local gradient recovery** schemes, [112], to get a more accurate approximation of the actual gradient:

1. simple averaging:

$$(\mathcal{G}_h \nabla w_h)(\mathbf{z}) := \frac{1}{m} \sum_{j=1}^{m} \nabla w_h|_{T_j}(\mathbf{z}) \,, \tag{2.54}$$

   here $m = \#\omega_z$ denotes the cardinality of the patch $\omega_\mathbf{z}$, i.e., the number of the triangles that share the node $\mathbf{z}$;

2. weighted averaging:

$$(\mathcal{G}_h \nabla w_h)(\mathbf{z}) := \sum_{j=1}^{m} \frac{|T_j|}{|\omega_\mathbf{z}|} \nabla w_h|_{T_j}(\mathbf{z}) \,, \tag{2.55}$$

   where $|T_j|$ and $|\omega_\mathbf{z}|$ are the areas of the triangle T_j and of the patch $\omega_\mathbf{z}$, respectively.

Then, this super-convergence result holds.

Theorem 2.6.1 *Consider a function $u \in W^3_\infty(\omega_z)$. Let u_I be the nodal interpolant of u and $\mathcal{G}_h \nabla u_h(\mathbf{z})$ be the recovered gradient produced by either the simple averaging, or by the weighted averaging. Then, there exists a constant C such that the following inequality holds:*

$$|(\mathcal{G}_h \nabla u_I)(\mathbf{z}) - \nabla u(\mathbf{z})| < Ch^2 ||u||_{3,\infty,\omega_\mathbf{z}} \,. \tag{2.56}$$

Here we have defined the functional space

$$W^3_\infty(\omega_z) := \{ f \in L^\infty(\omega_z) \,:\, D^\alpha f \in L^\infty(\omega_z) \, \forall |\alpha| \leq 3 \} \,.$$

Proof. The proof of this result is provided in [112]. In particular, the authors exploit the fact that the barycenter of a triangle is a point of super-convergence for the derivative of the linear interpolant.

2.6.2 The Non-Planar Case

In [110, 28], the superconvergence result of Theorem 2.6.1 is extended to non-planar surface meshes. Consider a two-dimensional surface triangular mesh embedded in the three-dimensional space Γ_h and a node $\mathbf{z} \in \omega_{\mathbf{z}}$.

Remark 2.6.1 *In this case ω_z, the patch of triangles T_j that share the node $\mathbf{z}$, i.e., $\mathbf{z} \in T_j$ for at least one j, does **not** necessarily lies on a plane, see Figure 2.6.*

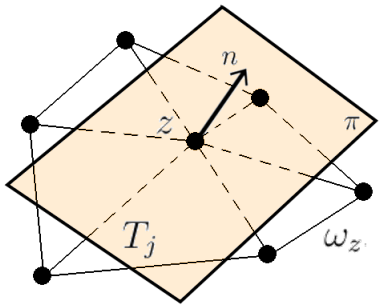


Figure 2.6: Patch $\omega_{\mathbf{z}}$ in the case of a non-planar domain, Γ_h, we highlighted the tangent plane π with normal $\mathbf{n}$.

In [110, 28] the authors propose different ways of extend both the simple and weighted averaging recovery schemes to the non-planar case. As before we consider a finite element space of piecewise linear functions on Γ_h, V_h, a finite element approximation $v_h \in V_h$ of a function $v \in V$, where V is a proper functional space associated with Γ. A more accurate evaluation of the actual tangential gradient $\nabla_\Gamma v$, can be obtained by one of the following gradient recovery schemes:

1. simple averaging:

$$(\mathcal{G}_h \nabla_{\Gamma_h} v_h)(\mathbf{z}) := \frac{1}{m} \sum_{j=1}^{m} \nabla_{\Gamma_h} v_h|_{T_j}(\mathbf{z}) \,, \tag{2.57}$$

   here $m = \#\omega_z$ is the cardinality of the patch $\omega_{\mathbf{z}}$;

2. weighted averaging:

$$(\mathcal{G}_h \nabla_{\Gamma_h} v_h)(\mathbf{z}) := \sum_{j=1}^{m} \frac{|T_j|}{|\omega_{\mathbf{z}}|} \nabla_{\Gamma_h} v_h|_{T_j}(\mathbf{z}) \,, \tag{2.58}$$

   where $|T_j|$ and $|\omega_{\mathbf{z}}|$ are the areas of the triangle T_j and of the patch $\omega_{\mathbf{z}}$, respectively.

3. tangential weighted averaging:

$$(\mathcal{GT}_h \nabla_{\Gamma_h} v_h)(\mathbf{z}) := \sum_{j=1}^{m} \frac{|T_j^E|}{|\omega_{\mathbf{z}}^E|} \nabla_{\Gamma_h} v_h|_{T_j}(\mathbf{z}) \,, \tag{2.59}$$

   where $|T_j^E|$ and $|\omega_{\mathbf{z}}^E|$ are the areas of the triangles T_j and of the patch $\omega_{\mathbf{z}}$, respectively, projected on the tangent plane to Γ at the node $\mathbf{z}$.

In [110, 28] it is proved a convergence result similar to the one provided in Theorem 2.6.1 for the non-planar case.

Theorem 2.6.2 *Let $u \in W_\infty^3(\Gamma)$, u^E and u_I^E be the linear interpolant of u^E on Γ_h. Let $\mathcal{R}\nabla_{\Gamma_h} u_I^E(\boldsymbol{z})$ be the recovered gradient produced by one of the averaging scheme defined in Equation (2.57), (2.58) or (2.59). If the patch $\omega_{\boldsymbol{z}}$ is $\mathcal{O}(h^2)$ symmetric, then there exists a constant C such that the following inequality holds*

$$|\mathcal{R}\nabla_{\Gamma_h} u_I^E(\boldsymbol{z}) - \nabla_\Gamma u(\boldsymbol{z})| < Ch^2 ||u||_{3,\infty,\Gamma} \,. \tag{2.60}$$

Chapter 3

Error Estimators for PDEs Defined on Surfaces

In this chapter we introduce some new anisotropic error estimators for PDEs defined on surfaces. Due to the novelty of the proposed approach, we limit our analysis to simple scalar PDEs to assess the robustness of the proposed estimators. In more details, we extend the theory provided in [40] to Partial Differential Equations defined on surfaces. Moving from a new anisotropic interpolator error estimator, we introduce two anisotropic a-posteriori error estimators for a standard Laplace-Beltrami problem and for a standard convection-diffusion problem defined on a surface. In particular, we deal with both a residual-based and a goal-oriented analysis. Finally, in the last section of this chapter, we extend the standard Zienkiewicz-Zhu error estimator to an anisotropic setting, starting from the theory provided in [114, 115].

Moving from these error estimators, we propose a metric-based anisotropic mesh adaptation procedure that employs local operations, see Section 1.2, to modify the mesh according to the information provided by these estimators.

The reliability of the proposed mesh optimization procedure is numerically investigated via several numerical examples.

3.1 Source of the Anisotropic Information

Since we are dealing with anisotropic mesh adaptation techniques, we are interested in identifying the size, the shape and the orientation of each element of the mesh. To achieve this goal, we follow the theory provided in [40].

We start from the affine transformation $F_T : \hat{T} \to T$, where $\hat{T}$ is the reference triangle in $\mathbb{R}^2$, whose vertices are the points

$$\hat{A} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \hat{B} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \hat{C} = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

where T is a generic non degenerate triangle embedded in $\mathbb{R}^3$, see Figure 3.1. This transformation is defined as

$$\mathbf{x} = \mathbf{M}_T \hat{\mathbf{x}} + \mathbf{b}_T, \quad \forall \hat{\mathbf{x}} \in \hat{T}, \tag{3.1}$$

where $\mathbf{x} \in T$, $\mathbf{M}_T \in \mathbb{R}^{3\times 2}$ and $\mathbf{b}_T \in \mathbb{R}^3$.

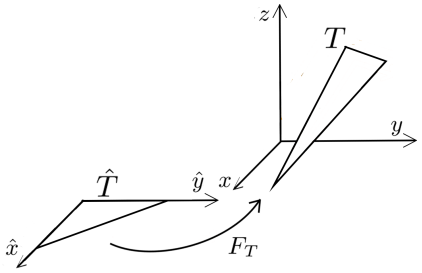


Figure 3.1: Scheme of the transformation $F_T : \hat{T} \subset \mathbb{R}^2 \to T \subset \mathbb{R}^3$.

In particular, we recover size, shape and orientation for each element of the grid by exploiting the singular value decomposition of $\mathbf{M}_T$. Unlike the theory provided in [40], in this framework $\mathbf{M}_T$ is a rectangular matrix, so we have to consider the extension of the singular value decomposition to rectangular matrices, see Theorem 3.1.1 and [52].

Theorem 3.1.1 *Consider a matrix $\boldsymbol{M}_T \in \mathbb{R}^{m\times n}$, then there exists a factorization of the form*

$$\boldsymbol{M}_T = \mathbf{U}_T \mathbf{S}_T \mathbf{V}_T^t, \tag{3.2}$$

where $\mathbf{U}_T \in \mathbb{R}^{m\times m}$ is an unitary matrix, $\mathbf{S}_T \in \mathbb{R}^{m\times n}$ is a diagonal matrix with non-negative real numbers on the diagonal, and $\mathbf{V}_T \in \mathbb{R}^{n\times n}$ is an unitary matrix and $\mathbf{V}^t$ denotes the conjugate transpose of $\mathbf{V}_T$. Such a factorization is called a singular value decomposition of $\boldsymbol{M}_T$. The diagonal entries of $\mathbf{S}_T$ are known as the singular values of $\boldsymbol{M}_T$. The m columns of $\mathbf{U}_T$ and the n columns of $\mathbf{V}_T$ are called the left-singular vectors and right-singular vectors of $\boldsymbol{M}_T$, respectively.

3.2 An Anisotropic Interpolation Error Estimate

In the first part of this section we recall the results provided in [25, 24] to get an isotropic a-priori error estimator for the Laplace-Beltrami equation. We introduce a suitable interpolation operator.

Definition 3.2.1 *Consider a surface Γ and a polyhedral approximation of this surface, Γ_h. Given a function $\psi : \Gamma \to \mathbb{R}$ with $\psi \in L^1(\Gamma)$, let*

$$\psi_{\mathbf{z}}^E := \frac{1}{\int_{\omega_{\mathbf{z}}} \phi_{\mathbf{z}} \, d\sigma_h} \int_{\omega_{\mathbf{z}}} \phi_{\mathbf{z}} \psi^E \, d\sigma_h \,, \tag{3.3}$$

where $\{\phi_{\mathbf{z}}\}_{\mathbf{z}\in\mathcal{N}}$ is a partition of the unity representing the finite element basis defined on the surface mesh Γ_h, $\omega_{\mathbf{z}}$ is the set of triangles in Γ_h that share the vertex $\mathbf{z}$, ψ^E is an extension of the function ψ to U_δ such that $\psi^E|_\Gamma = \psi$, see Equation (2.9), and $\mathcal{N}$ is the set of the vertices of Γ_h. Then, we introduce the interpolant operator

$$I_h \psi^E := \sum_{\mathbf{z}\in\mathcal{N}} \psi_{\mathbf{z}}^E \phi_{\mathbf{z}} \,. \tag{3.4}$$

Lemma 3.2.1 *Consider a surface Γ and its polyhedral approximation, Γ_h. Let $\psi \in H^1(\Gamma)$ and let $m_{\mathbf{z}}$ be the number of elements sharing the node $\mathbf{z}$. Let $\omega_{\mathbf{z}}^E$ be the lift of the patch $\omega_{\mathbf{z}}$ onto Γ. Then, for each vertex $\mathbf{z}$ of the mesh Γ_h, the following relation holds:*

$$||\psi^E - \psi_{\mathbf{z}}^E||_{L^2(\omega_{\mathbf{z}})} \leq C(\omega_{\mathbf{z}}) \max_{T\subset\omega_{\mathbf{z}}} \sqrt{|T|}\, m_{\mathbf{z}} \max_{T\subset\omega_{\mathbf{z}}} \frac{h_T}{\sqrt{|T|}} ||\nabla_\Gamma \psi||_{L^2(\omega_{\mathbf{z}}^E)} \,, \tag{3.5}$$

where $|T|$ and h_T are the area and the diameter of a triangle $T \in \omega_{\mathbf{z}}$, respectively, while $C(\omega_{\mathbf{z}})$ is a constant that depends on the patch $\omega_{\mathbf{z}}$.

Proof. The proof of relation (3.5) can be found in [25].

□

Relation (3.5) takes into account *only* the size of the mesh elements via the quantities $\sqrt{|T|}$ and h_T. Since we are interested in an estimate that includes also the shape and the orientation of the mesh elements, we have generalized estimate (3.5), i.e., we have properly included the anisotropic information of Section 3.1. In particular, our goal is to obtain an anisotropic bound for the interpolation error $\psi - I_h\psi^E$ associated with the operator (3.4), for any $\psi \in L^1(\Gamma)$. For this purpose, we prove the anisotropic counter part of Lemma 3.2.1.

Proposition 3.2.1 *Let $\psi \in H^1(\Gamma)$. Then, for each node $\mathbf{z} \in \mathcal{N}$, there exists a constant C such that*

$$||\psi^E - \psi^E_{\mathbf{z}}||_{L^2(\omega_{\mathbf{z}})} \leq C \sum_{T \in \omega_{\mathbf{z}}} \left(s^2_{1,T} (\mathbf{r}_{1,T})^t \, \mathbf{G}_T(\psi^E) \, \mathbf{r}_{1,T} + s^2_{2,T} (\mathbf{r}_{2,T})^t \, \mathbf{G}_T(\psi^E) \, \mathbf{r}_{2,T} \right)^{1/2}, \tag{3.6}$$

with $\psi^E : U_\delta \to \mathbb{R}$ the extension of ψ to U_δ according to definition (2.9), $\psi^E_{\mathbf{z}}$ defined as in (3.3), $\mathbf{G}_T$ the symmetric positive semi-definite matrix given by

$$\mathbf{G}_T(\psi^E) = \begin{pmatrix} \int_T (g_1)^2 \, d\sigma_h & \int_T g_1 g_2 \, d\sigma_h & \int_T g_1 g_3 \, d\sigma_h \\ \int_T g_2 g_1 \, d\sigma_h & \int_T (g_2)^2 \, d\sigma_h & \int_T g_2 g_3 \, d\sigma_h \\ \int_T g_3 g_1 \, d\sigma_h & \int_T g_3 g_2 \, d\sigma_h & \int_T (g_3)^2 \, d\sigma_h \end{pmatrix}, \tag{3.7}$$

where $g_i = (\nabla_{\Gamma_h} \psi^E)_i$ for $i = 1, 2, 3$ denotes the $i-$th component of the tangential gradient $\nabla_{\Gamma_h} \psi^E = \nabla \psi^E - (\mathbf{n}_h \cdot \nabla \psi^E) \mathbf{n}_h$ with respect to the standard cartesian coordinate system in $\mathbb{R}^3$ and ψ^E is the extension of ψ, such that $\psi^E|_\Gamma = \psi$.

Proof. The first part of this proof follows Lemma 3.2.1 in [25]. Moving from definition (3.3) and the Cauchy-Schwarz inequality, we get

$$||\psi^E_{\mathbf{z}}||_{L^2(\omega_{\mathbf{z}})} = |\omega_{\mathbf{z}}|^{1/2} |\psi^E_{\mathbf{z}}| \leq |\omega_{\mathbf{z}}|^{1/2} \frac{||\varphi_{\mathbf{z}}||_{L^2(\omega_{\mathbf{z}})}}{||\varphi_{\mathbf{z}}||_{L^1(\omega_{\mathbf{z}})}} ||\psi^E||_{L^2(\omega_{\mathbf{z}})}, \tag{3.8}$$

with $|\cdot|$ we denote the measure of a generic set in $\mathbb{R}^d$, with $d = 1, 2, 3$.

Now, by exploiting the map F_T defined in Equation (3.1), for each $T \in \omega_{\mathbf{z}}$, we consider the piecewise affine map $F_{\mathbf{z}} : \widehat{\omega}_{\mathbf{z}} \to \omega_{\mathbf{z}}$, where $\widehat{\omega}_{\mathbf{z}}$ is the union of the inverse image of all the triangles T that form the patch $\omega_{\mathbf{z}}$. Analogously, we denote by $\widehat{u}$ the inverse image of a generic function $u \in H^1(\Gamma_h)$. The L^2- and the L^1-norm in (3.8) can be easily computed coming back to the reference framework as

$$\|\varphi_{\mathbf{z}}\|^p_{L^p(\omega_{\mathbf{z}})} = \sum_{T \in \omega_{\mathbf{z}}} \int_T (\varphi_{\mathbf{z}})^p \, d\sigma_h = \sum_{T \in \omega_{\mathbf{z}}} \frac{|T|}{|\widehat{T}|} \int_{\widehat{T}} (\widehat{\varphi}_{\mathbf{z}})^p \, d\widehat{\sigma} = \frac{|\omega_{\mathbf{z}}|}{|\widehat{T}|} \int_{\widehat{T}} (\widehat{\varphi}_{\mathbf{z}})^p \, d\widehat{\sigma}, \tag{3.9}$$

where

$$\|\widehat{\varphi}_{\mathbf{z}}\|_{L^1(\widehat{T})} = 1/6, \quad \|\widehat{\varphi}_{\mathbf{z}}\|_{L^2(\widehat{T})} = 1/\sqrt{12} \quad \text{and} \quad |\widehat{T}| = 1/2.$$

By properly substituting (3.9) in (3.8), we get

$$||\psi^E_{\mathbf{z}}||_{L^2(\omega_{\mathbf{z}})} \leq \sqrt{\frac{3}{2}}\,||\psi^E||_{L^2(\omega_{\mathbf{z}})}\,. \tag{3.10}$$

In a similar way, picked a constant $K \in \mathbb{R}$ and moving from the definition (3.3) and thanks to the Cauchy-Schwarz inequality, we have

$$\begin{aligned}
\|\psi^E_{\mathbf{z}} - K\|_{L^2(\omega_{\mathbf{z}})} &= |\omega_{\mathbf{z}}|^{1/2}|\psi^E_{\mathbf{z}} - K| \\
&= \frac{|\omega_{\mathbf{z}}|^{1/2}}{\left|\int_{\omega_{\mathbf{z}}} \varphi_{\mathbf{z}}\, d\sigma_h\right|}\left|\int_{\omega_{\mathbf{z}}} \left(\psi^E - K\right)\varphi_{\mathbf{z}}\, d\sigma_h\right| \\
&\leq \frac{|\omega_{\mathbf{z}}|^{1/2}}{\|\varphi_{\mathbf{z}}\|_{L^1(\omega_{\mathbf{z}})}}\|\psi^E - K\|_{L^2(\omega_{\mathbf{z}})}\|\varphi_{\mathbf{z}}\|_{L^2(\omega_{\mathbf{z}})} \\
&\leq \sqrt{\frac{3}{2}}\,\|\psi^E - K\|_{L^2(\omega_{\mathbf{z}})}.
\end{aligned} \tag{3.11}$$

The triangle inequality yields

$$||\psi^E - \psi^E_{\mathbf{z}}||_{L^2(\omega_{\mathbf{z}})} \leq \left(1 + \sqrt{\frac{3}{2}}\right)||\psi^E - K||_{L^2(\omega_{\mathbf{z}})}\,. \tag{3.12}$$

Now, we properly choose the constant K in Equation (3.11) in order to exploit the spectral decomposition of $\mathbf{M}_T$. Via this decomposition we obtain an anisotropic bound for the right-hand side in (3.12). Indeed, we set

$$K = \frac{1}{|\widehat{T}|}\int_{\widehat{T}} \eta(\widehat{\mathbf{x}})\, d\widehat{\sigma}\,, \tag{3.13}$$

where η is a function defined on $\widehat{T}$ such that $\eta(\widehat{\mathbf{x}}) = \psi^E(F_T(\widehat{\mathbf{x}}))$, for any $\widehat{\mathbf{x}} \in \widehat{T}$. Thus, thanks to the standard Poincaré inequality, we have

$$\begin{aligned}
\|\psi^E - \psi^E_{\mathbf{z}}\|^2_{L^2(\omega_{\mathbf{z}})} &\leq C\sum_{T\in\omega_{\mathbf{z}}}\frac{|T|}{|\widehat{T}|}\int_{\widehat{T}}(\eta - K)^2\, d\widehat{\sigma} \\
&\leq C\sum_{T\in\omega_{\mathbf{z}}}\frac{|T|}{|\widehat{T}|}\int_{\widehat{T}}\left(\widehat{\nabla}\eta\right)^2 d\widehat{\sigma}\,,
\end{aligned}$$

with $\widehat{\nabla}$ the gradient operator with respect to the coordinate system of the reference plane $\hat{x}O\hat{y}$. We notice that

$$\widehat{\nabla}\eta = \mathbf{M}_T^t \nabla_{\Gamma_h}\psi^E\,,$$

with $\nabla_{\Gamma_h}\psi^E$ is the tangential gradient associated with Γ_h. Coming back to Γ_h, we get this inequality,

$$\|\psi^E - \psi^E_{\mathbf{z}}\|^2_{L^2(\omega_{\mathbf{z}})} \leq C \sum_{T\in\omega_{\mathbf{z}}} \int_T \left|\mathbf{M}_T^t \nabla_{\Gamma_h}\psi^E\right|^2 d\sigma_h\,.$$

To introduce anisotropic information, we simply resort to the SVD of $\mathbf{M}_T$, to have

$$\|\psi^E - \psi^E_{\mathbf{z}}\|^2_{L^2(\omega_{\mathbf{z}})} \leq C \sum_{T\in\omega_{\mathbf{z}}} \int_T \left(\left(\nabla_{\Gamma_h}\psi^E\right)^t \mathbf{U}_T\mathbf{S}_T\mathbf{S}_T^t\mathbf{U}_T^t\left(\nabla_{\Gamma_h}\psi^E\right)\right) d\sigma_h\,, \tag{3.14}$$

where matrix $\mathbf{V}_T$ does not contribute since $\mathbf{V}_T^t\mathbf{V}_T = \mathbf{I}$. Now, it can be easily checked that the matrix product $\mathbf{U}_T\mathbf{S}_T\mathbf{S}_T^t\mathbf{U}_T^t$ in Equation (3.14) can be expressed in terms of the anisotropic lengths $s_{1,T}$, $s_{2,T}$ and directions $\mathbf{r}_{1,T}$ and $\mathbf{r}_{2,T}$, as

$$\left(\nabla_{\Gamma_h}\psi^E\right)^t \mathbf{U}_T\mathbf{S}_T\mathbf{S}_T^t\mathbf{U}_T^t\left(\nabla_{\Gamma_h}\psi^E\right) = \sum_{i=1}^{2} s_{i,T}^2 \left(\nabla_{\Gamma_h}\psi^E\right)^t \mathbf{r}_{i,T}\otimes\mathbf{r}_{i,T}\left(\nabla_{\Gamma_h}\psi^E\right).$$

This leads to rewrite Equation (3.14) as

$$\|\psi^E - \psi^E_{\mathbf{z}}\|^2_{L^2(\omega_{\mathbf{z}})} \leq C \sum_{T\in\omega_{\mathbf{z}}} \left(\sum_{i=1}^{2} s_{i,T}^2 \int_T \left(\nabla_{\Gamma_h}\psi^E\right)^t \mathbf{r}_{i,T}\otimes\mathbf{r}_{i,T}\left(\nabla_{\Gamma_h}\psi^E\right) d\sigma_h\right),$$

where $\otimes$ denotes the standard outer product between vectors. Straightforward algebraic manipulations show that, for $i = 1, 2$, it holds

$$\left(\nabla_{\Gamma_h}\psi^E\right)^t \mathbf{r}_{i,T}\otimes\mathbf{r}_{i,T}\left(\nabla_{\Gamma_h}\psi^E\right) = \mathbf{r}_{i,T}^t\left(\nabla_{\Gamma_h}\psi^E\otimes\nabla_{\Gamma_h}\psi^E\right)\mathbf{r}_{i,T} = \mathbf{r}_{i,T}^t\mathbf{G}_T(\psi^E)\mathbf{r}_{i,T},$$

where $\mathbf{G}_T$ is the matrix defined in (3.7). This completes the proof.

□

A comparison between Equation (3.6) and the corresponding estimate defined in Equation (3.5) for shape-regular meshes, highlights the richer and more complex structure of the anisotropic result. Let us introduce the so-called stretching factor

$$s_T := \frac{s_{1,T}}{s_{2,T}} \geq 1\,, \tag{3.15}$$

we observe that the quantities $s_{1,T}$ and $s_{2,T}$ are related to the area of the physical triangle T, $s_{1,T}\, s_{2,T} = 2\,|T|$. Thus, we can rewrite Equation (3.6) as:

$$||\psi^E - \psi^E_{\mathbf{z}}||^2_{L^2(\omega_{\mathbf{z}})} \leq \sum_{T \in \omega_{\mathbf{z}}} 2\,|T| \left(s_T (\mathbf{r}_{1,T})^t\, \mathbf{G}_T(\psi^E)\, \mathbf{r}_{1,T} + \frac{1}{s_T} (\mathbf{r}_{2,T})^t\, \mathbf{G}_T(\psi^E)\, \mathbf{r}_{2,T} \right) , \tag{3.16}$$

This new version of Equation (3.6) highlights all the desired information about the anisotropy of the triangle T, i.e.,

- *size information:* the area $|T|$ of the triangle;
- *shape information:* the stretching factor s_T; high values of s_T correspond to highly stretched elements, while $s_T = 1$ corresponds to the equilateral triangle;
- *directional information:* the unitary vectors $\mathbf{r}_{1,T}$ and $\mathbf{r}_{2,T}$ give the orientation of the triangle.

Remark 3.2.1 *The dependence of the term defined in Equation (3.16) on $\mathbf{r}_{2,T}$ is implicit, due to the orthogonality relation between the two singular vectors $\mathbf{r}_{1,T}$ and $\mathbf{r}_{2,T}$.*

The next proposition represents the theoretical at the basis of the anisotropic mesh adaptive procedure of Subsection 3.2.2.

Proposition 3.2.2 *Let $\psi \in H^1(\Gamma)$ and let I_h^E denote the extension of the interpolant in (3.4) according to definition (2.9). Then, there exists a constant C such that*

$$\|\psi - I_h^E \psi^E\|_{L^1(\Gamma)} \leq C \sum_{T \in \mathcal{T}_h} \overline{\alpha}_T |T|^{3/2} \nu_T(\sigma_T, \mathbf{r}_{1,T}, \psi^E)\,, \tag{3.17}$$

where

$$\overline{\alpha}_T = \frac{1}{\sqrt{|T|}} \sum_{\mathbf{z} \in T} \|\varphi_{\mathbf{z}} \mu_h\|_{L^2(\omega_{\mathbf{z}})}\,,$$

with μ_h defined as in Equation (2.34) and $\varphi_{\mathbf{z}}$ the basis function associated with node $\mathbf{z}$, while

$$\nu_T(s_T, \mathbf{r}_{1,T}, \psi^E) = \left(s_T \mathbf{r}_{1,T}^t \overline{\mathbf{G}}_T(\psi^E) \mathbf{r}_{1,T} + \frac{1}{s_T} \mathbf{r}_{2,T}^t \overline{\mathbf{G}}_T(\psi^E) \mathbf{r}_{2,T} \right)^{1/2}, \tag{3.18}$$

with $\overline{\mathbf{G}}_T = \mathbf{G}_T/|T|$.

Proof. We employ the Jacobian μ_h in (2.34), the definition (3.4) of the interpolant operator and we exploit the fact that the set of functions $\{\varphi_{\mathbf{z}}\}_{\mathbf{z}\in\mathbb{R}^{\mathcal{N}}}$ represents a partition of unity. So we get

$$\begin{aligned}
\|\psi - I_h^E\psi^E\|_{L^1(\Gamma)} &= \int_\Gamma \left|\psi(\mathbf{x}) - I_h^E\psi(\mathbf{x})\right| d\sigma \\
&= \int_{\Gamma_h} \left|\psi^E(\mathbf{x}) - I_h\psi^E(\mathbf{x})\right| \left|\mu_h(\mathbf{x})\right| d\Gamma_h \\
&= \int_{\Gamma_h} \left|\psi^E(\mathbf{x}) - \sum_{\mathbf{z}\in\mathcal{N}} \psi_{\mathbf{z}}^E\varphi_{\mathbf{z}}(\mathbf{x})\right| \left|\mu_h(\mathbf{x})\right| d\sigma_h \\
&= \int_{\Gamma_h} \left|\sum_{\mathbf{z}\in\mathcal{N}} \left(\psi^E(\mathbf{x}) - \psi_{\mathbf{z}}^E\right)\varphi_{\mathbf{z}}(\mathbf{x})\right| \left|\mu_h(\mathbf{x})\right| d\sigma_h \\
&\leq \sum_{\mathbf{z}\in\mathcal{N}} \int_{\omega_{\mathbf{z}}} \left|\psi^E(\mathbf{x}) - \psi_{\mathbf{z}}^E\right| \left|\varphi_{\mathbf{z}}(\mathbf{x})\mu_h(\mathbf{x})\right| d\sigma_{\mathbf{z}},
\end{aligned}$$

where, since $\omega_{\mathbf{z}}$ is the compact support of $\varphi_{\mathbf{z}}$, we have localized the integral on Γ_h. Then, moving from the Cauchy-Schwarz inequality and thanks to estimate (3.16), we obtain

$$\begin{aligned}
\|\psi - I_h^E\psi^E\|_{L^1(\Gamma)} &\leq \sum_{\mathbf{z}\in\mathcal{N}} \|\psi^E - \psi_{\mathbf{z}}^E\|_{L^2(\omega_{\mathbf{z}})} \|\varphi_{\mathbf{z}}\mu_h\|_{L^2(\omega_{\mathbf{z}})} \\
&\leq C \sum_{\mathbf{z}\in\mathcal{N}} \|\varphi_{\mathbf{z}}\mu_h\|_{L^2(\omega_{\mathbf{z}})} \left(\sum_{T\in\omega_{\mathbf{z}}} |T| \left(s_T \mathbf{r}_{1,T}^t \mathbf{G}_T(\psi^E)\mathbf{r}_{1,T} + \frac{1}{s_T}\mathbf{r}_{2,T}^t \mathbf{G}_T(\psi^E)\mathbf{r}_{2,T}\right)\right)^{1/2} \\
&\leq C \sum_{\mathbf{z}\in\mathcal{N}} \sum_{T\in\omega_{\mathbf{z}}} \|\varphi_{\mathbf{z}}\mu_h\|_{L^2(\omega_{\mathbf{z}})} \sqrt{|T|} \left(s_T \mathbf{r}_{1,T}^t \mathbf{G}_T(\psi^E)\mathbf{r}_{1,T} + \frac{1}{s_T}\mathbf{r}_{2,T}^t \mathbf{G}_T(\psi^E)\mathbf{r}_{2,T}\right)^{1/2}.
\end{aligned}$$

A suitable reordering of the sums leads to the final result.

□

3.2.1 Geometric Error Estimate

Besides the interpolation error associated with the function ψ, in this context, we are also interested in fitting the surface Γ "as well as possible". Both the a-priori error estimate (3.5) proposed in [25] and the anisotropic counterpart (3.6) give

a prediction on the error due to the discretization of the function defined on the surface, but they do **not** explicitly contain any geometric contribution.

For this reason, we propose an estimate for the geometrical error. This estimate is essentially heuristic. In more details, since the surface Γ is defined via the zero level set of a signed distance function $d : \mathbb{R}^3 \to \mathbb{R}$, see Section 2.3, we resort to this function d in order to obtain an estimate of the geometrical error associated with the discrete surface Γ_h.

In particular, we identify the geometric error with the quantity $\|d - I_h^E d\|_{L^1(\Gamma)}$. Somehow, we are assuming that the discrete surface Γ_h coincides with the zero level set of the function $I_h^E d$. This identification leads us to provide an anisotropic estimate for the geometric error. Therefore, we can simply replace the function ψ in Proposition 3.2.2 with the signed distance d.

Proposition 3.2.3 *Let $d : \mathbb{R}^3 \to \mathbb{R}$ be the signed distance function associated with the implicit representation of the surface Γ and let I_h^E be the extension of the interpolant I_h in (3.4) according to definition (2.9). Then, if the distance function d is sufficiently regular, i.e., $d \in H^1(U_\delta)$, where $U_\delta \subset \mathbb{R}^3$ as defined in Section 2.3, there exists a constant C such that*

$$\|d - I_h^E d\|_{L^1(\Gamma)} \leq C \sum_{T \in \mathcal{T}_h} \overline{\alpha}_T |T|^{3/2} \xi_T(s_T, \mathbf{r}_{1,T}, d) \tag{3.19}$$

where

$$\xi_T(s_T, \mathbf{r}_{1,T}, d) = \left(s_T (\mathbf{r}_{1,T})^t \overline{\mathbf{G}}_T(d) \mathbf{r}_{1,T} + \frac{1}{s_T} (\mathbf{r}_{2,T})^t \overline{\mathbf{G}}_T(d) \mathbf{r}_{2,T} \right)^{1/2},$$

and with $\overline{\alpha}_T$ and $\overline{\mathbf{G}}_T$ defined as in Proposition 3.2.2.

3.2.2 From the Estimator to an Anisotropic Metric

To drive a mesh adaptation procedure, on one hand we may fix a desired accuracy, i.e., a maximum allowed error; on the other hand we may fix the maximum number of elements in the mesh. Fixed one of these two criteria, we may also decide to "equidistribute the error", i.e., each element contributes to the global error in the same way. The two rules above, combined with an equidistribution criterion, do not necessarily lead to the same adapted mesh.

We are interested in a strategy that minimizes the number of elements while equidistributing the error. Given a positive tolerance τ, an equidistribution of the

interpolation error is obtained by demanding:

$$\overline{\alpha}_T |T|^{3/2} \nu_T(s_T\,, \mathbf{r}_{1,T},\, \psi) = \frac{\tau}{\#\mathcal{T}}\,, \tag{3.20}$$

where $\#\mathcal{T}$ is the number of elements of the current mesh. After identifying the values $\mathbf{r}_{1,T}$ and s_T that verify the Equation (3.20), we may derive a piecewise constant distribution of s_T, $\mathbf{r}_{1,T}$ and $|T|$, and, consequently, a piecewise constant metric to predict the new adapted mesh.

Clearly Equation (3.20) is not sufficient to uniquely determine s_T, $\mathbf{r}_{1,T}$ and $|T|$. To achieve this goal we consequently add another requirement: the condition (3.20) has to be satisfied with the most economical adapted mesh, i.e., with the maximum possible value for the area of the triangle T. This condition is equivalent to demand that Equation (3.20) is satisfied under the constraint that ν_T is minimal.

Proposition 3.2.4 *Consider a fixed function $\psi \in H^1(\Gamma)$. Then, the minimum of the function $\nu_T(s_T,\, \mathbf{r}_{1,T},\, \psi^E)$ is*

$$\nu_{min} = \sqrt{\frac{\mu_1}{\mu_2}\mu_2 + \frac{\mu_2}{\mu_1}\mu_1}\,,$$

and it is reached by choosing the following values for the variables, $\mathbf{r}_{1,T}$, $\mathbf{r}_{2,T}$ and s_T,

$$\mathbf{r}_{1,T} = \mathbf{w}_2\,, \quad \mathbf{r}_{2,T} = \mathbf{w}_1\,, \quad s_T = \sqrt{\frac{\mu_1}{\mu_2}}$$

where $\mathbf{w}_1$, $\mathbf{w}_2$ and μ_1, μ_2 are the eigenvectors and eigenvalues of the matrix $\overline{\mathbf{G}}_T(\psi^E)$, respectively, and $\mu_1 > \mu_2$.

Proof. Since we have fixed a function $\psi \in H^1(\Gamma)$, we adopt the simplified notations $\overline{\mathbf{G}}_T$ and $\nu_T(s, \mathbf{r}_{1,T})$ instead of $\overline{\mathbf{G}}_T(\psi^E)$ and $\nu_T(s,\, \mathbf{r}_{1,T},\, \psi^E)$. We compute the minimum of the function

$$\nu_T(s, \mathbf{r}_{1,T}) = \left(s_T (\mathbf{r}_{1,T})^t\, \overline{\mathbf{G}}_T\, \mathbf{r}_{1,T} + \frac{1}{s_T} (\mathbf{r}_{2,T})^t\, \overline{\mathbf{G}}_T\, (\mathbf{r}_{2,T}) \right)\,, \tag{3.21}$$

with the following constraints:

(i) the vectors $\mathbf{r}_{1,T}$ and $\mathbf{r}_{2,T}$ are normalized, i.e.,

$$\mathbf{r}_{1,T} \cdot \mathbf{r}_{1,T} = \mathbf{r}_{2,T} \cdot \mathbf{r}_{2,T} = 1\,;$$

(ii) the vectors $\mathbf{r}_{1,T}$ and $\mathbf{r}_{2,T}$ are orthogonal, i.e.,

$$\mathbf{r}_{1,T} \cdot \mathbf{r}_{2,T} = \mathbf{r}_{1,T} \cdot \mathbf{r}_{2,T} = 0\,;$$

(iii) the vectors $\mathbf{r}_{1,T}$ and $\mathbf{r}_{2,T}$ are perpendicular to the normal $\mathbf{n}$ to the surface Γ, i.e.,

$$\mathbf{r}_{1,T} \cdot \mathbf{n} = \mathbf{r}_{2,T} \cdot \mathbf{n} = 0\,,$$

So the Lagrangian has the following form:

$$\begin{aligned} L(s_T,\, \mathbf{r}_{1,T},\, \mathbf{r}_{2,T}; \boldsymbol{\lambda}) \;&:=\; \left(s_T (\mathbf{r}_{1,T})^t\, \mathbf{G}_T\, \mathbf{r}_{1,T} + \frac{1}{s_T} (\mathbf{r}_{2,T})^t\, \mathbf{G}_T\, \mathbf{r}_{2,T} \right) + \\ &+\; \underbrace{\lambda_4(\mathbf{r}_{1,T}\mathbf{r}_{1,T} - 1) + \lambda_5(\mathbf{r}_{2,T}\mathbf{r}_{2,T} - 1)}_{(i)} + \\ &+\; \underbrace{\lambda_1(\mathbf{r}_{1,T}\mathbf{r}_{2,T} - 0)}_{(ii)} + \underbrace{\lambda_2(\mathbf{r}_{1,T}\mathbf{n} - 0) + \lambda_3(\mathbf{r}_{2,T}\mathbf{n} - 0)}_{(iii)}\,, \end{aligned} \tag{3.22}$$

where we have defined the following vector:

$$\boldsymbol{\lambda} := (\lambda_1\,, \lambda_2\,, \lambda_3\,, \lambda_4\,, \lambda_5\,)^t\,.$$

Now we proceed with the search of the minimum:

$$\frac{\partial L}{\partial s_T} \;=\; (\mathbf{r}_{1,T})^t\, \mathbf{G}_T\, \mathbf{r}_{1,T} - \frac{1}{s_T^2} (\mathbf{r}_{2,T})^t\, \mathbf{G}_T\, \mathbf{r}_{2,T} = 0\,, \tag{3.23}$$

$$\frac{\partial L}{\partial \mathbf{r}_{1,T}} \;=\; 2\, s_T\, \mathbf{G}_T \mathbf{r}_{1,T} + \lambda_1 \mathbf{r}_{2,T} + \lambda_2 \mathbf{n} + 2\lambda_4 \mathbf{r}_{1,T} = 0\,, \tag{3.24}$$

$$\frac{\partial L}{\partial \mathbf{r}_{2,T}} \;=\; \frac{2}{s_T}\, \mathbf{G}_T \mathbf{r}_{2,T} + \lambda_2 \mathbf{r}_{1,T} + \lambda_3 \mathbf{n} + 2\lambda_5 \mathbf{r}_{2,T} = 0\,. \tag{3.25}$$

If we multiply (3.24) by $(\mathbf{r}_{2,T})^t$ we obtain:

$$\begin{aligned} 0 \;&=\; 2\, s_T\, (\mathbf{r}_{2,T})^t \mathbf{G}_T \mathbf{r}_{1,T} + \lambda_1 (\mathbf{r}_{2,T})^t \mathbf{r}_{2,T} + \lambda_2 (\mathbf{r}_{2,T})^t \mathbf{n} + 2\lambda_4 (\mathbf{r}_{2,T})^t \mathbf{r}_{1,T} = \\ &=\; 2\, s_T\, (\mathbf{r}_{2,T})^t \mathbf{G}_T \mathbf{r}_{1,T} + \lambda_1\,. \end{aligned}$$

We recover a similar result when we multiply (3.25) by $(\mathbf{r}_{1,T})^t$, finally we have the following equalities:

$$2\, s_T\, (\mathbf{r}_{2,T})^t\, \mathbf{G}_T \mathbf{r}_{1,T} + \lambda_1 \;=\; 0\,, \tag{3.26}$$

$$\frac{2}{s_T}(\mathbf{r}_{1,T})^t\,\mathbf{G}_T\mathbf{r}_{2,T} + \lambda_1 \;=\; 0\,. \tag{3.27}$$

Then, we subtract these two equalities to get

$$s_T\,(\mathbf{r}_{2,T})^t\,\mathbf{G}_T\mathbf{r}_{1,T} = \frac{1}{s_T}(\mathbf{r}_{1,T})^t\,\mathbf{G}_T\mathbf{r}_{2,T}\,. \tag{3.28}$$

This equation is verified when $s_T = 1$ or when

$$(\mathbf{r}_{2,T})^t\,\mathbf{G}_T\mathbf{r}_{1,T} = (\mathbf{r}_{1,T})^t\,\mathbf{G}_T\mathbf{r}_{2,T} = 0\,. \tag{3.29}$$

If $\mathbf{r}_{1,T}$ and $\mathbf{r}_{2,T}$ are the eigenvectors of $\mathbf{G}_T$, Equation (3.29) is verified. Moving from Equation (3.23), we are able to find s_T:

$$\begin{aligned}
(\mathbf{r}_{1,T})^t\,\mathbf{G}_T\,\mathbf{r}_{1,T} - \frac{1}{s_T^2}(\mathbf{r}_{2,T})^t\,\mathbf{G}_T\,\mathbf{r}_{2,T} &= 0 \\
s_T^2\,(\mathbf{r}_{1,T})^t\,\mathbf{G}_T\,\mathbf{r}_{1,T} &= (\mathbf{r}_{2,T})^t\,\mathbf{G}_T\,\mathbf{r}_{2,T} \\
s_T &= \sqrt{\frac{(\mathbf{r}_{2,T})^t\,\mathbf{G}_T\,\mathbf{r}_{2,T}}{(\mathbf{r}_{1,T})^t\,\mathbf{G}_T\,\mathbf{r}_{1,T}}}\,.
\end{aligned}$$

Since $s_T \geq 1$, we choose the eigenvectors associated to the minimum and the maximum eigenvalues as $\mathbf{r}_{1,T}$ and $\mathbf{r}_{2,T}$, i. e.,

$$\begin{aligned}
\mathbf{w}_1 &= \mathbf{r}_{2,T}\,, \quad \text{with} \quad \mathbf{G}_T\,\mathbf{w}_1 = \mu_1\,\mathbf{w}_1\,, \\
\mathbf{w}_2 &= \mathbf{r}_{1,T}\,, \quad \text{with} \quad \mathbf{G}_T\,\mathbf{w}_2 = \mu_2\,\mathbf{w}_2\,,
\end{aligned}$$

where $\mu_1 \geq \mu_2$, we obtain

$$s_T = \sqrt{\frac{(\mathbf{r}_{2,T})^t\,\mathbf{G}_T\,\mathbf{r}_{2,T}}{(\mathbf{r}_{1,T})^t\,\mathbf{G}_T\,\mathbf{r}_{1,T}}} = \sqrt{\frac{\mu_1}{\mu_2}}\,. \tag{3.30}$$

Finally, moving from Equation (3.30) and (3.29), we have found a stationary point for the function ν_T satisfying the constraints (i), (ii) and (iii). This procedure leads us to identify a stationary point for ν_T corresponding to the following choice of the variables:

$$\mathbf{r}_{1,T} = \mathbf{w}_2\,, \quad \mathbf{r}_{2,T} = \mathbf{w}_1 \quad \text{and} \quad s_T = \sqrt{\frac{\mu_1}{\mu_2}}\,. \tag{3.31}$$

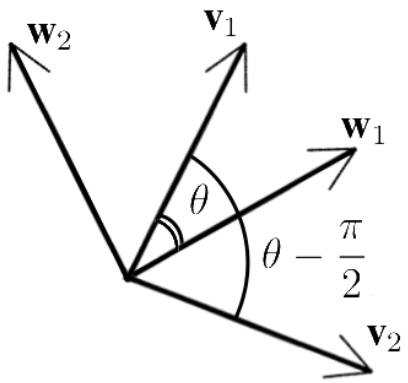


Figure 3.2: Vectors $\mathbf{v}_1$, $\mathbf{v}_2$ and eigenvectors $\mathbf{w}_1$, $\mathbf{w}_2$.

Now, we show that this stationary point is a minimum for the function ν_T. We consider two unitary vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ such that

$$\mathbf{v}_1 \cdot \mathbf{v}_2 = 0 \quad \text{and} \quad \mathbf{v}_1 \cdot \mathbf{n} = \mathbf{v}_2 \cdot \mathbf{n} = 0\,.$$

It is possible to decompose these vectors along the direction of the eigenvectors of $\mathbf{G}_T$, $\mathbf{w}_1$ and $\mathbf{w}_2$. In particular, if we call θ the angle between $\mathbf{v}_1$ and $\mathbf{w}_1$, see Figure 3.2, we have

$$\mathbf{v}_1 = \mathbf{w}_1 \cos\theta + \mathbf{w}_2 \sin\theta\,,$$

and

$$\mathbf{v}_2 = \mathbf{w}_1 \cos\left(\theta - \frac{\pi}{2}\right) + \mathbf{w}_2 \sin\left(\theta - \frac{\pi}{2}\right) = \mathbf{w}_1 \sin\theta - \mathbf{w}_2 \cos\theta\,.$$

Now, we fix

$$s_T = \sqrt{\frac{\mu_1}{\mu_2}} \geq 1\,,$$

and we analyse the behaviour of the function ν_T choosing

$$\mathbf{r}_{1,T} = \mathbf{v}_1 \quad \text{and} \quad \mathbf{r}_{2,T} = \mathbf{v}_2\,.$$

so

$$\nu_T(s, \mathbf{v}_1, \mathbf{v}_2) = \sqrt{\frac{\mu_1}{\mu_2}}(\mathbf{v}_1)^t\, \mathbf{G}_T\, \mathbf{v}_1 + \sqrt{\frac{\mu_2}{\mu_1}}(\mathbf{v}_2)^t\, \mathbf{G}_T\, (\mathbf{v}_2)$$

$$
\begin{aligned}
&= \sqrt{\frac{\mu_1}{\mu_2}}\left(\mathbf{w}_1\cos\theta+\mathbf{w}_2\sin\theta\right)^t\ \mathbf{G}_T\ \left(\mathbf{w}_1\cos\theta+\mathbf{w}_2\sin\theta\right)+\\
&+ \sqrt{\frac{\mu_2}{\mu_1}}\left(\mathbf{w}_1\sin\theta-\mathbf{w}_2\cos\theta\right)^t\ \mathbf{G}_T\ \left(\mathbf{w}_1\sin\theta-\mathbf{w}_2\cos\theta\right)=\\
&= \sqrt{\frac{\mu_1}{\mu_2}}(\mathbf{w}_1\,\mathbf{G}_T\,\mathbf{w}_1\cos^2\theta+\mathbf{w}_1\,\mathbf{G}_T\,\mathbf{w}_2 2\cos\theta\sin\theta+\mathbf{w}_2\,\mathbf{G}_T\,\mathbf{w}_2\sin^2\theta)+\\
&+ \sqrt{\frac{\mu_2}{\mu_1}}(\mathbf{w}_1\,\mathbf{G}_T\,\mathbf{w}_1\sin^2\theta+\mathbf{w}_1\,\mathbf{G}_T\,\mathbf{w}_2 2\cos\theta\sin\theta+\mathbf{w}_2\,\mathbf{G}_T\,\mathbf{w}_2\cos^2\theta)=\\
&= \sqrt{\frac{\mu_1}{\mu_2}}(\mathbf{w}_1\,\mathbf{G}_T\,\mathbf{w}_1\cos^2\theta+\mathbf{w}_2\,\mathbf{G}_T\,\mathbf{w}_2\sin^2\theta)+\\
&+ \sqrt{\frac{\mu_2}{\mu_1}}(\mathbf{w}_1\,\mathbf{G}_T\,\mathbf{w}_1\sin^2\theta+\mathbf{w}_2\,\mathbf{G}_T\,\mathbf{w}_2\cos^2\theta)=\\
&= \sqrt{\frac{\mu_1}{\mu_2}}(\mu_1\cos^2\theta+\mu_2\sin^2\theta)+\sqrt{\frac{\mu_2}{\mu_1}}(\mu_1\sin^2\theta+\mu_2\cos^2\theta)\,.
\end{aligned}
$$

At this level we observe that function ν_T depends only on the angle θ:

$$
\nu_T(\theta)=\sqrt{\frac{\mu_1}{\mu_2}}(\mu_1\cos^2\theta+\mu_2\sin^2\theta)+\sqrt{\frac{\mu_2}{\mu_1}}(\mu_1\sin^2\theta+\mu_2\cos^2\theta)\,. \tag{3.32}
$$

We compute the first order derivative of ν_T with respect to θ:

$$
\begin{aligned}
\frac{d\nu_T}{d\theta} &= \sqrt{\frac{\mu_1}{\mu_2}}(-2\mu_1\cos\theta\sin\theta+2\mu_2\sin\theta\cos\theta)+\\
&+ \sqrt{\frac{\mu_2}{\mu_1}}(2\mu_1\sin\theta\cos\theta-2\mu_2\cos\theta\sin\theta)\\
&= \sqrt{\frac{\mu_1}{\mu_2}}(\mu_2-\mu_1)\sin(2\theta)+\sqrt{\frac{\mu_2}{\mu_1}}(\mu_1-\mu_2)\sin(2\theta)=\\
&= (\mu_1-\mu_2)\left(\sqrt{\frac{\mu_2}{\mu_1}}-\sqrt{\frac{\mu_1}{\mu_2}}\right)\sin(2\theta)\,,
\end{aligned}
$$

We compute the second order derivative

$$
\frac{d^2\nu_T}{d\theta^2}=(\mu_1-\mu_2)\left(\sqrt{\frac{\mu_2}{\mu_1}}-\sqrt{\frac{\mu_1}{\mu_2}}\right)2\cos(2\theta)\,. \tag{3.33}
$$

If we take

$$
\theta=\frac{\pi}{2}\,,
$$

we have chosen the stationary point of Equation (3.31), and we have

$$\frac{d^2\nu_T}{d\theta^2}\left(\frac{\pi}{2}\right) = \underbrace{(\mu_1-\mu_2)}_{>0}\underbrace{\left(\sqrt{\frac{\mu_2}{\mu_1}}-\sqrt{\frac{\mu_1}{\mu_2}}\right)}_{<0}\underbrace{2\cos(\pi)}_{=-2} > 0\,.$$

So the stationary point

$$\mathbf{r}_{1,T} = \mathbf{w}_2\,, \quad \mathbf{r}_{2,T} = \mathbf{w}_1\,, \quad s = \sqrt{\frac{\mu_1}{\mu_2}}\,,$$

is a minimum for ν_T

□

Thanks to Proposition 3.2.4, we are able to obtain a piecewise constant metric, i.e., to find the quantities σ_1, σ_2, $\mathbf{u}_1$, $\mathbf{u}_2$ and $\mathbf{n}$ in Equation (1.14). Based on this result, we come back to the equidistribution requirement in Equation (3.20). Then we find the pair $s_{1,T}$ and $s_{2,T}$ that verifies the following relations:

$$\left(\frac{s_{1,T}s_{2,T}}{2}\right)^{3/2}\overline{\alpha}_T\nu_{\min} = \frac{\tau}{\#\mathcal{T}}\,, \tag{3.34}$$

$$\frac{s_{1,T}}{s_{2,T}} = \sqrt{\frac{\mu_1}{\mu_2}}\,, \tag{3.35}$$

where we have exploited the result $s_{1,T}s_{2,T} = 2|T|$. After some simple algebraic operations, we finally get:

$$s_{1,T} = \sqrt[6]{\frac{\tau^2}{\#\mathcal{T}\,\overline{\alpha}_T^2\,\nu_{\min}}}\sqrt[4]{\frac{\mu_1}{\mu_2}}, \qquad s_{2,T} = \sqrt[6]{\frac{\tau^2}{\#\mathcal{T}\,\overline{\alpha}_T^2\,\nu_{\min}}}\sqrt[4]{\frac{\mu_2}{\mu_1}}\,.$$

In this way we can define the piecewise constant metric, by setting

$$\sigma_1 = s_{1,T}\,, \quad \sigma_2 = s_{2,T}\,, \quad \mathbf{u}_1 = \mathbf{w}_2 \quad \text{and} \quad \mathbf{u}_2 = \mathbf{w}_1\,, \tag{3.36}$$

in Equation (1.14).

From the metric to the edge length

For a fixed tolerance τ, we are able to predict metric piecewise constant on each triangular element, see Equation (3.36), starting from the interpolation error estimator in Equation (3.17). To predict the edge length according to the metric field, we define the metric at each node of the mesh. In particular, if we consider a node $\mathbf{v}$ of the mesh, we compute the metric at this node by averaging the metric associated with the triangles sharing $\mathbf{v}$. Once we have this metric, we can use Equation (1.13) to easily predict the edge length.

After that a local modification is done, the metric of the nodes involved in the operation is properly modified. For instance, when we halve an edge or we contract an edge onto the corresponding middle-point, we define the metric at this new point by averaging the metrics at the end-points of this edge.

Local Operations

The idea behind the metric adaptation procedure is that the length of each edge measured in the new metric should be equal to 1, i.e., $||e||_M \approx 1$ for any edge of the new mesh.

In more details, to guarantee this condition, we accordingly modify all the mesh operations described in Section 1.2.

Edge Flipping

Edge flipping considers the quadrilateral formed by two adjacent triangles and it changes the shared edge with the other diagonal, see Figure 3.3. When we are dealing with triangular meshes and especially when we are considering surface triangular meshes, this kind of operation presents some drawbacks. In fact, it may lead to undesired artefacts, see Subsection 1.2.1.

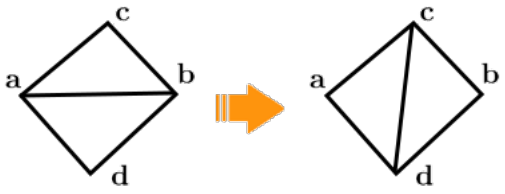


Figure 3.3: Flipping of the edge **ab**.

After guaranteeing the consistency of this operation on an edge $\mathbf{ab}$, we decide to perform the flipping of this edge moving from its length computed in the predicted metric field, $||\mathbf{ab}||_M$. More precisely, if the new edge $\mathbf{cd}$ is closer to the optimal length with respect to the edge $\mathbf{ab}$, i.e.,

$$\left| ||\mathbf{ab}||_M - 1 \right| > \left| ||\mathbf{cd}||_M - 1 \right|,$$

we flip the edge $\mathbf{ab}$.

In this way we build a new edge that has a length closer to the optimal value. Moving from this criterion, we have properly modified the Lawson's Flip Algorithm described in Subsection 1.2.1 to improve the length of the edges.

Edge Splitting

Edge splitting is one of the principal mesh modification operations to improve the surface approximation in localized areas of the domain. In this framework, the metric M identifies too long edges. We compute the length of an edge according to the predicted metric field, and, if $||e||_M \gg 1$, we halve the edge by inserting a new vertex at the corresponding middle point.

Since we are dealing with a not flat surface triangular mesh, the middle point of the edge e may not provide a good approximation of the surface itself. To avoid this drawback, we move the new inserted point on the surface via a suitable projection algorithm, see [58].

After this insertion is performed, we run the FLIPEDGES routine on the edges connected to the new point to locally improve the length of the edges involved in the splitting.

Edge Contraction

As described in Subsection 1.2.3, this can be considered as the inverse operation with respect to the edge splitting. Moreover, since it reduces both the number of elements and number of vertices of the mesh, edge contraction is a crucial operation to reduce the computational effort of a numerical simulation.

The metric field M identifies too short edges, i.e., the edges of the mesh such that $||e||_M \ll 1$. Then we contract the edge e into the projection of the middle point of the edge e onto the actual surface. Unfortunately, we cannot always apply this operation because it may lead to topological invalid configurations, see Subsection 1.2.3.

After a contraction is performed, we run the FLIPEDGES routine on the new edges in order to improve their length.

Node Smoothing

Smoothing is one of the classical methods to modify a mesh. As we have seen in Subsection 1.2.4, this operation is in contrast with the ones described above, since it does not modify the topology of the mesh, but it simply moves the nodes in a new position.

It has a physical interpretation. Given a point **v**, the edges connected to this point may be seen as a system of springs: the smoothing procedure aims at finding a new position of **v**, that minimizes the elastic energy of the system.

In the isotropic adaptation framework, the smoothing moves **v** in the barycenter of the patch of the triangles sharing the vertex **v**, see Figure 3.4. But, since we are dealing with a metric field, we have to take into account the effects of the distance distortion due to the metric. This means that the new location does not necessary coincide with the barycenter of the polygon.

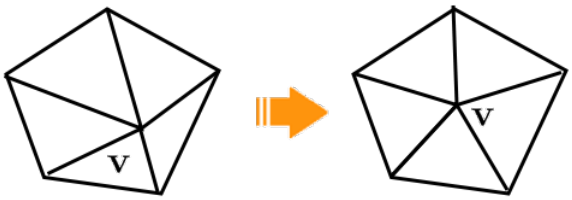


Figure 3.4: Smoothing of the node **v**.

The new position of the point **v** could lead to an invalid topological configuration, such as inverted triangles. As a consequence, we need to check the validity of the new configuration, see Subsection 1.2.4.

Once we have moved the point **v**, we consider all the edges connected to **v** and we run the FLIPEDGES algorithm to make the length of these new edges as close as possible to the optimal value.

The Adaptation Procedure

To get the adapted mesh that satisfies the constraint $||e||_M \approx 1, \forall e \in \mathcal{E}$, where $\mathcal{E}$ is the skeleton of the starting mesh Γ_h, we propose the iterative procedure described in Algorithm 2 that combines the previous local mesh operations. The inputs of this algorithm are:

- the actual surface Γ, given via an implicit function d;
- an initial mesh Γ_h;
- the tolerance τ;
- the function ψ defined on the surface Γ;
- the number of iterations, I and the number of smoothing iterations, K.

Algorithm 2 The mesh adaptation procedure

IMPROVEMETRICBASED(τ, Γ, Γ_h, ψ, I, K)

set $\Gamma_h^0 = \Gamma_h$;
compute associated the metric field M^0 according to τ;
i=0;
for $i < I$ **do**
  split all the edges such that $||e||_{M^i} > 1.5$ + FLIPEDGES localized;
  call the FLIPEDGES routine for all the edges;
  contract all the edges such that $||e||_{M^i} < 0.5$ + FLIPEDGES localized;
  call the FLIPEDGES routine for all the edges;
  $k = 0$;
  for $k < K$ **do**
    smooth all the vertices of the mesh + FLIPEDGES localized;
    call the FLIPEDGES routine for all the edges;
    $k = k + 1$;
  end for
  compute M^{i+1} moving from the error estimator computed and the adapted mesh Γ_h^i and τ;
  $\Gamma_h^{i+1} = \Gamma_h^i$;
  $i = i + 1$;
end for

We realize that the FLIPEDGES algorithm plays a significant role in the adaptation procedure. More precisely, this algorithm is employed to locally improve the mesh, lines 5,7 and 11, and to globally improve the edges length, line 6,8 and 12.

At the end of each iteration of the adaptation procedure, line 15, we have modified the triangular mesh Γ_h^i. Moving from this new adapted mesh, we compute the new metric field M^{i+1}, line 15, then we are ready for a new step of the adaptation procedure, line 5-14.

Moreover, we may recognize two main phases:

- the first one, from line 5 to line 8, modifies the mesh via edge splitting, line 5, and edge contraction, line 7. This step can be understood as a sampling phase where we increase or decrease the density of the mesh elements;

- the second phase, from line 10 to 14, can be identified as an optimization phase, since we do not add or remove any point of mesh, but we simply enhance the quality of the mesh via smoothing and flipping operations.

Merging the Discretization with the Geometric Estimators

Consider a function $\psi \in H^1(\Gamma)$ and a signed distance function d whose zero-level set coincides with the surface Γ. Starting from Proposition 3.2.2 and 3.2.3, we may associate with each triangle T two different quantities,

$$e_{T,\text{int}} := \overline{\alpha}_T |T|^{3/2} \nu_T(s_T\,, \mathbf{r}_{1,T}\,, \psi^E) \quad \text{and} \quad e_{T,\text{geo}} := \overline{\alpha}_T |T|^{3/2} \xi_T(s_T\,, \mathbf{r}_{1,T}\,, d).$$

The former provides a prediction of the error due to the interpolation of the function ψ defined on the surface; the latter furnishes a prediction of the error due to the approximation of the surface Γ with the mesh Γ_h. This means that we are able to define two different metrics that, separately, may drive an adaptation procedure to reduce the corresponding error. However, we are interested in reducing both $e_{T,\text{int}}$ and $e_{T,\text{geo}}$. So, the idea is to build a suitable unique metric that combines these two informations.

In the following, we analyse these three different strategies:

a) **intersection of metrics:** build a new metric that merges the metric associated with $e_{T,\text{int}}$ and $e_{T,\text{geo}}$;

b) **"maximum" metric:** for each element T, we consider the metric associated with the maximum value between $e_{T,\text{int}}$ and $e_{T,\text{geo}}$;

c) **weighted sum:** we build a new piecewise constant metric

$$M_{\text{glob}} = \alpha M_{\text{int}} + (1-\alpha) M_{\text{geo}} \,,$$

by considering a convex combination of the metrics M_{int} and M_{geo} defined by the local estimator $e_{T,\text{int}}$ and $e_{T,\text{geo}}$, respectively, with $\alpha \in (0,1)$ balancing the contribution of these two metrics on the global metric M_{glob}.

3.2.3 Numerical Results

Here we analyse reliability of the adaptive procedure introduced in the previous sections. We consider an initial mesh and we build four different metrics to drive the mesh adaptation:

- a metric derived only from the estimator in Equation (3.17) denoted by INT;
- intersection of metrics;
- the "maximum" metric;
- a weighted sum of metrics by choosing with $\alpha = 0.3$.

We use the adaptation procedure described in Algorithm 2, where we fix $I = 10$ and $K = 5$. Then, for each example, we consider:

a) the *number of the elements*: we look for meshes with a reduced number of elements to contain the computational effort.

b) the *stretching factor*: values of s_T close to 1 mean that the triangle T has a shape similar to the equilateral one, while high values of s_T refer to very stretched elements. To evaluate the level of anisotropy of Γ_h, we consider the maximum value for the stretching factor

$$s_{\text{max}} := \max_{T \in \mathcal{T}} s_T \,, \tag{3.37}$$

where $\mathcal{T}$ is the set of the triangles in Γ_h.

c) *the values of the error*: since we are dealing with an a-priori error analysis, we are able to compute the exact value of the quantity we are estimating on each triangle, i.e., the norm

$$||\psi^E - I_h \psi_h^E||_{L^1(T)} \,.$$

In particular, we compute the errors

$$e_{\text{tot}} = \sum_{T\in\mathcal{T}} ||\psi^E - \psi_h^E||_{L^1(T)} \quad \text{and} \quad e_{\max} = \max_{T\in\mathcal{T}} ||\psi^E - \psi_h^E||_{L^1(T)}\,. \quad (3.38)$$

d) the *isotropic case*: we evaluate e_{tot} and $e_{\max}$ also in the case of an isotropic mesh adaptation to compare the performance of these two adaptation processes;

e) the *geometrical distance*: to evaluate the mismatch between the polyhedral surface Γ_h of the actual surface Γ, for each adapted mesh we compute the quantity

$$d_{\max} = \max_{T\in\mathcal{T}} |\mathbf{p}_T - \mathbf{b}_T|\,, \quad (3.39)$$

where $\mathbf{b}_T$ is the barycenter of the triangle T, $\mathbf{p}_T$ is the projection of $\mathbf{b}_T$ on the surface Γ and $|\cdot|$ is the standard Euclidean norm.

We numerically verify the better "error-vs-number of elements" behaviour of the anisotropic meshes with respect to the corresponding isotropic case. In particular, if we fix about the same number of elements, we obtain a value for e_{tot} in general lower on the anisotropic mesh than the one computed on the corresponding isotropic grid. Vice-versa, we usually get about the same value of e_{tot} with a lower number of elements in the anisotropic adapted mesh than in the isotropic case.

The tolerances, τ, driving the adaptation procedure are specified in the tables below.

Example 1

We consider the surface Γ_1 defined by the following signed distance function

$$d_1 : [0,1]\times[0,1]\times[-0.2,0.2] \to \mathbb{R}\,,$$

such that

$$d_1(x,y,z) := 0.2\cos(\pi x)\cos(\pi y) - z\,,$$

and the function $f_1 : \Gamma_1 \to \mathbb{R}$:

$$f_1(x,y,z) := 4y(1-y)(1-e^{-ax} - (1-e^{-ax})x)\,.$$

In this test case we set $a = 1000$.

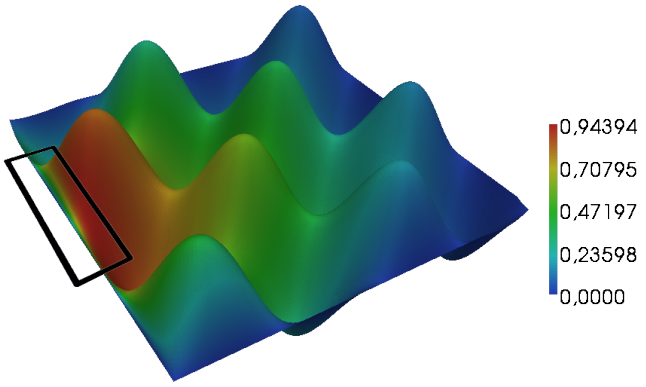


Figure 3.5: Surface Γ_1 and the function f_1, the boundary layer is highlighted by the black rectangle.

Function f_1 is characterized by a boundary layer along the y axes, see the highlighted zone in Figure 3.5.

As it was expected, in the anisotropic adapted meshes the triangles are aligned along the direction of the boundary layer, see Figure 3.6 and this behaviour becomes more evident from the details in Figure 3.7. In both these figures, we provide the final adapted meshes yielded by the four different metrics itemized above.

When we consider the mesh generated via the intersection of metrics, the directional features of the solution are not so clearly detected as with the other approaches, see Figure 3.6(d). In Table 3.3 we give a more quantitative evaluation of the triangle distortion in the new adapted meshes. These data confirm that the mesh obtained via the intersection of metrics is not so anisotropic. Since we are interested in anisotropic mesh adaptation, we neglect hereafter the adaptation strategy driven by the intersection of metric approach.

Moving from Table 3.1 and 3.2, we numerically verify the better "error-vs-number of elements" behaviour characterizing the anisotropic meshes. More precisely, in Table 3.1, we fix about the same number of elements and we show that, for each adapted mesh, we get a better accuracy in the anisotropic case. Vice-versa, in Table 3.2, we consider about the same accuracy on the interpolation error and we observe that fewer elements are enough in the anisotropic cases.

Finally, in Table 3.4 we collect the values of $d_{\max}$. Moving from these data, we may appreciate the effect due to the combination of the metrics associated with

$e_{T,\text{int}}$ and $e_{T,\text{geo}}$. Indeed, the adaptations based on the "maximum metric" and the weighted sum lead to a lower value for $d_{\max}$ with respect to the one provided by a mesh adaptation based only on the control of the interpolation error.

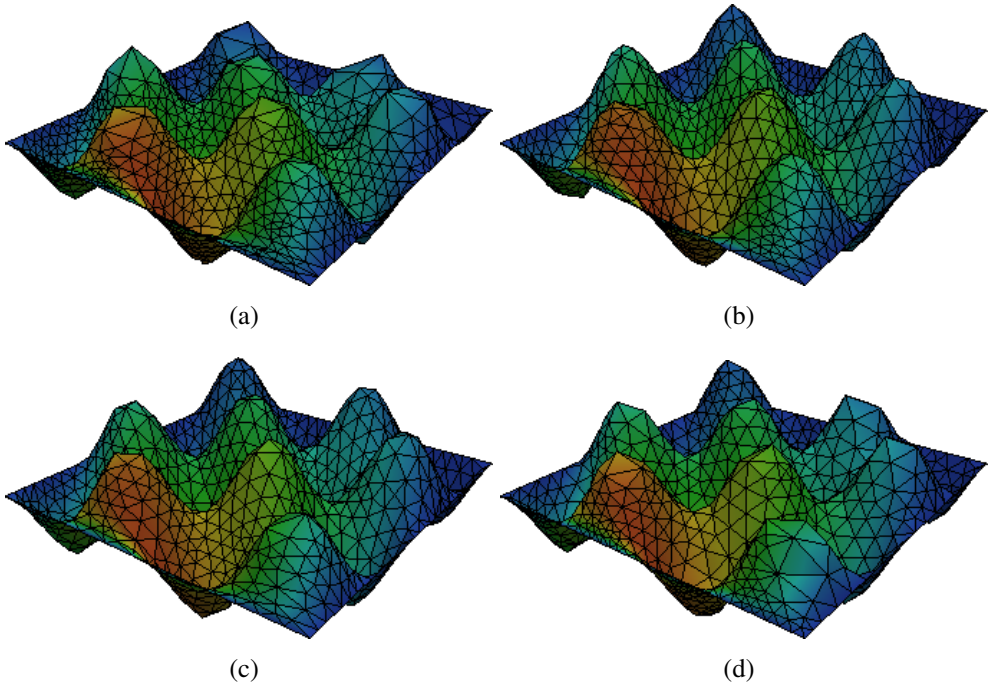
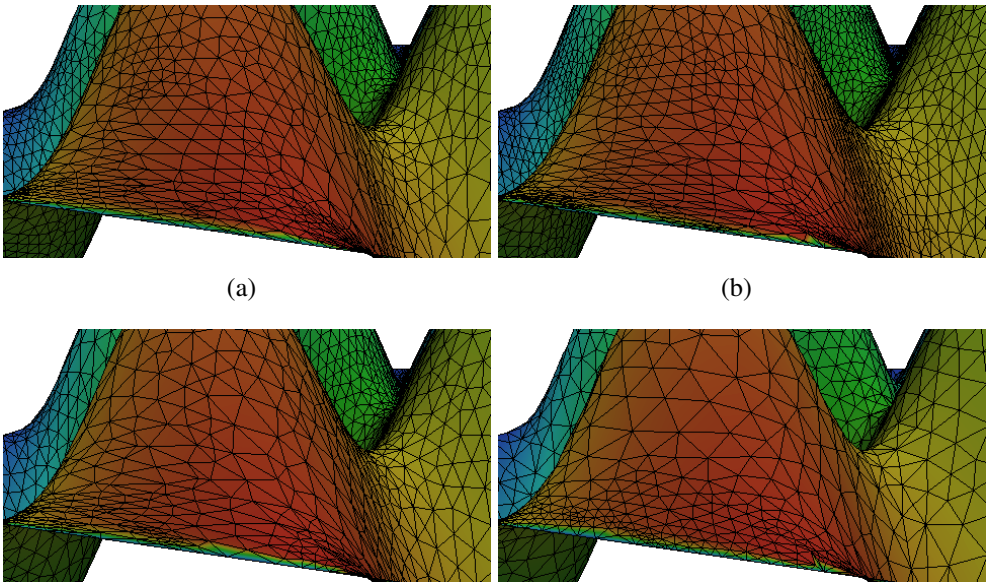


Figure 3.6: Adapted mesh driven by the INT estimator, (a), the "maximum" metric, (b), the weighted sum, (c) and intersection of metrics, (d).

	INT		"maximum" metric		weighted sum	
	iso	ani	iso	ani	iso	ani
Ele.	2664	2618	3483	3434	2952	2925
e_{tot}	3.654e-03	1.195e-03	3.648e-03	9.160e-04	4.334e-03	1.392e-03
$e_{\max}$	6.431e-05	3.135e-05	6.566e-05	1.669e-05	1.027e-04	5.424e-05
τ	2.000e-05	2.700e-05	2.000e-05	2.700e-05	2.000e-05	2.700e-05

Table 3.1: Comparison between isotropic and anisotropic meshes with about the same number of elements.



(a) (b)

(c) (d)

Figure 3.7: Details of the adapted mesh driven by the INT estimator, (a), the "maximum" metric, (b), the weighted sum, (c) and intersection of metrics, (d).

	INT		"maximum" metric		weighted sum	
	iso	ani	iso	ani	iso	ani
Ele.	11479	2009	18201	2024	16548	2031
e_{tot}	3.168e-03	3.529e-03	3.159e-03	3.373e-03	3.049e-03	3.441e-03
e_{max}	6.557e-05	1.814e-04	6.619e-05	2.089e-04	7.004e-05	1.651e-04
τ	0.500e-06	0.800e-03	0.500e-06	0.700e-06	0.500e-06	0.400e-06

Table 3.2: Comparison between isotropic and anisotropic meshes with about the same accuracy.

case	$s_{\max}$
INT	2.268e+01
intersection	3.586e+00
"maximum" metric	2.805e+01
weighted sum	3.697e+01

Table 3.3: Maximum stretching factor for the different anisotropic adaptation strategies.

case	$d_{\max}$
INT	2.078e-02
"maximum" metric	7.785e-03
weighted sum	8.537e-03

Table 3.4: Quantity defined in Equation (3.39) for three different strategies.

In Figure 3.8 left, we show the convergence rate of the estimate for both the interpolation and the geometric errors. As expected according to the results in the literature, see, e.g., [19], we have that the geometric error exhibits a convergence rate higher with respect to the one characterizing the interpolation error. On the other hand, in Figure 3.8 right, we compare the convergence trend of the error associated with the three types of metric combination we have assessed. The rate is very similar for the three approaches, with a slightly better trend for the weighted sum of metrics.

Example 2

We consider the surface Γ_2 defined by the signed distance function,

$$d_2 \,:\, [-1.5, 1.5] \,\times\, [-1.5, 1.5] \,\times\, [-0.5, 0.5] \,\to\, \mathbb{R}\,,$$

such that

$$d_2(x, y, z) := \left(r_1 - \sqrt{x^2 + y^2}\right)^2 + z^2 - r_0\,,$$

where $r_0 = 0.25$, $r_1 = 1$ and the function $f_2 : \Gamma_2 \to \mathbb{R}$ given by

$$f_2(x, y, z) := \tanh\left(40x\right),$$

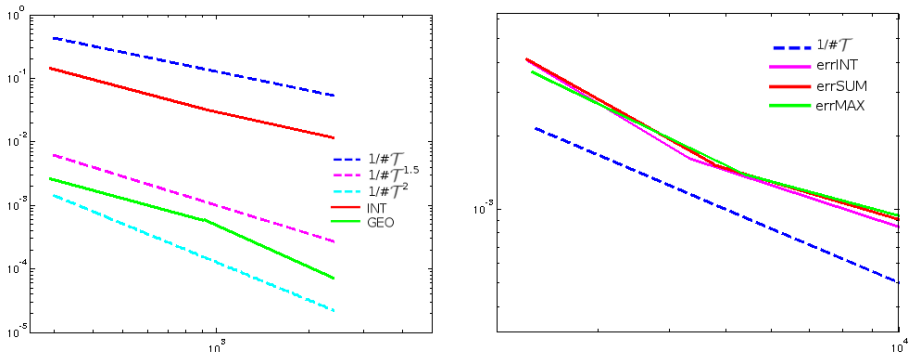


Figure 3.8: Convergence rate of the estimate for the interpolation (INT) and the geometric (GEO) errors, left. Convergence trend of the error associated with the three types of metric combination: intersection (errINT), weighted sum (errSUM) and "maximum" (errMAX) metric, right.

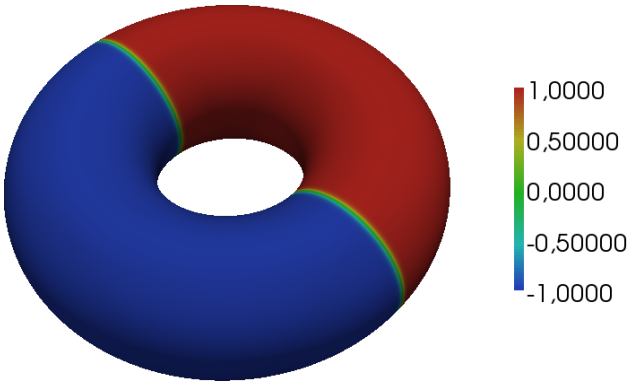


Figure 3.9: Surface Γ_2 and the function f_2.

see Figure 3.9.

In Figure 3.10, we appreciate the anisotropic nature of the adapted meshes yielded by the four adaptive strategies introduced above: the elements are perfectly aligned to follow the boundary layers. This is confirmed by the values in Table 3.5, where we collect the maximum stretching factor for each mesh. Even for this example, we can state that the intersection strategy does not create a really anisotropic mesh. In fact, the value $s_{\max}$ is lower compared with the other approaches.

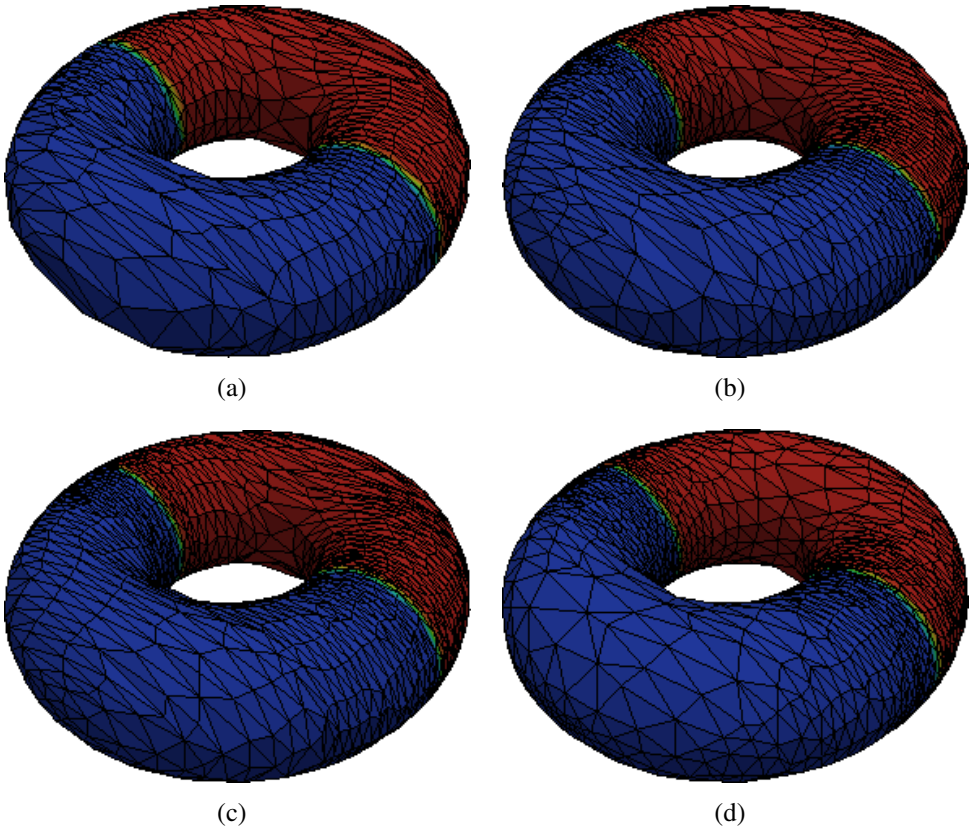
case	$s_{\max}$
INT	6.176e+01
intersection	9.967e+00
"maximum" metric	3.105e+01
weighted sum	2.601e+01

Table 3.5: Maximum stretching factor for the different anisotropic adaptation strategies.

Table 3.6 and 3.7 numerically confirm the advantages lead by anisotropic meshes with respect to the isotropic grids. In particular, if we consider about the same number of elements, we have lower value for the total error e_{tot} in the anisotropic case, see Table 3.6. On the contrary, we get about the same total error, e_{tot}, with a fewer elements in the anisotropic case, see Table 3.7.

	INT		"maximum" metric		weighted sum	
	isotropic	anisotropic	isotropic	anisotropic	isotropic	anisotropic
Ele.	3376	3640	3672	3652	1548	1660
e_{tot}	1.901e-01	8.884e-02	1.864e-01	8.887e-02	5.820e-01	3.463e-01
$e_{\max}$	2.237e-03	1.646e-03	1.848e-02	1.641e-03	1.648e-02	8.074e-03
τ	0.500e-01	3.000e+00	0.400e-01	3.000e+00	1.000e+00	4.000e+00

Table 3.6: Comparison between isotropic and anisotropic meshes with about the same number of elements.



(a) (b)

(c) (d)

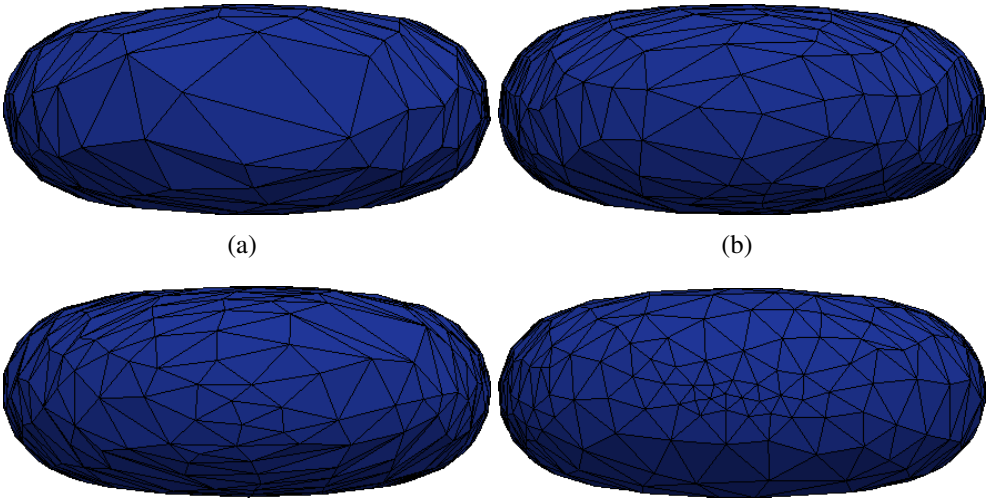
Figure 3.10: Adapted mesh driven by the INT estimator, (a), the "maximum" metric, (b), the weighted sum, (c) and intersection of metrics, (d).

	INT		"maximum" metric		weighted sum	
	isotropic	anisotropic	isotropic	anisotropic	isotropic	anisotropic
Ele.	11030	4732	11030	4746	8162	4084
e_{tot}	6.327e-02	6.490e-02	6.327e-02	6.420e-02	8.656e-02	8.733e-02
e_{max}	4.212e-04	1.019e-03	4.212e-04	1.019e-03	7.884e-04	1.775e-03
τ	0.500e-02	1.500e+00	0.500e-02	1.500e+00	0.650e-02	1.000e+00

Table 3.7: Comparison between isotropic and anisotropic meshes with about the same accuracy.

The adapted mesh, that takes into account only the interpolation error, does

not fit the surface as well as the other approaches that combine the interpolation and geometric information do. This behaviour is evident from the details in Figure 3.11, corresponding to a region where the function f_2 is constant. The adaptation driven only by the interpolation information does not refine this curved region, Figure 3.11(a), while the other three approaches lead to a refinement of the mesh, see Figure 3.11(b), 3.11(c) and 3.11(d). In Table 3.8 we numerically verify this behaviour.



(a) (b)

(c) (d)

Figure 3.11: Details of the adapted mesh $\Gamma_{2,h}$ with only INT 3.11(a), "maximum" metric 3.11(b), weighted sum 3.11(c) and intersection of metrics 3.11(d) strategies.

case	$d_{\max}$
INT	1.168e-01
"maximum" metric	2.359e-02
weighted sum	2.259e-02

Table 3.8: Quantity defined in Equation (3.39) for three different strategies.

Concerning the convergence rate analysis, we can essentially repeat the considerations done on the first test case.

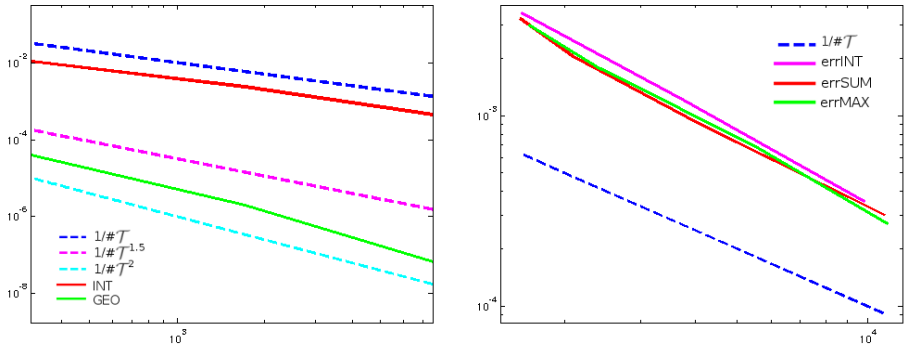


Figure 3.12: Convergence rate of the estimate for the interpolation (INT) and the geometric (GEO) errors, left. Convergence trend of the error associated with the three types of metric combination: intersection (errINT), weighted sum (errSUM) and "maximum" (errMAX) metric, right.

3.3 An Anisotropic A-Posteriori Error Estimator for the Energy Norm of the Laplace-Beltrami Problem

In this Section we deal with an anisotropic a-posteriori error analysis for the Laplace-Beltrami problem:

$$\begin{cases} -\Delta_\Gamma u &= f \quad \text{on } \Gamma\,, \\ u &= 0 \quad \text{on } \partial\Gamma\,, \end{cases} \tag{3.40}$$

where Γ is an arbitrary two-dimensional surface embedded in $\mathbb{R}^3$, and $-\Delta_\Gamma$ denotes the Lapalce-Beltrami operator. In Subsection 2.3.1, we have provided the weak formulation of Problem 2.3.1 and the corresponding discretization.

Before dealing with the anisotropic error analysis, we recall Lemma 3.3.1 in [40], and two useful inequalities in [31]. Then, we prove two preliminary results in Lemma 3.3.4 and Proposition 3.3.5, that play a key role in the proof of the desired anisotropic error estimator.

Definition 3.3.1 *Given a matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$, the matrix $\mathbf{B} \in \mathbb{R}^{m\times n}$ is the generalized inverse of the matrix A, if it satisfies the conditions:*

$$\mathbf{ABA} \;=\; \mathbf{A}\,,$$

$$\begin{aligned} \mathbf{BAB} &= \mathbf{B}\,, \\ (\mathbf{AB})^t &= \mathbf{AB}\,, \\ (\mathbf{BA})^t &= \mathbf{BA}\,. \end{aligned}$$

Definition 3.3.2 *Given a matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$, the norm of A is defined as follow:*

$$||\mathbf{A}|| := \max\left\{ ||\mathbf{A}\boldsymbol{v}|| \,:\, \boldsymbol{v} \in \mathbb{R}^m \, ||\boldsymbol{v}|| = 1 \right\}. \tag{3.41}$$

Proposition 3.3.1 *Consider a non degenerate triangle $T \subset \mathbb{R}^3$ and the reference triangle $\hat{T} \in \mathbb{R}^2$. Let $F_T : \hat{T} \to T$ be the affine transformation defined as:*

$$\mathbf{x} = \boldsymbol{M}_T\hat{\mathbf{x}} + \mathbf{b}_T\,, \tag{3.42}$$

with $\boldsymbol{M}_T \in \mathbb{R}^{3\times 2}$ and $\mathbf{b}_T \in \mathbb{R}^3$. Moving from the generalized inverse of the matrix $\boldsymbol{M}_T$, it is possible to define the inverse transformation $F_T^{-1} : T \to \hat{T}$ as:

$$\hat{\mathbf{x}} = (\boldsymbol{M}_T^t\boldsymbol{M}_T)^{-1}\boldsymbol{M}_T^t\mathbf{x} - (\boldsymbol{M}_T^t\boldsymbol{M}_T)^{-1}\boldsymbol{M}_T^t\mathbf{b}_T\,. \tag{3.43}$$

Proof. To get the transformation $F_T^{-1} : T \to \hat{T}$, we follow this chain of equalities:

$$\begin{aligned} \mathbf{x} &= \mathbf{M}_T\hat{\mathbf{x}} + \mathbf{b}_T \\ \mathbf{M}_T^t\mathbf{x} &= \mathbf{M}_T^t\mathbf{M}_T\hat{\mathbf{x}} + \mathbf{M}_T^t\mathbf{b}_T \\ (\mathbf{M}_T^t\mathbf{M}_T)^{-1}\mathbf{M}_T^t\mathbf{x} &= (\mathbf{M}_T^t\mathbf{M}_T)^{-1}\mathbf{M}_T^t\mathbf{M}_T\hat{\mathbf{x}} + (\mathbf{M}_T^t\mathbf{M}_T)^{-1}\mathbf{M}_T^t\mathbf{b}_T \\ (\mathbf{M}_T^t\mathbf{M}_T)^{-1}\mathbf{M}_T^t\mathbf{x} &= \hat{\mathbf{x}} + (\mathbf{M}_T^t\mathbf{M}_T)^{-1}\mathbf{M}_T^t\mathbf{b}_T\,. \end{aligned}$$

We immediately get Equation (3.43).

□

Proposition 3.3.2 *The singular values $s_{i,T} \neq 0$ for $i = 1, 2$ of the matrix $\boldsymbol{M}_T \in \mathbb{R}^{3\times 2}$ and the singular values $w_{i,T} \neq 0$ for $i = 1, 2$ associated with the generalized inverse of $\boldsymbol{M}_T$ satisfy the following relation:*

$$w_{i,T} = \frac{1}{s_{i,T}}\,, \quad \textit{for } i = 1, 2\,. \tag{3.44}$$

Proof. Suppose that $s_{i,T} \neq 0$, for $i = 1, 2$ and $\mathbf{v}_i$ and $\hat{\mathbf{v}}$ are the corresponding singular left and right vectors, respectively. Then, we exploit the following chain of equalities:

$$\begin{aligned} \mathbf{M}_T\hat{\mathbf{v}}_i &= s_i\mathbf{v}_i \\ (\mathbf{M}_T^t\mathbf{M}_T)^{-1}\mathbf{M}_T^t\mathbf{M}_T\hat{\mathbf{v}}_i &= s_{i,T}(\mathbf{M}_T^t\mathbf{M}_T)^{-1}\mathbf{M}_T^t\mathbf{v}_i \\ \hat{\mathbf{v}}_i &= s_{i,T}(\mathbf{M}_T^t\mathbf{M}_T)^{-1}\mathbf{M}_T^t\mathbf{v}_i\,. \end{aligned}$$

We can re-write the last equality as

$$(\mathbf{M}_T^t\mathbf{M}_T)^{-1}\mathbf{M}_T^t\mathbf{v}_i = \frac{1}{s_{i,T}}\hat{\mathbf{v}}_i\,,$$

and this completes the proof.

□

Proposition 3.3.3 *Let $s_{1,T}$ and $s_{2,T}$ be the singular value associated with the matrix $\boldsymbol{M}_T$ such that $s_{1,T} \geq s_{2,T}$ and let $w_{1,T}$ and $w_{2,T}$ be the singular value associated with the generalized inverse of $\boldsymbol{M}_T$ such that $w_{1,T} \geq w_{2,T}$. Then, the following relations hold:*

$$w_{1,T} = \frac{1}{s_{2,T}} \quad \textit{and} \quad w_{2,T} = \frac{1}{s_{1,T}}\,. \tag{3.45}$$

Proof. This statement simply follows form Proposition 3.3.2.

□

Lemma 3.3.1 *Given a triangle $T \in \mathbb{R}^3$ and the reference triangle $\hat{T} \in \mathbb{R}^2$, the following relation holds:*

$$s_{2,T}h_{\hat{T}} \leq h_T \leq s_{1,T}h_{\hat{T}}\,, \tag{3.46}$$

where $s_{1,T}$ and $s_{2,T}$ are the singular values of the matrix $\boldsymbol{M}_T$, defined in Equation (3.42), while $h_{\hat{T}}$ and h_T are the diameters of the triangles $\hat{T}$ and T, respectively.

Proof. Let us consider the map $F_T : \hat{T} \to T$ and its inverse F_T^{-1} defined as in Equation (3.42) and (3.43), respectively. In a triangle, the diameter h_T is an edge of T. Let A and B be the end-points of this edge, and let $\hat{A}$ and $\hat{B}$ be the vertices of $\hat{T}$ satisfying the following relations:

$$\begin{aligned} \mathbf{x}_A &= \mathbf{M}_T \hat{\mathbf{x}}_{\hat{A}} + \mathbf{b}_T \,, \\ \mathbf{x}_B &= \mathbf{M}_T \hat{\mathbf{x}}_{\hat{B}} + \mathbf{b}_T \,. \end{aligned}$$

Now, we make the difference between these two equalities to get:

$$\mathbf{x}_A - \mathbf{x}_B = \mathbf{M}_T (\hat{\mathbf{x}}_{\hat{A}} - \hat{\mathbf{x}}_{\hat{B}}) \,. \tag{3.47}$$

Then, we have

$$\begin{aligned} h_T = |\mathbf{x}_A - \mathbf{x}_B| &= |\mathbf{M}_T (\hat{\mathbf{x}}_{\hat{A}} - \hat{\mathbf{x}}_{\hat{B}})| \\ &\leq ||\mathbf{M}_T|| \, |\hat{\mathbf{x}}_{\hat{A}} - \hat{\mathbf{x}}_{\hat{B}}| \\ &\leq s_{1,T} \, |\hat{\mathbf{x}}_{\hat{A}} - \hat{\mathbf{x}}_{\hat{B}}| \\ &\leq s_{1,T} \, h_{\hat{T}} \,, \end{aligned}$$

here $s_{1,T}$ is the singular value of $\mathbf{M}_T$ associated with the longest norm, $|\cdot|$ is the Euclidean norm and $||\mathbf{M}_T||$ is the matrix norm defined in Definition 3.3.2. In this way we have proved the second inequality in Equation (3.46).

Let us focus on the first one. Consider the transformation F_T^{-1} defined in Equation (3.43). Let $\hat{A}$ and $\hat{B}$ be the end-points of $h_{\hat{T}}$, and A and B be the corresponding vertices of T such that:

$$\begin{aligned} \hat{\mathbf{x}}_A &= (\mathbf{M}_T^t \, \mathbf{M}_T)^{-1} \mathbf{M}_T^t \hat{\mathbf{x}}_A - (\mathbf{M}_T^t \, \mathbf{M}_T)^{-1} \mathbf{M}_T^t \mathbf{b}_T \\ \hat{\mathbf{x}}_B &= (\mathbf{M}_T^t \, \mathbf{M}_T)^{-1} \mathbf{M}_T^t \hat{\mathbf{x}}_B - (\mathbf{M}_T^t \, \mathbf{M}_T)^{-1} \mathbf{M}_T^t \mathbf{b}_T \,. \end{aligned}$$

As before, we make the difference between these two equalities and obtain:

$$\hat{\mathbf{x}}_A - \hat{\mathbf{x}}_B = (\mathbf{M}_T^t \, \mathbf{M}_T)^{-1} \mathbf{M}_T^t (\hat{\mathbf{x}}_A - \hat{\mathbf{x}}_B) \,. \tag{3.48}$$

So we have

$$\begin{aligned} h_{\hat{T}} = |\mathbf{x}_{\hat{A}} - \mathbf{x}_{\hat{B}}| &= |(\mathbf{M}_T^t \, \mathbf{M}_T)^{-1} \mathbf{M}_T^t (\hat{\mathbf{x}}_A - \hat{\mathbf{x}}_B)| \\ &\leq ||(\mathbf{M}_T^t \, \mathbf{M}_T)^{-1} \mathbf{M}_T^t|| \, |\hat{\mathbf{x}}_A - \hat{\mathbf{x}}_B| \\ &\leq w_1 \, |\hat{\mathbf{x}}_A - \hat{\mathbf{x}}_B| \\ &\leq w_1 \, h_T \,, \end{aligned}$$

where $w_{1,T}$ is the singular value of $(\mathbf{M}_T^t \mathbf{M}_T)^{-1}\mathbf{M}_T^t$ associated with the longest norm. But, moving from Proposition 3.3.2 and Proposition 3.3.3, we could state that

$$w_{1,T} = \frac{1}{s_{2,T}}\,,$$

where $s_{2,T}$ is the singular value of $\mathbf{M}_T$ associated with the smallest norm. Then, we have

$$h_{\hat{T}} \leq \frac{1}{s_{2,T}}\, h_T \quad \Rightarrow \quad s_{2,T} h_{\hat{T}} \leq h_T\,.$$

□

Lemma 3.3.2 *We consider a surface Γ and its discrete approximation Γ_h. Let d be the implicit function that defines Γ, $\mathbf{H}$ its Hessian, $\mathbf{P}$ and $\mathbf{P}_h$ the operator defined in Equation (2.12) for Γ and Γ_h, respectively, we define the quantity:*

$$\mathbf{A}_h^E := \frac{1}{\mu_h}\mathbf{P}(\boldsymbol{I} - d\mathbf{H})\mathbf{P}_h(\boldsymbol{I} - d\mathbf{H})\mathbf{P}\,.$$

Then, for each triangle T of Γ_h, the following estimate holds:

$$||\mathbf{A}_h^E - \mathbf{P}||_{L^\infty(T)} < c h_T^2\,, \tag{3.49}$$

where c is a constant and h_T is the diameter of the triangle T.

Proof. This result is provided in [31].

□

Lemma 3.3.3 *Consider a triangle T in Γ_h and let Υ be its image on the surface Γ. Let $v \in H^1(\Upsilon)$ and let $v_h \in H^1(T)$ be the piecewise linear approximation of v on Γ_h. Then the following equivalence relation holds:*

$$\frac{1}{c}\,||\nabla_{\Gamma_h} v_h|| \leq ||\nabla_\Gamma v|| \leq c\,||\nabla_{\Gamma_h} v_h||\,, \tag{3.50}$$

where c is a positive constant.

Proof. This result is provided in [31].

□

Lemma 3.3.4 *For any function $v : \mathbb{R}^3 \to \mathbb{R}$, with $v \in H^1(T)$, for any pair of strictly positive constants α and β and for any pair of orthogonal unitary vectors $\mathbf{r}_1$ and $\mathbf{r}_2$, that lies on the plane identified by T, it holds:*

$$\min(\alpha\,,\beta) \leq \frac{\alpha(\mathbf{r}_1)^t\,\mathbf{G}_T(v)\,\mathbf{r}_1 + \beta(\mathbf{r}_2)^t\,\mathbf{G}_T(v)\,\mathbf{r}_2}{|v|^2_{H^1(T)}} \leq \max(\alpha\,,\beta)\,, \tag{3.51}$$

where $\mathbf{G}_T$ is the matrix defined in Equation (3.7).

Proof. Without loss of generality, let us assume that $\alpha \geq \beta$. We consider the affine map $P_T : \pi_T \to \pi$, defined as:

$$\hat{\mathbf{x}} = \mathbf{N}_T\mathbf{x} + \mathbf{b}_T\,, \quad \forall \mathbf{x} \in \pi_T\,,$$

with $\mathbf{N}_T \in \mathbb{R}^{2\times 3}$, $\mathbf{b}_T \in \mathbb{R}^2$, that transforms the plane π_T identified by the triangle T, into the plane xOy, see Figure 3.13. Moreover, we can build this transformation so that there is no area distortion and the following relation holds:

$$F_T(\mathbf{r}_i) = \mathbf{q}_i\,, \quad \text{for } i = 1, 2\,,$$

where $\mathbf{q}_i - \mathbf{b}_T = \mathbf{e}_i$, $\mathbf{e}_1 = [1, 0]^t$ and $\mathbf{e}_2 = [0, 1]^t$.

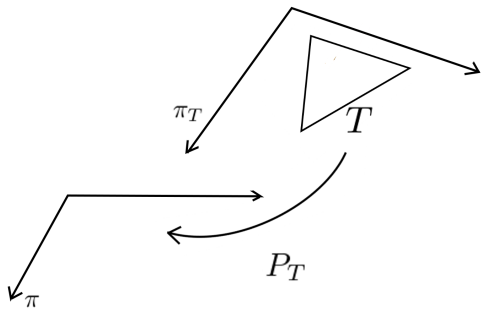


Figure 3.13: Transformation P_T.

Consider the function $\hat{v} : \mathbb{R}^2 \to \mathbb{R}$, with $\hat{v} := v \circ P_T$, so that we have the following relation:

$$\mathbf{G}_T(v) = \mathbf{N}_T^t\mathbf{G}_T(\hat{v})\mathbf{N}_T\,.$$

Moving from this relation we obtain

$$\begin{aligned} & \alpha(\mathbf{r}_1)^t\,\mathbf{G}_T(v)\,\mathbf{r}_1 + \beta(\mathbf{r}_2)^t\,\mathbf{G}_T(v)\,\mathbf{r}_2 = \\ = \;& \alpha(\mathbf{r}_1)^t\mathbf{N}_T^t\mathbf{G}_T(\hat{v})\mathbf{N}_T\,\mathbf{r}_1 + \beta(\mathbf{r}_2)^t\,\mathbf{N}_T^t\mathbf{G}_T(\hat{v})\mathbf{N}_T\,\mathbf{r}_2 \\ = \;& \alpha(\mathbf{N}_T\,\mathbf{r}_1)^t\,\mathbf{G}_T(\hat{v})(\mathbf{N}_T\,\mathbf{r}_1) + \beta(\mathbf{N}_T\,\mathbf{r}_2)^t\,\mathbf{G}_T(\hat{v})(\mathbf{N}_T\,\mathbf{r}_2) \\ = \;& \alpha(\mathbf{q}_1 - \mathbf{b}_T)^t\,\mathbf{G}_T(\hat{v})(\mathbf{q}_1 - \mathbf{b}_T) + \beta(\mathbf{q}_2 - \mathbf{b}_T)^t\,\mathbf{G}_T(\hat{v})(\mathbf{q}_2 - \mathbf{b}_T) \\ = \;& \alpha\mathbf{e}_1^t\,\mathbf{G}_T(\hat{v})\mathbf{e}_1 + \beta\mathbf{e}_2^t\,\mathbf{G}_T(\hat{v})\mathbf{e}_2\,. \end{aligned} \tag{3.52}$$

Since $\alpha \geq \beta > 0$ and $\mathbf{e}_i^t\,\mathbf{G}_T(\hat{v})\mathbf{e}_i \geq 0$ for $i = 1, 2$, we can bound the last term in (3.52) as:

$$\begin{aligned} \beta(\mathbf{e}_1^t\,\mathbf{G}_T(\hat{v})\mathbf{e}_1 + \mathbf{e}_2^t\,\mathbf{G}_T(\hat{v})\mathbf{e}_2) &\leq \alpha\mathbf{e}_1^t\,\mathbf{G}_T(\hat{v})\mathbf{e}_1 + \beta\mathbf{e}_2^t\,\mathbf{G}_T(\hat{v})\mathbf{e}_2 \\ \alpha\mathbf{e}_1^t\,\mathbf{G}_T(\hat{v})\mathbf{e}_1 + \beta\mathbf{e}_2^t\,\mathbf{G}_T(\hat{v})\mathbf{e}_2 &\leq \alpha(\mathbf{e}_1^t\,\mathbf{G}_T(\hat{v})\mathbf{e}_1 + \mathbf{e}_2^t\,\mathbf{G}_T(\hat{v})\mathbf{e}_2)\,. \end{aligned}$$

Moreover, we observe that

$$\begin{aligned} \mathbf{e}_1^t\,\mathbf{G}_T(\hat{v})\mathbf{e}_1 + \mathbf{e}_2^t\,\mathbf{G}_T(\hat{v})\mathbf{e}_2 &= \int_{T'}\left(\frac{\partial\hat{v}}{\partial\hat{x}}\right)^2 d\hat{\sigma} + \int_{T'}\left(\frac{\partial\hat{v}}{\partial\hat{y}}\right)^2 d\hat{\sigma} \\ &= |\hat{v}|^2_{H^1(T')} \\ &= |v|^2_{H^1(T)}\,, \end{aligned} \tag{3.53}$$

where T' is the image of T through the transformation P_T. Since there is no area distortion in the transformation P_T, the determinant of the Jacobian does not appear in the last term of Equation (3.53). Simple algebraic operations lead to Equation (3.51) and this completes the proof.

□

Proposition 3.3.4 *It is always possible to find a transformation $P_T : \pi_T \to \pi$ as demanded in Lemma 3.3.4.*

Proof. We consider two vectors $\mathbf{r}_1, \mathbf{r}_2 \in \mathbb{R}^3$ such that

$$\mathbf{r}_1 \perp \mathbf{r}_2 \quad \text{and} \quad |\mathbf{r}_1| = |\mathbf{r}_2| = 1\,.$$

The transformation P_T has the generic form $\hat{\mathbf{x}} = \mathbf{N}_T\mathbf{x} + \mathbf{b}_T$ and it satisfies the properties

$$\mathbf{q}_1 = \mathbf{N}_T\mathbf{r}_1 + \mathbf{b}_T \;\Rightarrow\; \mathbf{q}_1 - \mathbf{b}_T = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \tag{3.54}$$

$$\mathbf{q}_2 = \mathbf{N}_T\mathbf{r}_2 + \mathbf{b}_T \quad \Rightarrow \quad \mathbf{q}_2 - \mathbf{b}_T = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{3.55}$$

We consider

$$\mathbf{N}_T = \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix}, \quad \mathbf{b}_T = \begin{pmatrix} g \\ h \end{pmatrix},$$

and the vectors $\mathbf{r}_i = (x_i\,, y_i\,, z_i)^t$ for $i = 1, 2$. To verify properties (3.54) and (3.55), P_T has to satisfy the following equations:

$$\begin{aligned} \mathbf{q}_1 &= \begin{pmatrix} ax_1 + by_1 + cz_1 + g \\ dx_1 + ey_1 + fz_1 + h \end{pmatrix} = \begin{pmatrix} 1+g \\ 0+h \end{pmatrix}, \\ \mathbf{q}_2 &= \begin{pmatrix} ax_2 + by_2 + cz_2 + g \\ dx_2 + ey_2 + fz_2 + h \end{pmatrix} = \begin{pmatrix} 0+g \\ 1+h \end{pmatrix}. \end{aligned}$$

We have to solve the following overdetermined linear system, to find the desired unknown coefficients for $\mathbf{N}_T$ and $\mathbf{b}_T$.

$$\begin{aligned} \begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ \begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{pmatrix} \begin{pmatrix} d \\ e \\ f \end{pmatrix} &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{aligned}$$

Since $\mathbf{r}_1 \perp \mathbf{r}_2$, the rows of the matrix are linearly independent, so that these two systems have rank 2 and, for the theorem of Rouché-Capelli, we can always find the solution.

□

Proposition 3.3.5 *Consider a surface Γ and its discretization Γ_h. Let $\psi \in H^1(\Gamma)$, then for each edge e of the mesh Γ_h holds:*

$$||\psi^E - \psi^E_{\mathbf{z}}||_{L^2(e)} \leq c \sum_{T \in \omega_z} \sqrt{|e|} \left(\sum_{i=1}^{2} s^2_{i,T} (\mathbf{r}_{i,T})^t \, \mathbf{G}_T(\psi^E) \, \mathbf{r}_{i,T} \right)^{1/2}, \tag{3.56}$$

where c is a constant, $|e|$ is the length of the edge, ω_z is the patch of elements associated with the node $\mathbf{z}$, $\psi^E : U_\delta \to \mathbb{R}$ is the extension of ψ to U_δ as in Equation (2.9), $\psi^E_{\mathbf{z}}$ is defined in (3.3), $\mathbf{r}_{1,T}$, $\mathbf{r}_{2,T}$ and s_1, s_2 are the singular vectors and singular values of $(\boldsymbol{M}_T)^t$.

Proof. We consider the affine transformation $F_T : \hat{T} \to T$ defined in Section 3.1, in such a way that the generic edge e is mapped into the reference edge $\hat{e}$. We build the function $\eta : \hat{T} \to \mathbb{R}$, $\eta := F_T \circ \psi^E$, so that the left hand side of (3.56) can be written as

$$||\psi^E - \psi^E_{\mathbf{z}}||_{L^2(e)} \leq \sqrt{|e|}||\eta - \psi^E_{\mathbf{z}}||_{L^2(\hat{e})} \, .$$

We apply the standard trace inequality on the reference triangle to obtain:

$$\begin{aligned} \sqrt{|e|}||\eta - \psi^E_{\mathbf{z}}||_{L^2(\hat{e})} &\leq C\sqrt{|e|}(||\eta - \psi^E_{\mathbf{z}}||_{L^2(\hat{T})} + ||\hat{\nabla}\eta||_{L^2(\hat{T})}) \\ &\leq C\sqrt{|e|}\left(\sqrt{\frac{|\hat{T}|}{|T|}}\underbrace{||\psi^E - \psi^E_{\mathbf{z}}||_{L^2(T)}}_{(I)} + \underbrace{||\hat{\nabla}\eta||_{L^2(\hat{T})}}_{(II)}\right) . \end{aligned} \tag{3.57}$$

Now, we estimate separately the quantities (I) and (II) in Equation (3.57). We consider one of the two triangles that share the edge e, say T, to get

$$(I) \leq ||\psi^E - \psi^E_{\mathbf{z}}||_{L^2(\omega_{\mathbf{z}})} \, , \tag{3.58}$$

we can apply estimate (3.3). Now, we consider (II):

$$\begin{aligned} ||\hat{\nabla}\eta||^2_{L^2(\hat{T})} &\leq \frac{|\hat{T}|}{|T|}||(\mathbf{M}_T)^t\nabla_{\Gamma_h}\psi^E||^2_{L^2(T)} \\ &\leq \sum_{T\in\omega_{\mathbf{z}}}\frac{|\hat{T}|}{|T|}||(\mathbf{M}_T)^t\nabla_{\Gamma_h}\psi^E||^2_{L^2(T)} \\ &\leq \sum_{T\in\omega_{\mathbf{z}}}\frac{|\hat{T}|}{|T|}\int_T |(\mathbf{M}_T)^t\nabla_{\Gamma_h}\psi^E|^2 \, d\sigma_h \, . \end{aligned}$$

Following the inequalities in the proof of Proposition 3.2.1, we get

$$\sum_{T\in\omega_{\mathbf{z}}}\frac{|\hat{T}|}{|T|}\int_T |(\mathbf{M}_T)^t\nabla_{\Gamma_h}\psi^E|^2 \, d\sigma_h \leq \sum_{T\in\omega_{\mathbf{z}}}\frac{|\hat{T}|}{|T|}\left(\sum_{i=1}^{2} s^2_{i,T}(\mathbf{r}_{i,T})^t \, \mathbf{G}_T(\psi^E) \, \mathbf{r}_{i,T}\right) .$$

Then, we have

$$(II) \leq \left(\sum_{T\in\omega_{\mathbf{z}}}\frac{|\hat{T}|}{|T|}\left(s^2_{1,T}(\mathbf{r}_{1,T})^t \, \mathbf{G}_T(\psi) \, \mathbf{r}_{1,T} + s^2_{2,T}(\mathbf{r}_{2,T})^t \, \mathbf{G}_T(\psi) \, \mathbf{r}_{2,T}\right)\right)^{1/2}$$

$$\leq \sum_{T\in\omega_z} \sqrt{\frac{|\hat{T}|}{|T|}} \left(s_{1,T}^2 (r_{1,T})^t \, \mathbf{G}_T(\psi)\, r_{1,T} + s_{2,T}^2 (r_{2,T})^t \, \mathbf{G}_T(\psi)\, r_{2,T}\right)^{1/2} .$$

By properly combining the estimates for (I) and (II), we get the desired result (3.56).

□

Proposition 3.3.6 *Let u be the solution of problem (2.15) and let u_h be the solution of the corresponding finite element approximation. Then, the following estimate for the energy norm of the discretization error e_h, defined in Equation (2.41), holds:*

$$|||e_h||| \leq \left(\sum_{T\in\Gamma_h} \rho_T(u_h) (s_{1,T}^2 (\mathbf{r}_{1,T})^t \, \mathbf{G}_T(e_h)\, \mathbf{r}_{1,T} + s_{2,T}^2 (\mathbf{r}_{2,T})^t \, \mathbf{G}_T(e_h)\, \mathbf{r}_{2,T})^{1/2} \right)^{1/2} , \tag{3.59}$$

with $|||e_h||| = \sqrt{\mathcal{A}^{LB}(e_h, e_h)}$ the energy norm, where $\mathcal{A}^{LB}(\cdot\,,\cdot)$ is the bilinear form associated with the weak formulation of Problem (2.15) and

$$\rho_T(u_h) := r_T + r_{T,e} + g_T + d_T \,, \tag{3.60}$$

with

$$r_T := \sum_{z\in T} ||f^E \mu_h + \Delta_{\Gamma_h} u_h||_{L^2(\omega_z)} \,, \qquad r_{T,e} := \sum_{z\in T} \sum_{e\in\omega_z} \sqrt{|e|}\; |||[[\nabla_{\Gamma_h} u_h]]|||_{L^2(e)} \,, \tag{3.61}$$

the internal and the edge residual contributions,

$$g_T := \frac{h_T^2}{\sqrt{s_{2,T}}} ||\nabla_{\Gamma_h} u_h||_{L^2(T)} \,,$$

the geometrical contribution associated with the element T and

$$d_T = \sum_{z\in T} \sqrt{\frac{3}{2}} ||f^E \mu_h - f_h||_{L^2(\omega_z)} \,, \tag{3.62}$$

the data contribution. Moreover, f^E is a proper extension of the function f in (2.15), according to Equation (2.9), f_h is the piecewise linear approximation of f on Γ_h defined via Equation (2.22), μ_h is defined as in Equation (2.34) and where we

have defined the jump of the normal derivative across the generic edge e of the mesh Γ_h as

$$[[\nabla_{\Gamma_h} u_h]] = \begin{cases} 0 & e \in \partial\Gamma_h\,, \\ \nabla_{\Gamma_h} u_h|_{T_1} \cdot \mathbf{w}_1 + \nabla_{\Gamma_h} u_h|_{T_2} \cdot \mathbf{w}_2 & e \notin \partial\Gamma_h\,. \end{cases} \tag{3.63}$$

Here $\mathbf{w}_i$ is an unitary vector, perpendicular to both the edge e and the outward normal to the triangle T_i, see Figure 3.14.

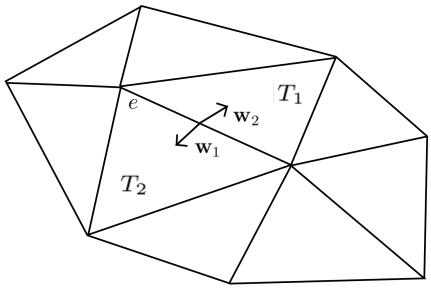


Figure 3.14: Unitary vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ associated with the edge e and with the triangles T_1 and T_2, respectively.

Proof. Let $\psi \in H_0^1(\Gamma)$ and let $\psi_h \in H_0^1(\Gamma_h)$ be its piecewise linear approximation. Following [31], we have:

$$\begin{aligned} &\int_\Gamma \nabla_\Gamma (u - u_h^E) \cdot \nabla_\Gamma \psi \, d\sigma = \\ &= \int_{\Gamma_h} f^E \mu_h \psi^E \, d\sigma_h - \int_\Gamma [\mathbf{P} - \mathbf{A}_h^E] \nabla_\Gamma u_h^E \cdot \nabla_\Gamma \psi \, d\sigma - \int_{\Gamma_h} \nabla_{\Gamma_h} u_h \cdot \nabla_{\Gamma_h} \psi^E \, d\sigma_h\,, \end{aligned} \tag{3.64}$$

where f^E and ψ^E are the extension of the function f and ψ, respectively, to the set U_δ, such that $f^E|_\Gamma = f$ and $\psi^E|_\Gamma = \psi$, see Equation (2.9). In the framework of partial differential equations defined on surfaces embedded in $\mathbb{R}^3$, the well-known Galerkin orthogonality does not hold, so the additional term:

$$-\int_\Gamma [\mathbf{P} - \mathbf{A}_h^E] \nabla_\Gamma u_h^E \cdot \nabla_\Gamma \psi \, d\sigma\,,$$

appears in Equation (3.64). We can write Equation (3.64) also for the piecewise approximation ψ_h of ψ:

$$\begin{aligned}&\int_{\Gamma} \nabla_{\Gamma}(u - u_h^E) \cdot \nabla_{\Gamma} \psi_h^E \, d\sigma = \\ &= \int_{\Gamma_h} f^E \mu_h \psi_h \, d\sigma_h - \int_{\Gamma} [\mathbf{P} - \mathbf{A}_h^E] \nabla_{\Gamma} u_h^E \cdot \nabla_{\Gamma} \psi_h^E \, d\sigma - \int_{\Gamma_h} \nabla_{\Gamma_h} u_h \cdot \nabla_{\Gamma_h} \psi_h \, d\sigma_h \,,\end{aligned} \tag{3.65}$$

where ψ_h^E is the extension to the domain U_δ of the function ψ_h, such that $\psi_h^E|_{\Gamma_h} = \psi_h$, see Equation (2.9). Moving from the weak discrete formulation of Problem 2.3.1, we can re-write the last term of Equation (3.65) in such a way:

$$\int_{\Gamma_h} \nabla_{\Gamma_h} u_h \cdot \nabla_{\Gamma_h} \psi_h \, d\sigma_h = \int_{\Gamma_h} f_h \psi_h \, d\sigma_h \,.$$

Exploiting Equation (3.64) and (3.65), we proceed as follows:

$$\begin{aligned}&\int_{\Gamma} \nabla_{\Gamma}(u - u_h^E) \cdot \nabla_{\Gamma} \psi \, d\sigma = \int_{\Gamma} \nabla_{\Gamma}(u - u_h^E) \cdot \nabla_{\Gamma} \psi \, d\sigma + \\ &+ \int_{\Gamma} \nabla_{\Gamma}(u - u_h^E) \cdot \nabla_{\Gamma} \psi_h^E \, d\sigma - \int_{\Gamma} \nabla_{\Gamma}(u - u_h^E) \cdot \nabla_{\Gamma} \psi_h^E \, d\sigma = \\ = \; &\int_{\Gamma_h} f^E \mu_h \psi^E \, d\sigma_h - \int_{\Gamma} [\mathbf{P} - \mathbf{A}_h^E] \nabla_{\Gamma} u_h^E \cdot \nabla_{\Gamma} \psi \, d\sigma + \\ &- \int_{\Gamma_h} \nabla_{\Gamma_h} u_h \cdot \nabla_{\Gamma_h} \psi^E \, d\sigma_h + + \int_{\Gamma_h} f^E \mu_h \psi_h \, d\sigma_h + \\ &- \int_{\Gamma} [\mathbf{P} - \mathbf{A}_h^E] \nabla_{\Gamma} u_h^E \cdot \nabla_{\Gamma} \psi_h^E \, d\sigma - \int_{\Gamma_h} f_h \psi_h \, d\sigma_h + \\ &- \int_{\Gamma_h} f^E \mu_h \psi_h \, d\sigma_h + \int_{\Gamma} [\mathbf{P} - \mathbf{A}_h^E] \nabla_{\Gamma} u_h^E \cdot \nabla_{\Gamma} \psi_h^E \, d\sigma + \\ &\int_{\Gamma_h} \nabla_{\Gamma_h} u_h \cdot \nabla_{\Gamma_h} \psi_h \, d\sigma_h = \\ = \; &\int_{\Gamma_h} f^E \mu_h (\psi^E - \psi_h) \, d\sigma_h - \int_{\Gamma_h} \nabla_{\Gamma_h} u_h \cdot \nabla_{\Gamma_h} (\psi^E - \psi_h) \, d\sigma_h + \\ &- \int_{\Gamma} [\mathbf{P} - \mathbf{A}_h^E] \nabla_{\Gamma} u_h^E \cdot \nabla_{\Gamma} \psi \, d\sigma + \int_{\Gamma_h} (f^E \mu_h - f_h) \, \psi_h \, d\sigma_h\end{aligned} \tag{3.66}$$

We analyse separately each term at the right-hand side in Equation (3.66)

$$
\begin{aligned}
&-\int_{\Gamma_h} \nabla_{\Gamma_h} u_h \cdot \nabla_{\Gamma_h}(\psi^E - \psi_h)\, d\sigma_h = \\
&= \sum_{T\in\Gamma_h} \int_T \nabla_{\Gamma_h} u_h (\psi^E - \psi_h)\, d\sigma_h - \sum_{e\in T} \int_e \nabla_{\Gamma_h} u_h \cdot \mathbf{w}_e (\psi^E - \psi_h)\, d\sigma_h = \\
&= \int_{\Gamma_h} \Delta_{\Gamma_h} u_h (\psi^E - \psi_h)\, d\sigma_h - \frac{1}{2} \sum_{T\in\Gamma_h} \int_{\partial T} [[\nabla_{\Gamma_h} u_h]] (\psi^E - \psi_h)\, d\sigma_h = \\
&= \int_{\Gamma_h} \Delta_{\Gamma_h} u_h (\psi^E - \psi_h)\, d\sigma_h - \frac{1}{2} \sum_{T\in\Gamma_h} \sum_{e\in\partial T} \int_e [[\nabla_{\Gamma_h} u_h]] (\psi^E - \psi_h)\, d\sigma_h \,,
\end{aligned}
\tag{3.67}
$$

where $\mathbf{w}_e$ is unitary vector perpendicular to both the edge e and the outward pointing normal of the triangle T. Unlike the classical theory provided in [40], since we are dealing with triangular meshes of non planar surfaces, in the jump term

$$
[[\nabla_{\Gamma_h} u_h]] = \begin{cases} 0 & e \in \partial\Gamma_h \,, \\ \nabla_{\Gamma_h} u_h|_{T_1} \cdot \mathbf{w}_1 + \nabla_{\Gamma_h} u_h|_{T_2} \cdot \mathbf{w}_2 & e \notin \partial\Gamma_h \,. \end{cases}
$$

the equality, $\mathbf{w}_2 = -\mathbf{w}_1$, does not generally holds, see Figure 3.14.

Moving from Equation (3.67), we can combine the first two terms in Equation (3.66) as

$$
\begin{aligned}
&\int_{\Gamma_h} f^E \mu_h (\psi^E - \psi_h)\, d\sigma_h - \int_{\Gamma_h} \nabla_{\Gamma_h} u_h \cdot \nabla_{\Gamma_h}(\psi^E - \psi_h)\, d\sigma_h = \\
&= \int_{\Gamma_h} (f^E \mu_h + \Delta_{\Gamma_h} u_h)(\psi^E - \psi_h)\, d\sigma_h - \underbrace{\frac{1}{2} \sum_{T\in\Gamma_h} \sum_{e\in\partial T} \int_e [[\nabla_{\Gamma_h} u_h]] (\psi^E - \psi_h)\, d\sigma_h}_{(i)} \,,
\end{aligned}
$$

since each internal edge of the mesh is taken into account two times in (i), we multiply this term by $1/2$, as in the planar two-dimensional framework. At this level we decompose the right-hand side of Equation (3.66) as the sum of three terms, i.e.,

$$
\int_\Gamma \nabla_\Gamma (u - u_h^E) \cdot \nabla_\Gamma \psi\, d\sigma = R + G + D \,, \tag{3.68}
$$

where

$$
R = \int_{\Gamma_h} (f^E \mu_h + \Delta_{\Gamma_h} u_h)(\psi^E - \psi_h)\, d\sigma_h - \frac{1}{2} \sum_{T\in\Gamma_h} \sum_{e\in\partial T} \int_e [[\nabla_{\Gamma_h} u_h]] (\psi^E - \psi_h)\, d\sigma_h \,,
$$

identifies the *residual part*,

$$G = -\int_{\Gamma} [\mathbf{P} - \mathbf{A}_h^E] \nabla_\Gamma u_h^E \cdot \nabla_\Gamma \psi \, d\sigma \,,$$

provides the *geometric contribution*, due to the absence of the Galerkin orthogonality, and

$$D = \int_{\Gamma_h} (f^E \mu_h - f_h) \psi_h \, d\sigma_h \,,$$

is the *data part*, i.e., the term due to the approximation of the function f via the discrete function f_h. Now, we analyse each of these three terms separately.

Residual Part

Consider the internal residual in R:

$$\begin{aligned}
\int_{\Gamma_h} (f^E \mu_h \;+\; \Delta_{\Gamma_h} u_h)(\psi^E - \psi_h) \, d\sigma_h = \\
&= \sum_{\mathbf{z} \in \Gamma_h} \int_{\omega_\mathbf{z}} (f^E \mu_h + \Delta_{\Gamma_h} u_h) \phi_\mathbf{z} (\psi^E - \psi_\mathbf{z}^E) \, d\sigma_h \\
&\leq \sum_{\mathbf{z} \in \Gamma_h} ||\phi_\mathbf{z} (f^E \mu_h + \Delta_{\Gamma_h} u_h)||_{L^2(\omega_\mathbf{z})} ||\psi^E - \psi_\mathbf{z}^E||_{L^2(\omega_\mathbf{z})} \\
&\leq \sum_{\mathbf{z} \in \Gamma_h} ||f^E \mu_h + \Delta_{\Gamma_h} u_h||_{L^2(\omega_\mathbf{z})} ||\psi^E - \psi_\mathbf{z}^E||_{L^2(\omega_\mathbf{z})} \,,
\end{aligned}$$

where the functions $\phi_\mathbf{z}$ constitute a partition of the unity and $\omega_\mathbf{z}$ represents the corresponding support. Moving from Proposition 3.2.1, we have the following anisotropic a-priori estimate:

$$||\psi^E - \psi_\mathbf{z}^E||_{L^2(\omega_\mathbf{z})} \leq C \left(\sum_{T \in \omega_\mathbf{z}} \left(\sum_{i=1}^{2} s_{i,T}^2 (\mathbf{r}_{i,T})^t \, \mathbf{G}_T(\psi^E) \, \mathbf{r}_{i,T} \right) \right)^{1/2} . \tag{3.69}$$

We proceed in the same way with the boundary residual:

$$\sum_{T \in \Gamma_h} \sum_{e \in \partial T} \int_e [[\nabla_{\Gamma_h} u_h]] (\psi^E - \psi_h) \, d\sigma_h =$$

$$
\begin{aligned}
&= \sum_{\mathbf{z}\in\Gamma_h}\sum_{e\in\omega_\mathbf{z}} \int_e [[\nabla_{\Gamma_h} u_h]]\phi_\mathbf{z}(\psi^E - \psi^E_\mathbf{z})\, d\sigma_h \\
&\leq C \sum_{\mathbf{z}\in\Gamma_h}\sum_{e\in\omega_\mathbf{z}} |||[[\nabla_{\Gamma_h} u_h]]|||_{L^2(e)} ||\psi^E - \psi^E_\mathbf{z}||_{L^2(e)}\,,
\end{aligned}
$$

and, via Proposition 3.3.5, we get the following anisotropic estimate for the global residual contribution:

$$
\begin{aligned}
R \;\leq\; & \sum_{\mathbf{z}\in\Gamma_h} \left[||f^E\mu_h + \Delta_{\Gamma_h} u_h||_{L^2(\omega_\mathbf{z})} \sum_{T\in\omega_\mathbf{z}} \left(\sum_{i=1}^{2} s^2_{i,T} (\mathbf{r}_{i,T})^t\, \mathbf{G}_T(\psi^E)\, \mathbf{r}_{i,T} \right)^{1/2} + \right. \\
& \left. + \sum_{e\in\omega_\mathbf{z}} \sqrt{|e|}\, |||[[\nabla_{\Gamma_h} u_h]]|||_{L^2(e)} \sum_{T\in\omega_\mathbf{z}} \left(\sum_{i=1}^{2} s^2_{i,T} (\mathbf{r}_{i,T})^t\, \mathbf{G}_T(\psi^E)\, \mathbf{r}_{i,T} \right)^{1/2} \right] \\
= \; & \sum_{T\in\Gamma_h} (r_T + r_{T,e}) \left(\sum_{i=1}^{2} s^2_{i,T} (\mathbf{r}_{i,T})^t\, \mathbf{G}_T(\psi^E)\, \mathbf{r}_{i,T} \right)^{1/2},
\end{aligned}
$$

where we have exploited definitions (3.61).

Geometrical Part

We consider the term G:

$$
-\int_\Gamma [\mathbf{P} - \mathbf{A}^E_h] \nabla_\Gamma u^E_h \cdot \nabla_\Gamma \psi\, d\sigma = \int_\Gamma [\mathbf{A}^E_h - \mathbf{P}]\, \nabla_\Gamma u^E_h \cdot \nabla_\Gamma \psi\, d\sigma \tag{3.70}
$$

We decompose the domain Γ in a set of "curved" triangles Υ. Moving from Lemma 3.3.2 and 3.3.3 and from the Cauchy Schwarz and the Holder inequality, we have

$$
\begin{aligned}
&\int_\Gamma [\mathbf{A}^E_h - \mathbf{P}]\, \nabla_\Gamma u^E_h \cdot \nabla_\Gamma \psi\, d\sigma = \\
&\qquad = \sum_{\Upsilon\in\Gamma} \int_\Upsilon [\mathbf{A}^E_h - \mathbf{P}]\, \nabla_\Gamma u^E_h \cdot \nabla_\Gamma \psi\, d\sigma \\
&\qquad \leq c \sum_{T\in\Gamma_h} ||\mathbf{A}^E_{h\,h} - \mathbf{P}||_{L^\infty(T)}\, ||\nabla_{\Gamma_h} u_h||_{L^2(T)} ||\nabla_{\Gamma_h} \psi^E||_{L^2(T)} \\
&\qquad \leq c \sum_{T\in\Gamma_h} h^2_T\, ||\nabla_{\Gamma_h} u_h||_{L^2(T)} ||\nabla_{\Gamma_h} \psi^E||_{L^2(T)}\,,
\end{aligned}
$$

where h_T is the diameter of the triangle T. Lemma 3.3.4 gives the following estimate:

$$||\nabla_{\Gamma_h}\psi^E||_{L^2(T)} \leq \frac{1}{\sqrt{s_{2,T}}}\left(s_{1,T}^2(\mathbf{r}_{1,T})^t\,\mathbf{G}_T(\psi^E)\,\mathbf{r}_{1,T} + s_{2,T}^2(\mathbf{r}_{2,T})^t\,\mathbf{G}_T(\psi^E)\,\mathbf{r}_{2,T}\right)^{1/2}.$$

Then, thanks to Lemma 3.3.1, we get this estimate:

$$\int_\Gamma [\mathbf{A}_h^E - \mathbf{P}]\,\nabla_\Gamma u_h^E \cdot \nabla_\Gamma \psi\, d\sigma \leq$$
$$\leq c\sum_{T\in\Gamma_h} g_T\left(s_{1,T}^2(\mathbf{r}_{1,T})^t\,\mathbf{G}_T(\psi^E)\,\mathbf{r}_{1,T} + s_{2,T}^2(\mathbf{r}_{2,T})^t\,\mathbf{G}_T(\psi^E)\,\mathbf{r}_{2,T}\right)^{1/2}.$$

Data Part

Consider the term D, we have:

$$\begin{aligned}\int_{\Gamma_h}(f^E\mu_h - f_h)\psi_h\, d\sigma_h &= \sum_{\mathbf{z}\in\Gamma_h}\int_{\omega_\mathbf{z}}\phi_\mathbf{z}(f^E\mu_h - f_h)\psi_\mathbf{z}^E\, d\sigma_h\\ &\leq \sum_{\mathbf{z}\in\Gamma_h}||\phi_\mathbf{z}(f^E\mu_h - f_h)||_{L^2(\omega_\mathbf{z})}||\psi_\mathbf{z}^E||_{L^2(\omega_\mathbf{z})}\\ &\leq \sum_{\mathbf{z}\in\Gamma_h}||f^E\mu_h - f_h||_{L^2(\omega_\mathbf{z})}||\psi_\mathbf{z}^E||_{L^2(\omega_\mathbf{z})}\end{aligned}$$

During the proof of Proposition 3.2.1, we have proved the following inequality

$$||\psi_\mathbf{z}^E||_{L^2(\omega_\mathbf{z})} \leq \sqrt{\frac{3}{2}}\,||\psi^E||_{L^2(\omega_\mathbf{z})},$$

then, by exploiting the same computation in the proof of Proposition 3.2.1, we get

$$\begin{aligned}\sum_{\mathbf{z}\in\Gamma_h}||f^E\mu_h - f_h||_{L^2(\omega_\mathbf{z})}||\psi_h^E||_{L^2(\omega_\mathbf{z})} &\leq \sum_{\mathbf{z}\in\Gamma_h}\sqrt{\frac{3}{2}}||f^E\mu_h - f_h||_{L^2(\omega_\mathbf{z})}||\psi^E||_{L^2(\omega_\mathbf{z})}\\ &\leq \sum_{\mathbf{z}\in\Gamma_h}\sqrt{\frac{3}{2}}||f^E\mu_h - f_h||_{L^2(\omega_\mathbf{z})}\left(\sum_{T\in\omega_\mathbf{z}}\left(\sum_{i=1}^{2}s_{i,T}^2(\mathbf{r}_{i,T})^t\,\mathbf{G}_T(\psi^E)\,\mathbf{r}_{i,T}\right)\right)^{1/2}\\ &\leq \sum_{\mathbf{z}\in\Gamma_h}\sqrt{\frac{3}{2}}||f^E\mu_h - f_h||_{L^2(\omega_\mathbf{z})}\sum_{T\in\omega_\mathbf{z}}\left(s_{1,T}^2\left(\sum_{i=1}^{2}s_{i,T}^2(\mathbf{r}_{i,T})^t\,\mathbf{G}_T(\psi^E)\,\mathbf{r}_{i,T}\right)\right)^{1/2}\end{aligned}$$

$$\leq \sum_{T\in\Gamma_h} d_T \left(\sum_{i=1}^{2} s_{i,T}^2 (\mathbf{r}_{i,T})^t \, \mathbf{G}_T(\psi^E) \, \mathbf{r}_{i,T} \right)^{1/2} ,$$

where we have employed the definition in Equation (3.62). Then, if we choose $\psi = e_h$, we obtain the estimate (3.59).

□

Since estimate (3.59) depends on the exact solution u via the error e_h, it is directly not useful to drive an anisotropic adaptation procedure. On the other hand, we expect that the right-hand side of this estimate goes to zero as the mesh becomes finer and finer. To make computable the right-hand side of (3.59), we exploit another standard approach, i.e., we resort to a suitable recovered solution, [114, 115].

But, in this framework we are dealing with a function u_h defined on a surface Γ_h embedded in the three-dimensional space, so we consider the extension of the gradient recovery procedure to an arbitrary surface proposed by H. Wei et al. in [110], see Subsection 2.6.2.

Let u_h be the discrete solution to Problem 2.3.2 and let $\nabla_{\Gamma_h} u_h$ be the tangential gradient on the discrete surface. We denote the tangential recovered gradient by

$$\nabla_{\Gamma_h}^{WCY} u_h = \left[(\nabla_{\Gamma_h}^{WCY} u_h)_i \right] , \quad \text{for } i = 1, 2, 3 ,$$

so that the components of the matrix $\mathbf{G}_T(e_h)$ are:

$$\begin{aligned} [\mathbf{G}_T(e_h^*)]_{i,j} &= \int_T \frac{\partial e_h^*}{\partial x_i} \frac{\partial e_h^*}{\partial x_j} \\ &= \int_T \Big((\nabla_{\Gamma_h}^{WCY} u_h)_i - (\nabla_{\Gamma_h} u_h)_i \Big) \Big((\nabla_{\Gamma_h}^{WCY} u_h)_j - (\nabla_{\Gamma_h} u_h)_j \Big) , \end{aligned} \tag{3.71}$$

with $i, j = 1, 2, 3$, and with $x_1 = x$, $x_2 = y$, $x_3 = z$.

3.3.1 From the Estimator to an Anisotropic Metric

As in the a-priori adaptation procedure, we need to define a suitable metric. Before starting the adaptation process, we fix a tolerance on the error and then we equidistribute the error by balancing the contribution given by each element. Thus,

given a tolerance τ, we demand that the following relation holds for each triangle T of the mesh:

$$\rho_T(u_h)(s_{1,T}^2(\mathbf{r}_{1,T})^t\,\mathbf{G}_T(e_h^*)\,\mathbf{r}_{1,T} + s_{2,T}^2(\mathbf{r}_{2,T})^t\,\mathbf{G}_T(e_h^*)\,\mathbf{r}_{2,T})^{1/2} = \frac{\tau^2}{\#\mathcal{T}}\,, \quad (3.72)$$

where $\#\mathcal{T}$ is the number of triangles in the initial mesh Γ_h. We recall that $2|T| = s_{1,T}s_{2,T}$ and we apply a suitable factor that leads us to consider the "adimensionalized" quantities

$$\overline{\mathbf{G}}_T(e_h^*) = \mathbf{G}_T(e_h^*)/|T| \quad \text{and} \quad \overline{\rho}_T(u_h) = \rho_T(u_h)/\sqrt{|T|}\,.$$

Then, Equation (3.72) becomes:

$$\sqrt{2}|T|^{3/2}\overline{\rho}_T(u_h)\left(\underbrace{s_T(\mathbf{r}_{1_T})^t\,\overline{\mathbf{G}}_T(e_h^*)\,\mathbf{r}_{1,T} + \frac{1}{s_T}(\mathbf{r}_{2,T})^t\,\overline{\mathbf{G}}_T(e_h^*)\,\mathbf{r}_{2,T}}_{(I)}\right)^{1/2} = \frac{\tau^2}{\#\mathcal{T}}\,, \quad (3.73)$$

where s_T is the stretching factor defined in Equation (3.15).

We observe that the quantity (I) depends on:

- the stretching factor s_T;
- the vectors $\mathbf{r}_{1,T}$ and $\mathbf{r}_{2,T}$ that lie in the plane identified by the triangle T.

Since $\mathbf{r}_{1,T}$ and $\mathbf{r}_{2,T}$ are orthonormal, we introduce the quantity:

$$\nu_T(s_T, \mathbf{r}_{1,T}) := \left(s_T(\mathbf{r}_{1_T})^t\,\overline{\mathbf{G}}_T(e_h)\,\mathbf{r}_{1,T} + \frac{1}{s_T}(\mathbf{r}_{2,T})^t\,\overline{\mathbf{G}}_T(e_h)\,\mathbf{r}_{2,T}\right)^{1/2},$$

Then, Equation (3.73) can be re-written in a more compact form as:

$$|T|^{3/2}\overline{\rho}_T(u_h)\nu_T(s_T, \mathbf{r}_{1,T}) = \frac{\tau^2}{\#\mathcal{T}}\,. \quad (3.74)$$

Equation (3.74) is not enough to uniquely determine s_T, $\mathbf{r}_{1,T}$ and $|T|$. We proceed as in the a-priori case and we add another requirement: we demand that the condition (3.74) is satisfied with the most economical adapted mesh, i.e., we aim at minimizing the number of elements. This corresponds to the request that condition (3.74) has to be verified with the maximum possible value of the area

of T. So, Equation (3.74) has to be satisfied under the constraint that ν_T is minimum. Following [40], we compute the solution to this constrained minimization problem:

$$\mathbf{r}_{1,T} = \mathbf{w}_2\,, \quad \mathbf{r}_{2,T} = \mathbf{w}_1\,, \quad s_T = \sqrt{\frac{\mu_1}{\mu_2}}$$

where $\mathbf{w}_1$, $\mathbf{w}_2$ and μ_1, μ_2 are the eigenvectors and eigenvalues of the matrix $\overline{\mathbf{G}}_T(e_h^*)$, with $\mu_1 > \mu_2$. At this point, we are able to fix the quantities σ_1, σ_2, $\mathbf{u}_1$, $\mathbf{u}_2$ and $\mathbf{n}$ of Equation (1.14) for each triangle $T \in \mathcal{T}$, i.e., we may define a piecewise constant metric as

$$\sigma_1^2 = q s_T\,, \quad \sigma_2^2 = \frac{q}{s_T}\,, \quad \mathbf{u}_1 = \mathbf{w}_2\,, \quad \mathbf{u}_2 = \mathbf{w}_1\,,$$

with

$$q = \sqrt[3]{\frac{4\tau^4}{\overline{\rho}_T(u_h)\nu_{\text{min}}}} \quad \text{and} \quad \nu_{\text{min}} := \sqrt{\frac{\mu_1}{\mu_2}}\mu_2 + \sqrt{\frac{\mu_2}{\mu_1}}\mu_1\,.$$

3.3.2 Numerical Results

The aim of this section is to numerically verify that the metric derived from the anisotropic a-posteriori error estimate, proposed in Proposition 3.3.6, generates an anisotropic mesh whose triangles are aligned according to the directions of the solution to a Laplace-Beltrami problem.

In particular, for each example, we focus on the following quantities:

a) *number of the elements*: we look for meshes with a reduced number of elements to contain the computational costs;

b) *stretching factor*: to evaluate the distortion of the mesh elements. The stretching factor s_T is defined in Equation (3.15). We recall that values of s_T close to 1 mean that the element T is similar to an equilateral triangle, while high values of s_T refer to very stretched elements. In particular, we compute the maximum value for the stretching factor, i.e.,

$$s_{\text{max}} := \max_{T\in\mathcal{T}} s_T\,,$$

where $\mathcal{T}$ is the set of the triangles in Γ_h.

c) *evaluation of the error*: to evaluate the error, we build a reference solution u_{ref} on a really fine surface mesh and we compare it with the discrete solution u_h obtained by the adaptation process. Then, we compute:

$$\begin{aligned} e_{\text{tot}} &= \sum_{T\in\mathcal{T}} ||u^E_{\text{ref}} - u_h||_{L^2(T)}\,, \\ e_{\text{max}} &= \max_{T\in\mathcal{T}} ||u^E_{\text{ref}} - u_h||_{L^2(T)}\,, \\ e_{\text{min}} &= \min_{T\in\mathcal{T}} ||u^E_{\text{ref}} - u_h||_{L^2(T)}\,, \end{aligned} \tag{3.75}$$

where $\mathcal{T}$ is the set of the triangles in Γ_h and u^E_{ref} is the extension of the reference solution to U_δ, see Equation (2.9).

d) *isotropic case*: we evaluate all the quantities defined in Equation (3.76) in the case of an isotropic mesh adaptation to compare the performance of the adaptive processes. In particular, we compare meshes with about the same number of elements or about the same value of the total error, e_{tot}.

The tolerances, τ, driving the adaptation procedure are specified in the tables below.

Example 3

We consider the following problem:

$$\begin{cases} -\Delta_\Gamma u &= -\frac{1}{(x-1.01)^2} & \text{on } \Gamma_3, \\ u &= 0 & \text{on } \partial\Gamma_3. \end{cases} \tag{3.76}$$

The surface Γ_3 is defined by the zero level set of the function

$$d_3 \,:\, [0,1] \,\times\, [0,1] \,\times\, [-0.05, 0.05] \to\, \mathbb{R}\,,$$

defined as:

$$d_3(x,\, y,\, z) := 0.05 \sin(4\pi x) \sin(4\pi y) - z\,. \tag{3.77}$$

The solution u_{ref} computed on a very fine mesh shows a boundary layer along the axes as highlighted in Figure 3.15.

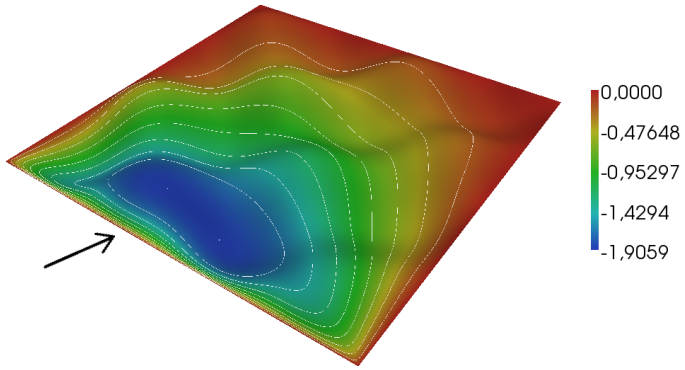


Figure 3.15: The reference solution u_{ref} of example 3 computed on a really fine mesh, the iso-lines highlight the presence of the boundary layer.

In Figure 3.16 the isotropic and the anisotropic adapted meshes are compared together with a corresponding detail shown in Figure 3.17. We may appreciate that the triangles of the anisotropic mesh are properly aligned, in order to capture the trend of the solution.

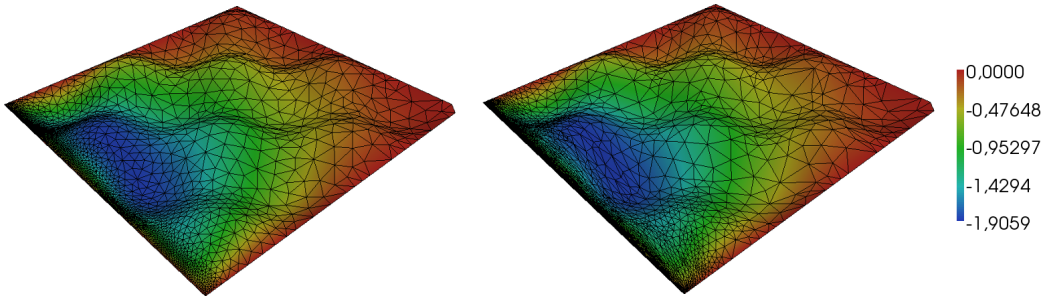


Figure 3.16: Isotropic adapted mesh on the left, the anisotropic adapted mesh on the right.

In Table 3.9, we provide a more quantitative analysis for the resulting meshes. In particular, we observe that the mesh yielded by the proposed adaptive procedure are really anisotropic, as highlighted by the associated high values of $s_{\max}$. From the data in Table 3.9, we may observe that the isotropic adapted meshes have a value of $s_{\max}$, but it remains too limited to consider the mesh anisotropic.

Moreover, we numerically check the typical behaviour of the anisotropic mesh

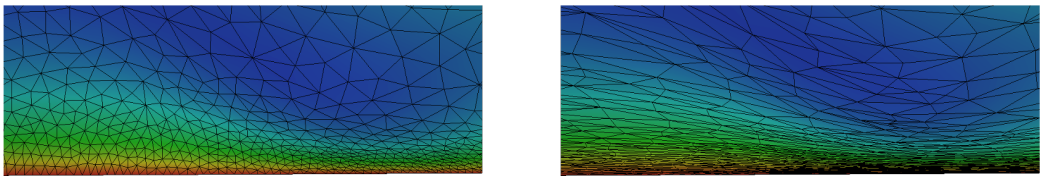


Figure 3.17: Detail of the isotropic adapted mesh, on the left, and of the anisotropic adapted mesh, on the right.

adaptation: if we fix about the same number of elements, we get a higher accuracy on the anisotropic mesh, i.e., a lower error. Vice-versa, the anisotropic mesh provides the same error as in the isotropic case but with a reduced number of elements.

	iso	ani
Ele.	4787	4686
e_{tot}	4.644e-01	1.999e-01
e_{max}	1.571e-02	4.952e-03
e_{min}	1.774e-07	5.812e-08
s_{max}	2.495e+00	8.433e+01
τ	2.000e-01	2.000e-01

	iso	ani
Ele.	2021	1144
e_{tot}	1.198e+00	9.958e-01
e_{max}	4.847e-02	4.929e-02
e_{min}	1.035e-07	2.147e-07
s_{max}	2.593e+00	5.569e+01
τ	0.100e-01	0.600e-01

Table 3.9: Comparison between the isotropic and the anisotropic mesh adaptation procedure for about the same number of elements, on the left, and for about the same accuracy of the total error, on the right.

Example 4

We consider the following problem:

$$\begin{cases} -\Delta_\Gamma u &= -\frac{1}{(x-1.01)^2} + \frac{1}{(y-1.01)^2} & \text{on } \Gamma_4, \\ u &= 0 & \text{on } \partial\Gamma_4. \end{cases} \tag{3.78}$$

The surface Γ_4 is defined by the zero level set of the function

$$d_4 \,:\, [0,1] \,\times\, [0,1] \,\times\, [0,0.6] \to\, \mathbb{R}\,,$$

defined as:

$$d_4(x,\, y,\, z) := 0.6 \sin(\pi x) \sin(\pi y) - z\,. \tag{3.79}$$

The solution u_{ref} is computed on a very fine mesh and it exhibits two boundary layers along the axes as highlighted by the arrows in Figure 3.18.

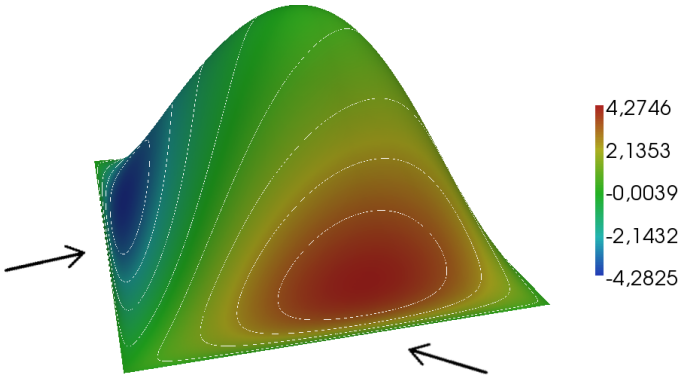


Figure 3.18: The reference solution u_{ref} of example 4 computed on a really fine mesh, the iso-lines highlight the presence of the boundary layer.

As in the previous example, we compare an isotropic with an anisotropic adapted mesh, see Figure 3.19. From the details in Figure 3.20 and 3.21, we recognize that the triangles are aligned according to the directions of the solution.

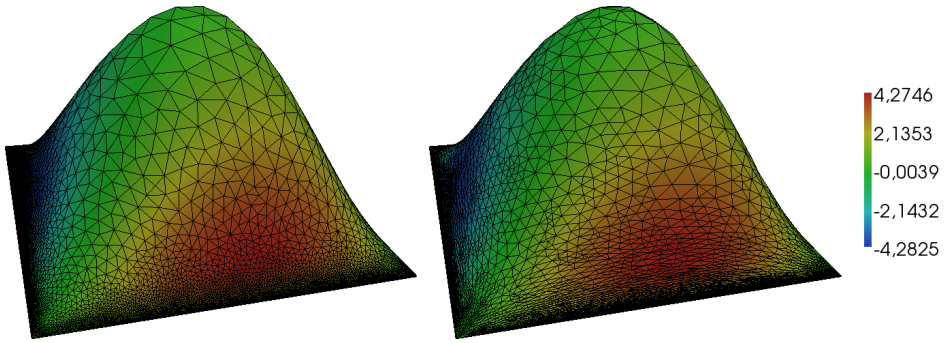


Figure 3.19: Isotropic adapted mesh on the left, the anisotropic adapted mesh on the right.

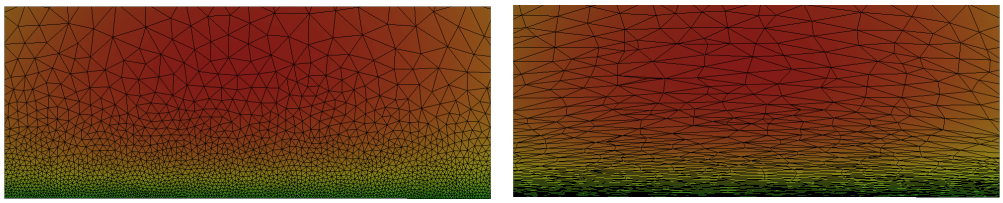


Figure 3.20: Detail of the isotropic adapted mesh, on the left, and of the anisotropic adapted mesh, on the right.

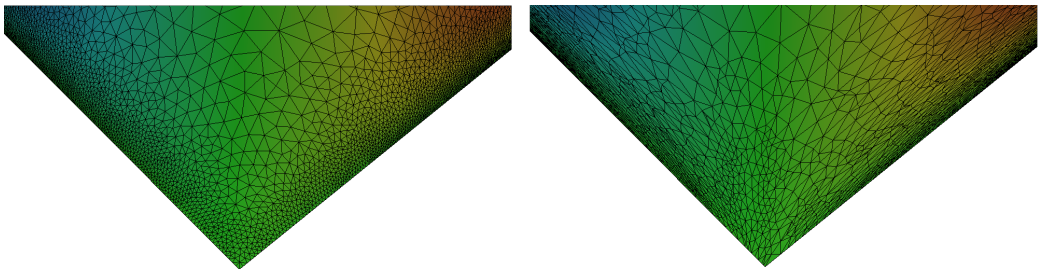


Figure 3.21: On the left a detail of the isotropic adapted mesh, on the right the same detail on the anisotropic adapted mesh.

Finally, in Table 3.10, we collect some quantitative results and, even in this case, we numerically assert the expected better behaviour of the anisotropic adapted meshes.

	iso	ani
Ele.	1249	1117
e_{tot}	4.770e+00	2.138e+00
e_{max}	7.041e-02	3.258e-02
e_{min}	4.993e-07	6.534e-08
s_{max}	2.999e+00	6.782e+01
τ	2.000e-01	8.000e-01

	iso	ani
Ele.	1249	522
e_{tot}	4.770e+00	4.485e+00
e_{max}	7.041e-02	7.606e-02
e_{min}	4.993e-07	3.090e-07
s_{max}	2.999e+00	4.387e+01
τ	8.000e-01	8.000e-01

Table 3.10: Comparison between the isotropic and the anisotropic mesh adaptation procedure for about the same number of elements, on the left, and for about the same accuracy of the total error, on the right.

3.4 Anisotropic A-Posteriori Error Estimator for the Energy Norm: Convection-Diffusion Problem

Let us focus on Problem 2.5.2. We first prove an analogous of Equation (3.64) in the case of problem (2.51).

Proposition 3.4.1 *Let u and u_h be the solutions of Problem 2.5.1 and 2.5.2, respectively. Then, the following relation holds:*

$$\begin{aligned}
&\int_\Gamma \left[\nabla_\Gamma(u-u_h^E)\cdot\nabla_\Gamma\phi_h^E + \mathbf{v}^E\cdot\nabla_\Gamma(u-u_h^E)\phi_h^E\right]\,d\sigma = \\
=\ &\int_{\Gamma_h} f^E\phi_h^E\mu_h\,d\sigma_h + \\
&+\int_\Gamma \left[(\mathbf{A}_h^E-\mathbf{I})\nabla_\Gamma u_h^E\cdot\nabla_\Gamma\phi_h^E + \mathbf{v}^E\cdot(\mathbf{B}_h^E-\mathbf{I})\nabla_\Gamma u_h^E\phi_h^E\right]\,d\sigma + \\
&-\int_{\Gamma_h}\left[\nabla_{\Gamma_h}u_h\cdot\nabla_{\Gamma_h}\phi_h + \mathbf{v}^E\cdot\nabla_{\Gamma_h}u_h\phi_h\right]\,d\sigma_h\,,
\end{aligned} \tag{3.80}$$

with $\phi_h \in H_0^1(\Gamma)$ and where ϕ_h^E is a proper extension of ϕ_h to U_δ such that $\phi_h^E|_{\Gamma_h} = \phi_h$, while $\mathbf{v}^E$ and f^E extend the vector field $\mathbf{v}$ and the function f to U_δ, respectively.

Proof. Let us consider the functions u and u_h that verify Equation (2.49) and (2.52), respectively. Moving from Equation (2.8), we extend both the functions u_h and ϕ_h, i.e., we define $u_h^E : U_\delta \to \mathbb{R}$ such that $u_h^E|_{\Gamma_h} = u_h$ and $\phi_h^E : U_\delta \to \mathbb{R}$ such that $\phi_h^E|_{\Gamma_h} = \phi_h$. Via these transformations, we get

$$\begin{aligned}
&\int_\Gamma \mathbf{P}_h(\mathbf{I}-d\mathbf{H})\nabla_\Gamma u_h^E\cdot\mathbf{P}_h(\mathbf{I}-d\mathbf{H})\nabla_\Gamma\phi_h^E\frac{1}{\mu_h}d\sigma + \\
&+\int_\Gamma \mathbf{v}^E\cdot\mathbf{P}_h(\mathbf{I}-d\mathbf{H})\nabla_\Gamma u_h^E\phi_h^E\frac{1}{\mu_h}d\sigma = \\
&=\int_\Gamma f_h\frac{1}{\mu_h}\phi_h^E d\sigma\,,
\end{aligned} \tag{3.81}$$

where $\mu_h = d\sigma/d\sigma_h$, $\mathbf{H}$ is the Hessian of the signed distance function d that defines the surface Γ and $\mathbf{P}_h$ is the operator on the surface Γ_h, see Equation (2.7). Now, we define the quantities:

$$\mathbf{A}_h^E = \frac{1}{\mu_h}\mathbf{P}(\mathbf{I}-d\mathbf{H})\mathbf{P}_h(\mathbf{I}-d\mathbf{H})\mathbf{P}\,,$$

$$\mathbf{B}_h^E = \frac{1}{\mu_h}\mathbf{P}_h(\mathbf{I} - d\mathbf{H})\mathbf{P}\,,$$
$$F_h^E = \frac{1}{\mu_h} f_h^E\,.$$

We observe that the matrices $\mathbf{P}$, $\mathbf{P}_h$ and $\mathbf{H}$ are symmetric. Then, via Proposition 2.4.1, we can re-write Equation (3.81) as

$$\int_\Gamma \left[\mathbf{A}_h^E \nabla_\Gamma u^E \cdot \nabla_\Gamma \phi_h^E + \mathbf{v}^E \cdot \mathbf{B}_h^E \nabla_\Gamma u_h^E \phi_h^E\right] d\sigma = \int_\Gamma F_h^E \phi_h^E \, d\sigma\,. \tag{3.82}$$

Then, we have

$$\begin{aligned}
\int_\Gamma \nabla_\Gamma u_h^E \cdot \nabla_\Gamma \phi_h^E + \mathbf{v}^E \cdot \nabla_\Gamma u_h^E \phi_h^E d\sigma &= \int_\Gamma F_h^E \phi_h^E \, d\sigma + \\
&+ \int_\Gamma \nabla_\Gamma u_h^E \cdot \nabla_\Gamma \phi_h^E + \mathbf{v}^E \cdot \nabla_\Gamma u_h^E \phi_h^E d\sigma + \\
&- \int_\Gamma \left[\mathbf{A}_h^E \nabla_\Gamma u_h^E \cdot \nabla_\Gamma \phi_h^E + \mathbf{v}^E \cdot \mathbf{B}_h^E \nabla_\Gamma u_h^E \phi_h^E\right] d\sigma = \\
= \int_\Gamma F_h^E \phi_h^E \, d\sigma &- \int_\Gamma \left[(\mathbf{A}_h^E - \mathbf{I})\nabla_\Gamma u_h^E \cdot \nabla_\Gamma \phi_h^E + \mathbf{v}^E \cdot (\mathbf{B}_h^E - \mathbf{I})\nabla_\Gamma u_h^E \phi^E\right] d\sigma\,.
\end{aligned} \tag{3.83}$$

Moreover, we have

$$\int_\Gamma F_h^E \phi_h^E \, d\sigma = \int_{\Gamma_h} f_h \phi_h \, d\sigma_h = \int_{\Gamma_h} \left[\nabla_{\Gamma_h} u_h \cdot \nabla_{\Gamma_h} \phi_h + \mathbf{v}^E \cdot \nabla_{\Gamma_h} u_h \phi_h\right] d\sigma_h\,. \tag{3.84}$$

Then, if we subtract (3.83) properly combined with (3.84) from Equation (2.49), we complete the proof.

□

We are now ready to derive the desired a-posteriori error estimator.

Proposition 3.4.2 *Let u be the solution to Problem (2.49) and let u_h be the corresponding finite element approximation, solution to Equation (2.52). Then, the following estimate holds:*

$$|||e_h||| \leq \left(\sum_{T\in\Gamma_h} \rho_T(u_h)(s_{1_T}^2(\mathbf{r}_{1_T})^t\, \mathbf{G}_T(e_h)\, \mathbf{r}_{1,T} + s_{2,T}^2(\mathbf{r}_{2,T})^t\, \mathbf{G}_T(e_h)\, \mathbf{r}_{2,T})^{1/2}\right)^{1/2}, \tag{3.85}$$

with $|||e_h||| = \sqrt{\mathcal{A}^{CV}(e_h, e_h)}$*, the energy norm, where* $\mathcal{A}^{CV}(\cdot\,,\cdot)$ *is the bilinear form associated with the weak formulation of problem* (2.48)*, and where*

$$\rho_T(u_h) := r_T + r_{T,e} + g_T + d_T\,, \tag{3.86}$$

with

$$\begin{aligned} r_T &:= \sum_{z\in T} ||f^E\mu_h + \Delta_{\Gamma_h} u_h - \mathbf{v}^E\cdot\nabla_{\Gamma_h} u_h||_{L^2(\omega_z)}\,, \\ r_{T,e} &:= \sum_{z\in T}\sum_{e\in\omega_z} \sqrt{|e|}\;|||[[\nabla_{\Gamma_h} u_h]]|||_{L^2(e)}\,, \end{aligned} \tag{3.87}$$

are the internal and boundary residuals associated with the element T*,*

$$g_T := \frac{h_T^2}{\sqrt{s_{2,T}}}\,||\nabla_{\Gamma_h} u_h||_{L^2(T)} + \sum_{z\in T} h_{\omega_z}\,||\mathbf{v}^E\cdot\nabla_{\Gamma_h} u_h||_{L^2(\omega_z)}\,,$$

is the geometrical contribution with

$$h_{\omega_z} := \max_{T\in\omega_z} h_T\,,$$

and

$$d_T = \sum_{z\in T}\sqrt{\frac{3}{2}}||f^E\mu_h - f_h||_{L^2(\omega_z)}\,,$$

is the data contribution.

Proof. Let $\psi\in H_0^1(\Gamma)$ and let $\psi_h\in H_0^1(\Gamma_h)$ be the associated linear approximation. Moving from Equation (3.80), we have

$$\begin{aligned} \int_\Gamma \left[\nabla_\Gamma(u-u_h^E)\cdot\nabla_\Gamma\psi + \mathbf{v}^E\cdot\nabla_\Gamma(u-u_h^E)\psi\right]\,d\sigma &= \int_{\Gamma_h} f^E\psi_h^E\mu_h\,d\sigma_h + \\ + \int_\Gamma \left[(\mathbf{A}_h^E-\mathbf{P})\nabla_\Gamma u_h^E\cdot\nabla_\Gamma\psi + \mathbf{v}^E\cdot(\mathbf{B}_h^E-\mathbf{P})\nabla_\Gamma u_h^E\psi\right]\,d\sigma + \\ - \int_{\Gamma_h}\left[\nabla_{\Gamma_h}u_h\cdot\nabla_{\Gamma_h}\psi^E + \mathbf{v}^E\cdot\nabla_{\Gamma_h}u_h\psi^E\right]\,d\sigma_h\,, \end{aligned} \tag{3.88}$$

where f^E and ψ^E are the extension of the functions f and ψ, respectively, to the set U_δ such that $f^E|_\Gamma = f$ and $\psi^E|_\Gamma = \psi$. Even in this case, the Galerkin orthogonality does not hold so that we have the additional term:

$$\int_\Gamma \left[(\mathbf{A}_h^E-\mathbf{P})\nabla_\Gamma u_h^E\cdot\nabla_\Gamma\psi + \mathbf{v}^E\cdot(\mathbf{B}_h^E-\mathbf{P})\nabla_\Gamma u_h^E\psi\right]\,d\sigma\,.$$

We can write Equation (3.88) for ψ_h:

$$\begin{aligned}\int_\Gamma \left[\nabla_\Gamma(u-u_h^E)\cdot\nabla_\Gamma\psi_h^E+\mathbf{v}^E\cdot\nabla_\Gamma(u-u_h^E)\psi_h^E\right]\,d\sigma=\int_{\Gamma_h} f^E\psi_h\mu_h\,d\sigma+\\+\int_\Gamma\left[(\mathbf{A}_h^E-\mathbf{P})\nabla_\Gamma u_h^E\cdot\nabla_\Gamma\psi_h^E+\mathbf{v}^E\cdot(\mathbf{B}_h^E-\mathbf{P})\nabla_\Gamma u_h^E\psi_h^E\right]\,d\sigma+\\-\int_{\Gamma_h}\left[\nabla_{\Gamma_h}u_h\cdot\nabla_{\Gamma_h}\psi_h+\mathbf{v}^E\cdot\nabla_{\Gamma_h}u_h\psi_h\right]\,d\sigma_h\,. \qquad (3.89)\end{aligned}$$

where ψ_h^E is the extension to the domain U_δ of the function ψ_h such that $\psi_h^E|_{\Gamma_h}=\psi_h$. Moving from Equation (2.52), we can re-write the last term in Equation (3.89) as

$$\int_{\Gamma_h}\left[\nabla_{\Gamma_h}u_h\cdot\nabla_{\Gamma_h}\psi_h+\mathbf{v}^E\cdot\nabla_{\Gamma_h}u_h\psi_h\right]\,d\sigma_h=\int_{\Gamma_h}f_h\psi_h\,d\sigma_h\,.$$

Exploiting Equations (3.88) and (3.89) and by mimicking the procedure followed to get Equation (3.66), we have

$$\begin{aligned}&\int_\Gamma\left[\nabla_\Gamma(u-u_h^E)\cdot\nabla_\Gamma\psi+\mathbf{v}^E\cdot\nabla_\Gamma(u-u_h^E)\psi\right]\,d\sigma=\\
&=\int_\Gamma\left[\nabla_\Gamma(u-u_h^E)\cdot\nabla_\Gamma\psi+\mathbf{v}^E\cdot\nabla_\Gamma(u-u_h^E)\psi\right]\,d\sigma+\\
&\quad+\int_\Gamma\left[\nabla_\Gamma(u-u_h^E)\cdot\nabla_\Gamma\psi_h^E+\mathbf{v}^E\cdot\nabla_\Gamma(u-u_h^E)\psi_h^E\right]\,d\sigma+\\
&\quad-\int_\Gamma\left[\nabla_\Gamma(u-u_h^E)\cdot\nabla_\Gamma\psi_h^E+\mathbf{v}^E\cdot\nabla_\Gamma(u-u_h^E)\psi_h^E\right]\,d\sigma=\\
&=\int_{\Gamma_h}f^E\mu_h\psi^E\,d\sigma_h+\int_\Gamma\left[(\mathbf{A}_h^E-\mathbf{P})\nabla_\Gamma u_h^E\cdot\nabla_\Gamma\psi+\mathbf{v}^E\cdot(\mathbf{B}_h^E-\mathbf{P})\nabla_\Gamma u_h^E\psi\right]\,d\sigma+\\
&\quad-\int_{\Gamma_h}\left[\nabla_{\Gamma_h}u_h\cdot\nabla_{\Gamma_h}\psi+\mathbf{v}^E\cdot\nabla_{\Gamma_h}u_h\psi\right]\,d\sigma_h-\int_{\Gamma_h}f_h\psi_h\,d\sigma_h+\\
&\quad-\int_{\Gamma_h}f^E\mu_h\psi_h^E\,d\sigma_h+\int_{\Gamma_h}\left[\nabla_{\Gamma_h}u_h\cdot\nabla_{\Gamma_h}\psi_h+\mathbf{v}^E\cdot\nabla_{\Gamma_h}u_h\psi_h\right]\,d\sigma_h\\
&=\int_{\Gamma_h}f^E\mu_h(\psi^E-\psi_h)\,d\sigma_h+\int_{\Gamma_h}(f^E\mu_h-f_h)\psi_h\,d\sigma_h+\\
&\quad+\int_\Gamma\left[(\mathbf{A}_h^E-\mathbf{P})\nabla_\Gamma u_h^E\cdot\nabla_\Gamma\psi+\mathbf{v}^E\cdot[\mathbf{B}_h^E-\mathbf{P}]\nabla_\Gamma u_h^E\psi\right]\,d\sigma+\\
&\quad-\int_{\Gamma_h}\left[\nabla_{\Gamma_h}u_h\cdot\nabla_{\Gamma_h}(\psi^E-\psi_h)\,d\sigma_h+\mathbf{v}^E\cdot\nabla_{\Gamma_h}u_h(\psi^E-\psi_h)\right]\,d\sigma_h\,. \qquad (3.90)\end{aligned}$$

Integrating by parts the last term at the right-hand side of Equation (3.90), we have

$$
\begin{aligned}
&-\int_{\Gamma_h} \nabla_{\Gamma_h} u_h \cdot \nabla_{\Gamma_h}(\psi^E - \psi_h)\, d\sigma_h = \\
&= \int_{\Gamma_h} \Delta_{\Gamma_h} u_h (\psi^E - \psi_h)\, d\sigma_h - \frac{1}{2} \sum_{T\in\Gamma_h} \sum_{e\in\partial T} \int_e [[\nabla_{\Gamma_h} u_h]](\psi^E - \psi_h)\, d\sigma_h .
\end{aligned}
$$

where $[[\nabla_{\Gamma_h} u_h]]$ is the jump of the normal derivative across the generic edge e of the mesh defined as in Equation (3.63). Thanks to this equation, we can combine the first and the last term of the right-hand side of of Equation (3.90) and we obtain

$$
\begin{aligned}
&\int_{\Gamma_h} f^E \mu_h (\psi^E - \psi_h)\, d\sigma_h + \\
&-\int_{\Gamma_h} \left[\nabla_{\Gamma_h} u_h \cdot \nabla_{\Gamma_h}(\psi^E - \psi_h) + \mathbf{v}^E \cdot \nabla_{\Gamma_h} u_h (\psi^E - \psi_h)\right]\, d\sigma_h = \\
&= \int_{\Gamma_h} (f^E \mu_h + \Delta_{\Gamma_h} u_h - \mathbf{v}^E \cdot \nabla_{\Gamma_h} u_h)(\psi^E - \psi_h)\, d\sigma_h + \\
&\quad -\frac{1}{2} \sum_{T\in\Gamma_h} \sum_{e\in\partial T} \int_e [[\nabla_{\Gamma_h} u_h]](\psi^E - \psi_h)\, d\sigma_h \,.
\end{aligned}
\tag{3.91}
$$

Thus, the left-hand side in Equation (3.90) can be decomposed as

$$
\int_\Gamma \left[\nabla_\Gamma (u - u_h^E) \cdot \nabla_\Gamma \psi + \mathbf{v}^E \cdot \nabla_\Gamma (u - u_h^E)\, \psi\right]\, d\sigma = R + G + D \,, \tag{3.92}
$$

where

$$
\begin{aligned}
R = &\int_{\Gamma_h} (f^E \mu_h + \Delta_{\Gamma_h} u_h - \mathbf{v}^E \cdot \nabla_{\Gamma_h} u_h)(\psi^E - \psi_h)\, d\sigma_h + \\
&-\frac{1}{2} \sum_{T\in\Gamma_h} \sum_{e\in\partial T} \int_e [[\nabla_{\Gamma_h} u_h]](\psi^E - \psi_h)\, d\sigma_h \,,
\end{aligned}
$$

represents the *residual*,

$$
G = \int_\Gamma \left[(\mathbf{A}_h^E - \mathbf{P})\nabla_\Gamma u_h^E \cdot \nabla_\Gamma \psi + \mathbf{v}^E \cdot (\mathbf{B}_h^E - \mathbf{P})\nabla_\Gamma u_h^E \psi\right]\, d\sigma,
$$

represents the *geometrical contribution*, due to the absence of the Galerkin orthogonality, while

$$D = \int_{\Gamma_h} (f^E \mu_h - f_h)\psi_h \, d\sigma_h \,,$$

is the *data part*, due to the approximation of the function f via f_h. Now, we analyse separately these three terms by following the proof of Proposition 3.3.6.

Residual Part

By properly exploiting the Cauchy-Schwarz inequality together with estimates (3.6) and (3.56), we have

$$\begin{aligned}
R &= \int_{\Gamma_h} (f^E \mu_h + \Delta_{\Gamma_h} u_h - \mathbf{v}^E \cdot \nabla_{\Gamma_h} u_h)(\psi^E - \psi_h)\, d\sigma_h + \\
&\quad -\frac{1}{2} \sum_{T \in \Gamma_h} \sum_{e \in \partial T} \int_e [[\nabla_{\Gamma_h} u_h]](\psi^E - \psi_h)\, d\sigma_h \leq \\
&\leq \sum_{\mathbf{z} \in \Gamma_h} \left(||f^E \mu_h + \Delta_{\Gamma_h} u_h - \mathbf{v}^E \cdot \nabla_{\Gamma_h} u_h||_{L^2(\omega_\mathbf{z})} ||\psi^E - \psi^E_\mathbf{z}||_{L^2(\omega_\mathbf{z})} + \right. \\
&\quad \left. + \frac{1}{2} \sum_{e \in \omega_\mathbf{z}} |||[[\nabla_{\Gamma_h} u_h]]||_{L^2(e)} ||\psi^E - \psi^E_\mathbf{z}||_{L^2(e)} \right) = \\
&\leq C \sum_{\mathbf{z} \in \Gamma_h} \left(||f^E \mu_h + \Delta_{\Gamma_h} u_h - \mathbf{v}^E \cdot \nabla_{\Gamma_h} u_h||_{L^2(\omega_\mathbf{z})} \sum_{T \in \omega_\mathbf{z}} \left(\sum_{i=1}^{2} s^2_{i,T} (\mathbf{r}_{i,T})^t \, \mathbf{G}_T(\psi^E)\, \mathbf{r}_{i,T} \right)^{1/2} + \right. \\
&\quad \left. + \sum_{e \in \omega_\mathbf{z}} \sqrt{|e|}\, |||[[\nabla_{\Gamma_h} u_h]]||_{L^2(e)} \sum_{T \in \omega_\mathbf{z}} \left(\sum_{i=1}^{2} s^2_{i,T} (\mathbf{r}_{i,T})^t \, \mathbf{G}_T(\psi^E)\, \mathbf{r}_{i,T} \right)^{1/2} \right) = \\
&= C \sum_{T \in \Gamma_h} (r_T + r_{T,e}) \left(\sum_{i=1}^{2} s^2_{i,T} (\mathbf{r}_{i,T})^t \, \mathbf{G}_T(\psi^E)\, \mathbf{r}_{i,T} \right)^{1/2} ,
\end{aligned}$$

where we have exploited the definition of r_T and $r_{T,e}$ in Equation (3.87).

Geometrical Part

The first part of the term G is similar to the one of the geometrical part of Proposition 3.3.6, so we have

$$\int_\Gamma (\mathbf{A}_h^E - \mathbf{P})\,\nabla_\Gamma u_h^E \cdot \nabla_\Gamma \psi \, d\sigma \leq$$
$$\leq C \sum_{T\in\Gamma_h} \frac{h_T^2}{\sqrt{s_{2,T}}} ||\nabla_{\Gamma_h} u_h||_{L^2(T)} \left(\sum_{i=1}^{2} s_{i,T}^2 (\mathbf{r}_{i,T})^t \, \mathbf{G}_T(\psi^E)\, \mathbf{r}_{i,T} \right)^{1/2} . \tag{3.93}$$

Then, the second contribution in the term G is more complex due to the presence of the matrix $(\mathbf{B}_h^E - \mathbf{P})$. First of all we have to find a bound for this term. Since the discrete surface belongs to U_δ, the following relations hold:

$$|d| < ch^2 \quad \text{and} \quad |1-\mu_h| < ch^2 \,,$$

where c is a constant and h is the discretization step, see Lemma 2.4.3. Thus, we have

$$\begin{aligned}
\mathbf{B}_h^E - \mathbf{P} &= \left(\frac{1}{\mu_h} \mathbf{P}_h (\mathbf{I} - d\mathbf{H})\mathbf{P} - \mathbf{I} \right) \mathbf{P} \\
&= \mathbf{P}_h\mathbf{P} - d\mathbf{P}_h\mathbf{H}\mathbf{P}\mathbf{P} - \mathbf{P} + o(h^2) \\
&= \mathbf{P}_h\mathbf{P} - d\mathbf{P}_h\mathbf{H}\mathbf{P} - \mathbf{P} + o(h^2) \\
&= \mathbf{P}_h\mathbf{P} - \mathbf{P} + o(h^2) \,.
\end{aligned}$$

Let us focus on the term $\mathbf{P}_h\mathbf{P} - \mathbf{P}$. We recall that the tensor product between vectors, $\mathbf{n}\otimes\mathbf{n}$, can be written as the product $\mathbf{n}\mathbf{n}^t$, when we identify $\mathbf{n}$ as a column vector. We follow this chain of equalities

$$\begin{aligned}
|\mathbf{P}_h\mathbf{P} - \mathbf{P}| &= |(\mathbf{P}_h - \mathbf{I})\mathbf{P}| \\
&= |(\mathbf{I} - \mathbf{n}_h\otimes\mathbf{n}_h - \mathbf{I})(\mathbf{I} - \mathbf{n}\otimes\mathbf{n})| \\
&= |(\mathbf{I} - \mathbf{n}_h\mathbf{n}_h^t - \mathbf{I})(\mathbf{I} - \mathbf{n}\mathbf{n}^t)| \\
&= |-\mathbf{n}_h\mathbf{n}_h^t + \mathbf{n}_h\mathbf{n}_h^t\mathbf{n}\mathbf{n}^t| \\
&= |\mathbf{n}_h|\,|\mathbf{n}_h^t + \mathbf{n}_h^t\mathbf{n}\mathbf{n}^t| \\
&= |\mathbf{n}_h^t + \mathbf{n}_h^t\mathbf{n}\mathbf{n}^t| \,.
\end{aligned}$$

We orient the reference system to get $\mathbf{n}_h = (0\,,0\,,-1)^t$ and $\mathbf{n} = (n_x\,,n_y\,,n_z)^t$. We consequently have

$$\begin{aligned} |\mathbf{n}_h^t + \mathbf{n}_h^t \mathbf{n}\mathbf{n}^t| &= \sqrt{(n_z n_x)^2 + (n_z n_y)^2 + (n_z n_z)^2} \\ &= \sqrt{1-(n_z)^2} \\ &= \sqrt{n_x^2 + n_y^2} \\ &= |\mathbf{n} \wedge \mathbf{n}_h|\,, \end{aligned}$$

where $\mathbf{n} \wedge \mathbf{n}_h$ is the vector product between $\mathbf{n}$ and $\mathbf{n}_h$. We have generated the mesh Γ_h such that the angle between the normals $\mathbf{n}$ and $\mathbf{n}_h$ is bounded and, in particular, we have $|\mathbf{n} \wedge \mathbf{n}_h| < ch_T$, see [31] and Chapter 2. Thus, it yields:

$$||\mathbf{B}_h^E - \mathbf{P}||_{L^\infty(T)} \leq Ch_T\,. \tag{3.94}$$

Then, if we consider a patch $\omega_{\mathbf{z}}$, we have

$$\max_{T\in\omega_{\mathbf{z}}} ||\mathbf{B}_h^E - \mathbf{P}||_{L^\infty(T)} \leq h_{\omega_{\mathbf{z}}}\,.$$

Consider $\omega_{\mathbf{z}}^E$, i.e., the patch $\omega_{\mathbf{z}}$ lifted on the surface Γ, now we proceed with this sequence of inequalities

$$\begin{aligned} \int_\Gamma \mathbf{v}^E \cdot [\mathbf{B}_h^E - \mathbf{P}]\,\nabla_\Gamma u_h^E \psi\, d\sigma &\leq \sum_{\mathbf{z}\in\Gamma_h} \int_{\omega_{\mathbf{z}}^E} \mathbf{v}^E \cdot [\mathbf{B}_h^E - \mathbf{P}]\,\nabla_\Gamma u_h^E \psi\, d\sigma \\ &\leq c \sum_{\mathbf{z}\in\Gamma_h} \int_{\omega_{\mathbf{z}}^E} \mathbf{v}^E \cdot [\mathbf{B}_h^E - \mathbf{P}]\,\nabla_\Gamma u_h^E \psi\, d\sigma \\ &\leq c \sum_{\mathbf{z}\in\Gamma_h} \max_{T\in\omega_{\mathbf{z}}} ||\mathbf{B}_h^E - \mathbf{P}||_{L^\infty(T)} \int_{\omega_{\mathbf{z}}^E} \mathbf{v}^E \cdot \nabla_\Gamma u_h^E \psi\, d\sigma \\ &\leq c \sum_{\mathbf{z}\in\Gamma_h} h_{\omega_{\mathbf{z}}} ||\mathbf{v}^E \cdot \nabla_{\Gamma_h} u_h||_{L^2(\omega_{\mathbf{z}})} ||\psi^E||_{L^2(\omega_{\mathbf{z}})}\,, \end{aligned}$$

Then, by exploiting the same computation of the Data Part in Proposition 3.3.6, we get

$$||\psi||_{L^2(\omega_{\mathbf{z}}^E)} \leq ||\psi^E||_{L^2(\omega_{\mathbf{z}})} \leq \left(\sum_{i=1}^{2} s_{i,T}^2 (\mathbf{r}_{i,T})^t\, \mathbf{G}_T(\psi^E)\, \mathbf{r}_{i,T} \right)^{1/2} \tag{3.95}$$

Finally, we have

$$\int_{\Gamma} \mathbf{v}^E \cdot [\mathbf{B}_h^E - \mathbf{P}] \nabla_{\Gamma} u_h^E \psi \, d\sigma \leq$$
$$\leq c \sum_{T \in \omega_{\mathbf{z}}} \left(\sum_{\mathbf{z} \in T} h_{\omega_{\mathbf{z}}} ||\mathbf{v}^E \cdot \nabla_{\Gamma_h} u_h||_{L^2(\omega_{\mathbf{z}})} \right) \left(\sum_{i=1}^{2} s_{i,T}^2 (\mathbf{r}_{i,T})^t \, \mathbf{G}_T(\psi^E) \, \mathbf{r}_{i,T} \right)^{1/2} \tag{3.96}$$

By combining Equations (3.93) and (3.96) we complete the estimate for the geometric part.

Data Part

This part of the estimator is essentially the same as the Data Part in Proposition 3.3.6.

This completes the proof.

□

Remark 3.4.1 *In estimate* (3.85) *we have neglected the error due to the discretization of the vector field* $\mathbf{v}^E$.

Remark 3.4.2 *The estimate* (3.85) *is not computable, since the error* e_h *depends on the exact solution. To overcome this issue, we proceed in the same way as the anisotropic a-posteriori error estimator of the Laplace-Beltrami, i.e., we exploit a proper gradient recovery scheme to make this estimate computable, see Section 3.3.*

3.4.1 From the Estimator to an Anisotropic Metric

As in Section 3.3.1, we compute a metric starting from the estimator (3.85) that will drive the adaptation procedure. To achieve this goal, we follow exactly the same approach as in Subsection 3.3.1. Actually, we obtain exactly the same element-wise metric as defined in Equation (3.75), but now the quantity $\overline{\rho}_T(u_h)$ is defined in Proposition 3.4.1.

3.4.2 Numerical Results

In this subsection we numerically show the effectiveness of the mesh adaptation procedure derived from the estimator described in Section 3.4. To achieve this goal, we consider the same quantities taken into account in Subsection 3.3.2 and the tolerances, τ, driving the adaptation procedure are specified in the tables below.

Example 5

We consider the following problem:

$$\begin{cases} -\Delta_\Gamma u + \mathbf{v}\cdot\nabla_\Gamma u &= \frac{1}{(x-1.2)^2} - \frac{1}{(x+0.2)^2} & \text{on } \Gamma_5\,, \\ u &= 0 & \text{on } \partial\Gamma_5\,, \end{cases} \tag{3.97}$$

where $\mathbf{v} = (0., 50., 0.)^t$, while the surface Γ_5 is defined as the zero level set of the implicit function

$$d_5 \,:\, [0,1] \,\times\, [0,1] \,\times\, [0,0.2] \to\, \mathbb{R}\,,$$

defined as:

$$d_5(x,\, y,\, z) := 0.2\sin(\pi x)\sin(\pi y) - z\,. \tag{3.98}$$

The solution u_{ref} is computed on a very fine mesh and it shows three boundary layers along the axes highlighted in Figure 3.22.

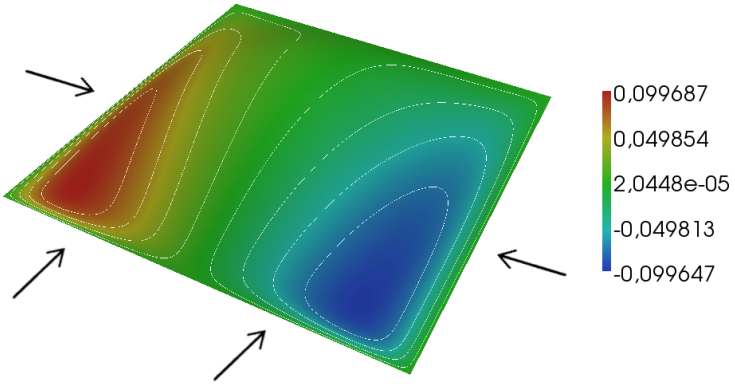


Figure 3.22: Reference solution u_{ref} for example 5, the arrows highlight the boundary layers.

In Figure 3.23, the isotropic and anisotropic adapted meshes are compared. Moreover, from the enlarged views in Figure 3.24 and 3.25, we recognize that the triangles are properly aligned according to the directionality of the solution, i.e., they are stretched according to the direction of the gradient of the solution.

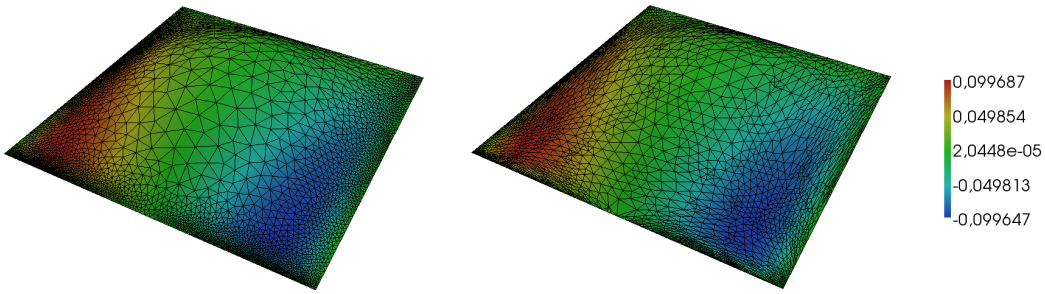


Figure 3.23: Isotropic adapted mesh, on the left, and anisotropic adapted mesh, on the right.

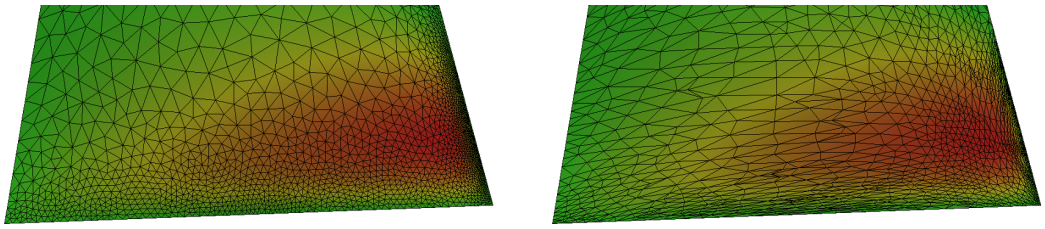


Figure 3.24: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

Then, in Table 3.11, we provide the value of the errors defined in Equation (3.76) and the maximum value of the stretching factor. In particular, we observe that, even in this case, the anisotropic adapted meshes provide better results, with respect to the isotropic grids, both in terms of elements and accuracy.

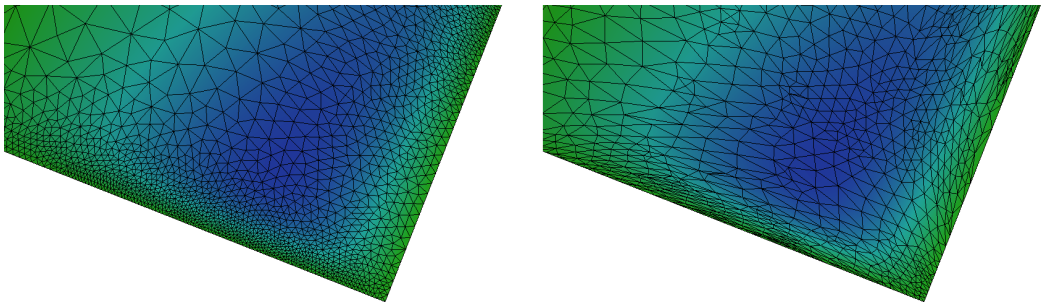


Figure 3.25: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

	iso	ani
Ele.	12351	11128
e_{tot}	4.144e-02	1.171e-02
e_{max}	1.093e-03	2.055e-04
e_{min}	3.737e-09	7.675e-10
s_{max}	2.660e+00	3.080e+01
τ	5.000e-04	3.000e-04

	iso	ani
Ele.	8111	4195
e_{tot}	2.710e-02	2.396e-02
e_{max}	3.349e-04	2.711e-03
e_{min}	1.987e-08	8.400e-09
s_{max}	2.525e+00	2.925e+01
τ	2.000e-03	1.000e-03

Table 3.11: Comparison between the isotropic and anisotropic mesh adaptation procedure for about the same number of elements, on the left, and for about the same accuracy, on the right.

Example 6

We consider the following problem:

$$\begin{cases} -\Delta_\Gamma u + \mathbf{v} \cdot \nabla_\Gamma u &= \frac{1}{(x-1.01)^2} - \frac{1}{(x+0.01)^2} & \text{on } \Gamma_6\,, \\ u &= 0 & \text{on } \partial\Gamma_6\,, \end{cases} \tag{3.99}$$

where $\mathbf{v} = (30., 50., 0.)^t$. The surface Γ_6 is defined by the zero level set of the function

$$d_6 \,:\, [0,1] \,\times\, [0,1] \,\times\, [-0.1, 0.1] \to\, \mathbb{R}\,,$$

defined as:

$$d_6(x,\, y,\, z) := 0.1 \sin(3\pi x) \sin(3\pi y) - z\,. \tag{3.100}$$

The solution u_{ref} is computed on with a very fine mesh and it exhibits three boundary layers along the axes highlighted in Figure 3.26.

As in the previous example we provide both the isotropic and the anisotropic adapted meshes in Figure 3.27. From the details in Figure 3.28, 3.29 and 3.30, we see that the triangles are aligned according to the behaviour of the solution and, in Table 3.12, we numerically prove that the anisotropic adapted meshes offer as expected better results than the isotropic grids.

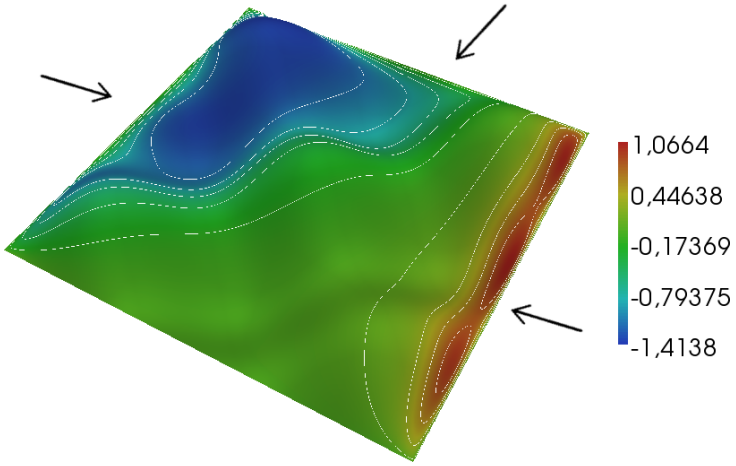


Figure 3.26: Reference solution u_{ref} for example 6, the arrows highlight the boundary layers.

	iso	ani
Ele.	5760	5375
e_{tot}	1.413e+00	5.768e-01
e_{max}	2.125e-02	1.208e-02
e_{min}	1.262e-07	1.639e-08
s_{max}	4.281e+00	3.139e+01
τ	1.200e-01	2.500e-02

	iso	ani
Ele.	5407	2138
e_{tot}	1.518e+00	1.463e+00
e_{max}	2.380e-02	2.638e-02
e_{min}	9.558e-08	1.121e-08
s_{max}	4.585e+00	5.392e+01
τ	1.300e-01	1.300e-01

Table 3.12: Comparison between the isotropic and anisotropic mesh adaptation procedure for about the same number of elements, on the left, and for about the same accuracy, on the right.

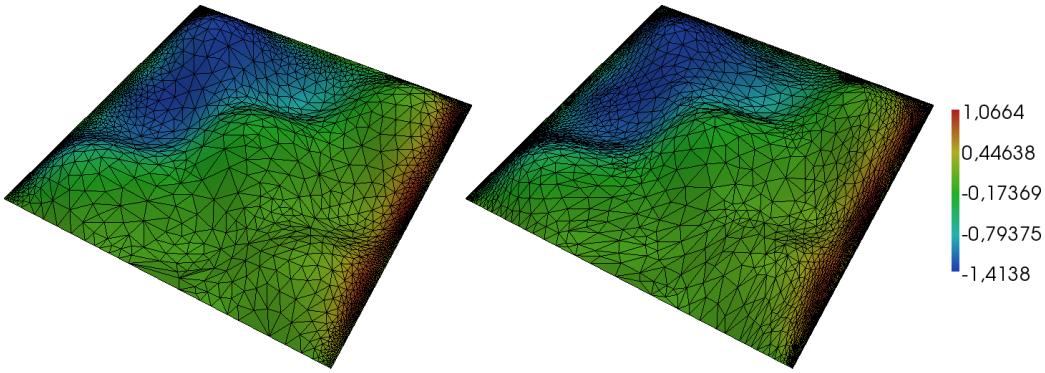


Figure 3.27: Isotropic adapted mesh, on the left, and anisotropic adapted mesh, on the right.

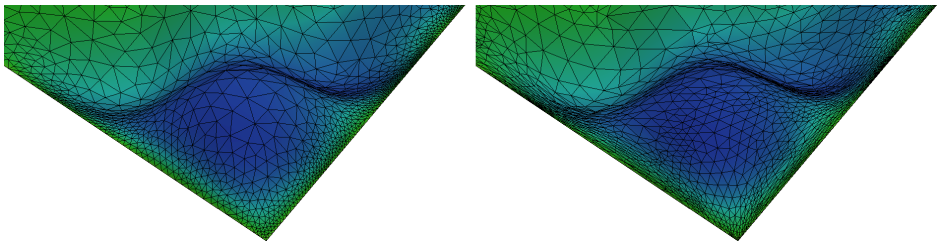


Figure 3.28: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

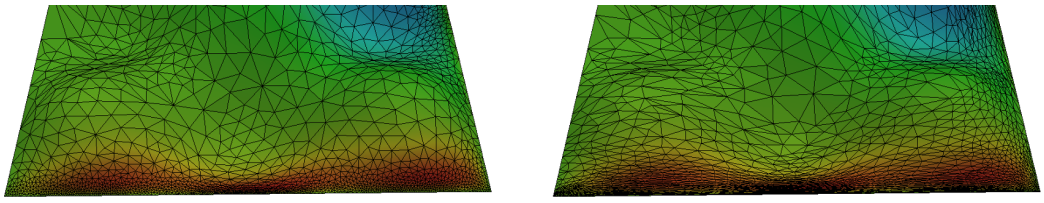


Figure 3.29: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

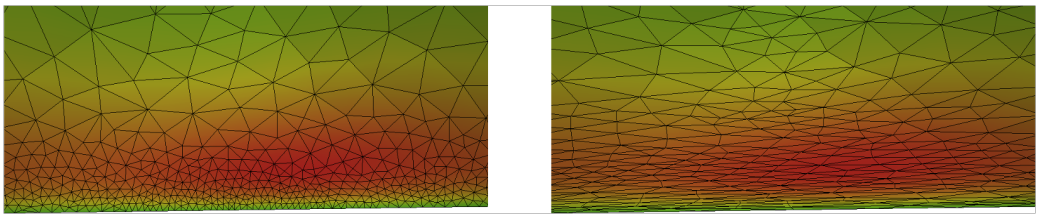


Figure 3.30: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

3.5 Anisotropic Goal-Oriented A-Posteriori Error Analysis

In many applications one may be interested in physically relevant quantity, for instance the concentration of a certain substance in critical areas of the computational domain or localized or mean values, fluxes across sections of the domain. In all these cases, the goal-oriented adaptation procedure is the most useful approach to get a mesh which is suited to approximate a quantity of interest, [5].

There are two main ingredients behind this kind of analysis. The first one is the introduction of a suitable goal functional $J : H_0^1(\Gamma) \to \mathbb{R}$, that mathematically represents the quantity of interest. So far we have considered **only** linear functional. The second one is the definition of an auxiliary problem, the so-called adjoin or dual problem, related in a proper way to the goal functional and the primal problem

$$\mathcal{L}(u) = f \ \text{ in } \Gamma\,, \tag{3.101}$$

completed with suitable boundary conditions on $\partial\Gamma$, where $\mathcal{L}$ is a generic differential operator, f is a sufficiently regular function, Γ is a surface and $\partial\Gamma$ denotes the corresponding boundary. Then, the dual problem is

Problem 3.5.1 *Find $z \in H_0^1(\Gamma)$, such that:*

$$J(\phi) = \mathcal{A}^*(z,\, \phi)\,, \qquad \forall \phi \in H_0^1(\Gamma)\,, \tag{3.102}$$

where $\mathcal{A}^*(\cdot,\cdot)$ denotes the adjoin bilinear form associated with the primal one $\mathcal{A}^*(\cdot,\cdot)$, Problem 2.3.1 or 2.5.1.

Let us define the discrete version of Problem 3.5.1. We consider a counterpart of the functional J, associated with the polyhedral surface, i.e., a functional $J_h :\ H_0^1\ (\Gamma_h) \to \mathbb{R}$, so that the discrete version of Problem (3.5.1) is:

Problem 3.5.2 *Find $z_h \in H_0^1(\Gamma_h)$ such that:*

$$J_h(\phi_h) = \mathcal{A}_h^*(z_h,\, \phi_h)\,, \qquad \forall \phi_h \in H_0^1(\Gamma_h)\,, \tag{3.103}$$

where now $\mathcal{A}_h^*(\cdot,\cdot)$ is the adjoin bilinear form associated with the discrete primal problem, Problem 2.3.2 or 2.5.1.

Now, we can provide a goal-oriented anisotropic a-posteriori error estimator for the functional error $J(e_h)$.

Proposition 3.5.1 *Let u and z be the solutions to the primal and the dual problem and let u_h and z_h be the solutions of their corresponding finite element approximations. Then, the following estimate for the functional error e_h defined in Equation (2.41) holds:*

$$|J(e_h)| \leq \sum_{T \in \Gamma_h} \rho_T^*(u_h)(s_{1,T}^2(\mathbf{r}_{1,T})^t\, \mathbf{G}_T(e_{zz})\, \mathbf{r}_{1,T} + s_{2,T}^2(\mathbf{r}_{2,T})^t\, \mathbf{G}_T(e_{zz})\, \mathbf{r}_{2,T})^{1/2}\,, \tag{3.104}$$

where the quantity $\rho_T^*(u_h)$ depends on the primal problem. More precisely, it is defined in Equation (3.60) or Equation (3.86) for the Problem 2.3.2 or 2.5.2, respectively, and $e_{zz} = z^E - z_h$.

Proof. Let us choose the test function ϕ of the dual problem coinciding with e_h. This immediately yields:

$$J(e_h) = \mathcal{A}^*(z,\, e_h) = \mathcal{A}(e_h,\, z)\,,$$

thanks to the Lagrange identity. At this point we can follow exactly the proof of Proposition 3.3.6 or of Proposition 3.4.2 if we are considering Problem 2.3.2 or 2.5.2, but ψ needs to be replaced by the dual solution z.

□

Even in this case, the right hand side of Equation (3.104) depends on the exact solution of the adjoin problem defined in Equation (3.103). As in the previous section, the idea is to replace z with a computable quantity obtained via a suitable recovered gradient procedure. In particular, we adopt the generalization of the standard Zienkiewicz-Zhu error estimator to surface embedded in the three-dimensional space proposed by [110]. Hence, starting from the discrete solution

z_h of Problem (3.5.2), we introduce the tangential recovered gradient $\nabla_{\Gamma_h}^{WCY} z_h$, to define the components of the recovered matrix $\mathbf{G}_T(e_{zz}^*)$ as

$$\begin{aligned}[\mathbf{G}_T(e_{zz}^*)]_{i,j} &= \int_T \frac{\partial e_{zz}^*}{\partial x_i}\frac{\partial e_{zz}^*}{\partial x_j} \\ &= \int_T \Big((\nabla_{\Gamma_h}^{WCY} z_h)_i - (\nabla_{\Gamma_h} z_h)_i\Big)\Big((\nabla_{\Gamma_h}^{WCY} z_h)_j - (\nabla_{\Gamma_h} z_h)_j\Big)\,,\end{aligned} \tag{3.105}$$

where $i,j = 1,2,3$ run on the components of the discrete gradient ∇_{Γ_h}, while $x_1 = x$, $x_2 = y$, $x_3 = z$.

3.5.1 From the Estimator to an Anisotropic Metric

As in the a-posteriori analysis based on the control of the energy norm, starting from the estimate in Equation (3.104), we have to derive a suitable metric to drive the mesh adaptation procedure. Even in this case, we fix a priori a tolerance on the total error and then we "equidistribute the error" by averaging the contribution given by each element. We consider

$$\rho_T^*(u_h)(s_{1,T}^2(\mathbf{r}_{1,T})^t\,\mathbf{G}_T(e_{zz})\,\mathbf{r}_{1,T} + s_{2,T}^2(\mathbf{r}_{2,T})^t\,\mathbf{G}_T(e_{zz})\,\mathbf{r}_{2,T})^{1/2} = \frac{\tau}{\#\mathcal{T}}\,, \tag{3.106}$$

where $\#\mathcal{T}$ are the number of triangles in the initial mesh Γ_h, and we follow the same computations as in Subsection 3.3.1 to get the piecewise constant metric. Finally, the desired metric is given by putting

$$\sigma_1^2 = q s_T\,, \quad \sigma_2^2 = \frac{q}{s_T}\,, \quad \mathbf{u}_1 = \mathbf{w}_2\,, \quad \mathbf{u}_2 = \mathbf{w}_1\,,$$

in Equation (1.14), where $\mathbf{w}_1$, $\mathbf{w}_2$ are the eigenvectors of $\mathbf{G}_T(e_{zz}^*)$ corresponding to the eigenvalue μ_1 and μ_2, respectively, with $\mu_1 \geq \mu_2$, $s_T = \sqrt{\mu_1/\mu_2}$ and

$$q = \sqrt[3]{\frac{4\tau^2}{\overline{\rho}_T(u_h)\nu_{\min}}} \quad \text{and} \quad \nu_{\min} := \sqrt{\frac{\mu_1}{\mu_2}}\mu_2 + \sqrt{\frac{\mu_2}{\mu_1}}\mu_1\,.$$

3.5.2 Numerical Results

In this section we numerically check the effectiveness of the proposed goal-oriented mesh adaptation procedure. In particular, we focus on the following quantities:

a) the *number of the elements*;

b) the *stretching factor*: defined as in Equation (3.37);

c) the *value of the error*: we consider a reference value $J(u_{\text{ref}})$, where u_{ref} is a discrete solution associated with a really fine surface mesh. Then, we compute the relative error:
$$e_J := \frac{|J(u_{\text{ref}}) - J_h(u_h)|}{|J(u_{\text{ref}})|}\,, \tag{3.107}$$
where J_h is the functional computed on the adapted mesh.

d) the *isotropic case*: all the quantities in a) b) and c) are evaluated also in the case of an isotropic mesh adaptation to investigate the performances of both these two mesh adaptation processes. As in the previous section, the comparison is performed in terms of number of elements and of accuracy.

The tolerances, τ, driving the adaptation procedure are specified in the tables below.

Example 7

We consider the Laplace-Beltrami problem defined in Equation (3.78) and the functional
$$J_1(\phi) = \int_{\Gamma_h} \nabla_\Gamma \phi \cdot \nabla_\Gamma e_h \, d\sigma\,, \tag{3.108}$$
where Γ_h is the surface adapted mesh.

In Figure 3.31, we show the whole mesh while some details are furnished in Figure 3.32 and 3.33. Table 3.13 shows that if we consider an isotropic and an anisotropic mesh with about the same value for the relative error, Equation (3.107), the anisotropic grid demands a lower number of elements.

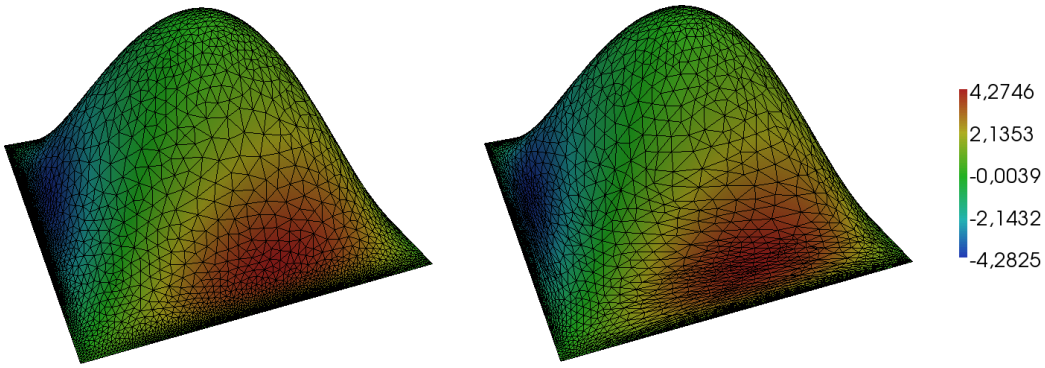


Figure 3.31: Isotropic adapted mesh, on the left, and anisotropic adapted mesh, on the right.

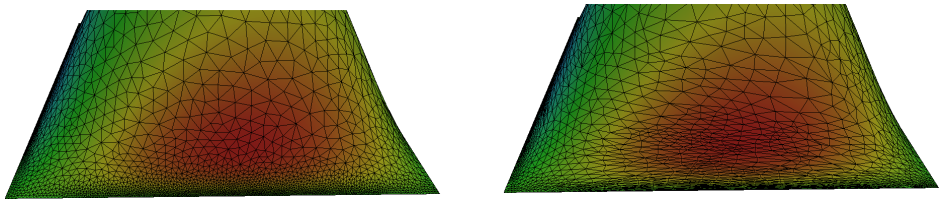


Figure 3.32: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

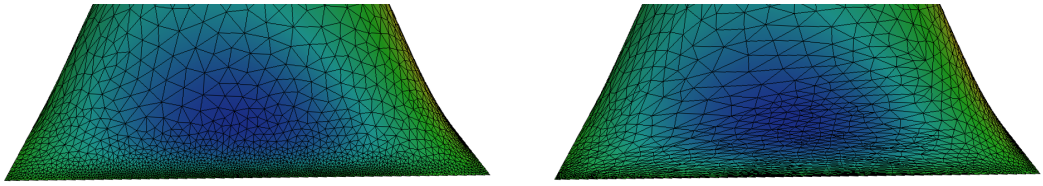


Figure 3.33: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

	Ele.	e_{J_1}	$s_{\max}$	τ
iso	9394	5.225e-03	3.089e+00	4.000e-02
ani	4962	4.208e-03	3.115e+01	1.000e-02

Table 3.13: Comparison between the isotropic and anisotropic mesh adaptation for about the same accuracy on e_{J_1}.

Example 8

We consider the same problem as in the previous example but for a different choice of the goal functional, now we consider

$$J_2(\phi) = \int_{\Omega_h} \nabla_\Gamma \phi \cdot \nabla_\Gamma e_h \, d\sigma \,, \tag{3.109}$$

where $\Omega_h \subset \Gamma_h$ is the portion of the surface, as highlighted in Figure 3.34 on the left.

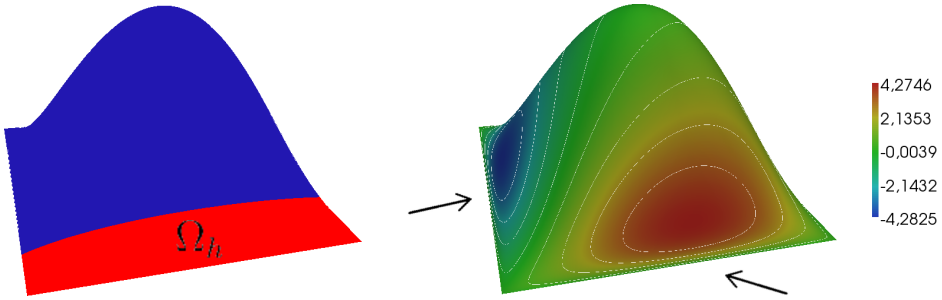


Figure 3.34: Portion Ω_h of the surface associated with the functional J_2 is defined, on the left; the reference solution u_{ref} , on the right.

In Figure 3.35, we show the resulting mesh for both the isotropic and the anisotropic case. Since the functional involves **only** a portion of the domain, Ω_h, **only** this part of the surface Γ_h is properly adapted. The solution problem defined in Equation (3.78) presents two boundary layers as it is highlighted in Figure 3.34 on the right. We notice that only the boundary layer inside Ω_h is captured, see Figure 3.36, while the other one is not detected since it lies outside Ω_h, see Figure 3.37.

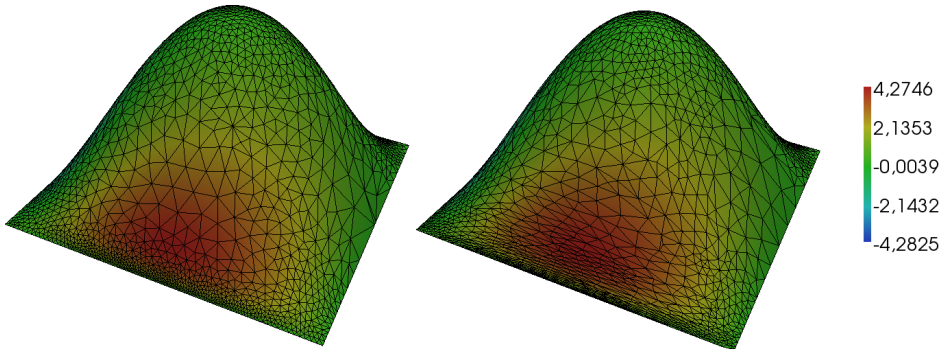


Figure 3.35: Isotropic adapted mesh, on the left, and anisotropic adapted mesh, on the right.

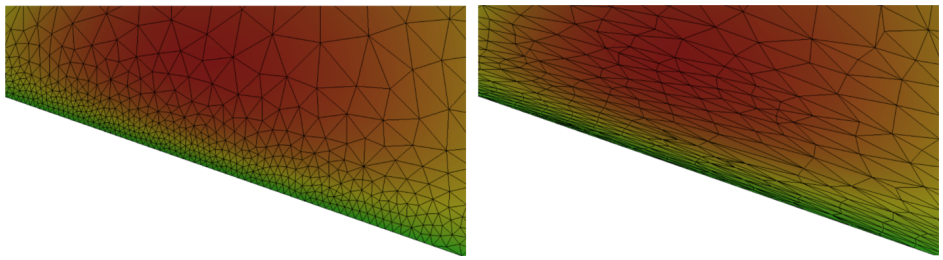


Figure 3.36: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

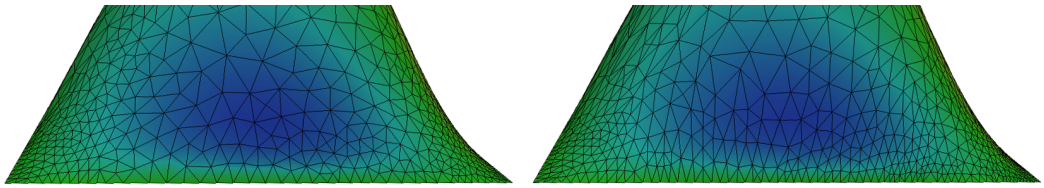


Figure 3.37: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

From Table 3.14, we deduce that, if we fix about the same number of elements, the anisotropic adapted mesh provides a better approximation of the functional J_2 than the isotropic one.

	Ele.	e_{J_2}	$s_{\max}$	τ
iso	3541	1.491e-02	2.492e+00	4.200e-01
ani	3202	7.805e-03	4.619e+01	4.000e-02

Table 3.14: Comparison between the isotropic and anisotropic mesh adaptation for about the same number of elements.

Example 9

We consider the convection-diffusion problem defined in Equation (3.97) and the functional

$$J_3(\phi) = \int_{\Gamma_h} \nabla_\Gamma \phi \cdot \nabla_\Gamma e_h \, d\sigma_h \, . \tag{3.110}$$

where Γ_h is the surface adapted mesh.

In Figure 3.38 we show the whole mesh for both an isotropic and an anisotropic mesh adaptation. Then, the details in Figure 3.39 and 3.40 on the right highlight the anisotropic nature of the mesh. A more quantitative analysis on this test is provided by Table 3.15: the high value of $s_{\max}$ underlines the anisotropic nature of the mesh. Moreover, if we fix about the same value for the error defined in Equation (3.107), the anisotropic mesh guarantees the desired accuracy but with a reduced number of elements than the isotropic grid.

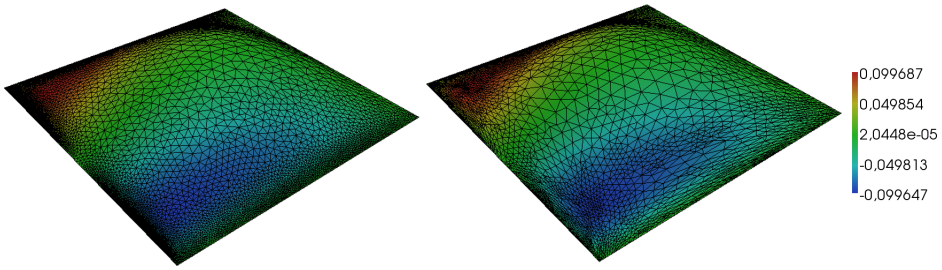


Figure 3.38: Isotropic adapted mesh, on the left, and anisotropic adapted mesh, on the right.

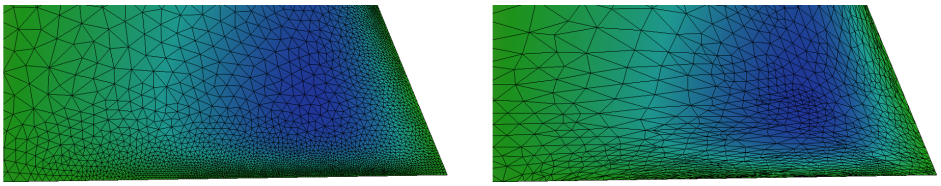


Figure 3.39: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

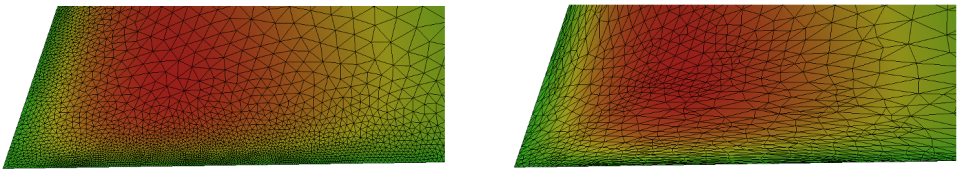


Figure 3.40: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

	Ele.	e_{J_3}	$s_{\max}$	τ
iso	6087	5.409e-03	2.841e+00	1.000e-06
ani	2283	6.183e-03	2.219e+01	1.000e-06

Table 3.15: Comparison between the isotropic and anisotropic mesh adaptation for about the same accuracy on e_{J_3}.

Example 10

We consider the convection-diffusion problem defined in Equation (3.99) and the functional

$$J_4(\phi) = \frac{1}{|\Omega_h|} \int_{\Omega_h} \phi \, d\sigma \,, \tag{3.111}$$

where $\Omega_h \subset \Gamma_h$ is the portion of the surface highlighted in Figure 3.41 on the left, while $|\Omega_h|$ denotes the corresponding area.

As before, the region Ω_h involved in the definition of the functional includes only one of the three boundary layers of the solution. Thus, we expect that in the final adapted mesh the triangles will be **not** adapted and suitably oriented to

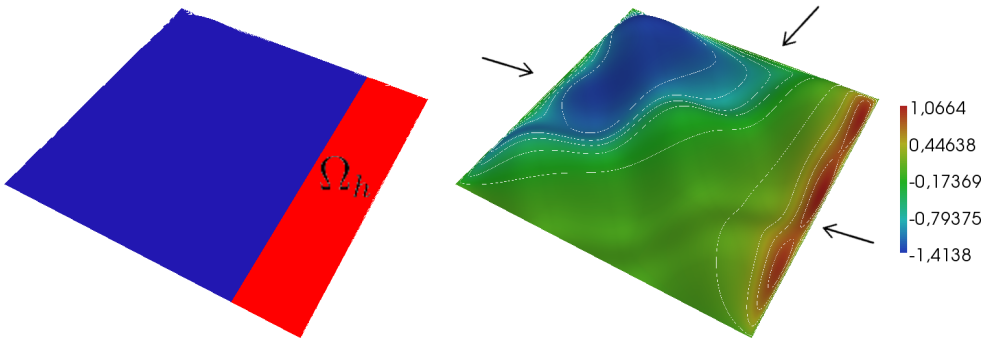


Figure 3.41: Portion Ω_h of the surface associated with the functional J_2 is defined, on the left; the reference solution u_{ref} , on the right.

follow the two layers outside Ω_h, as shown in Figure 3.42 and by the details in Figure 3.43 and 3.44.

In Table 3.16 we compare the performance of the isotropic with the anisotropic mesh adaptation procedure. In particular, we fix about the same number of elements and we observe that the anisotropic adapted mesh provides an approximation for J_4 more accurate than the one obtained by an isotropic mesh. Moreover, the high value of s_{max} confirms the strong anisotropic nature of the adapted mesh.

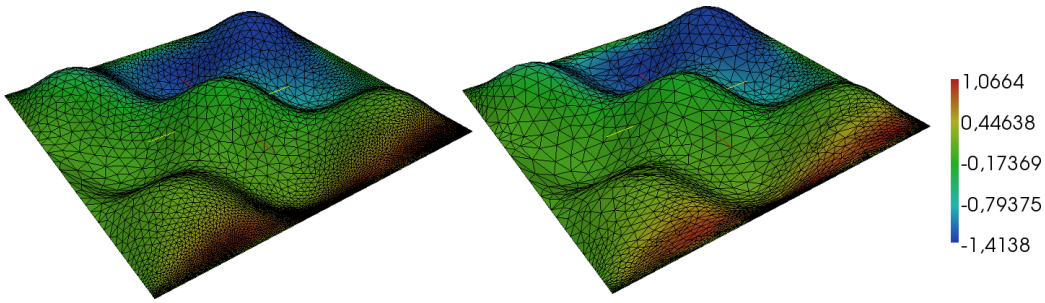


Figure 3.42: Isotropic adapted mesh, on the left, and anisotropic adapted mesh, on the right.

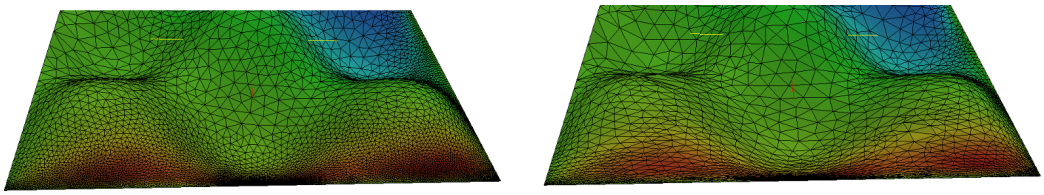


Figure 3.43: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

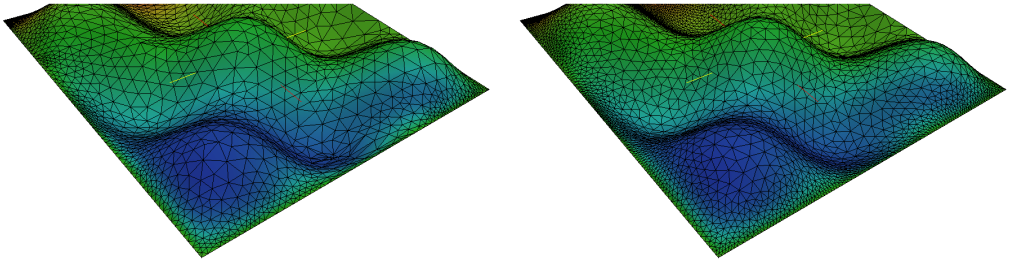


Figure 3.44: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

	Ele.	e_{J_4}	$s_{\max}$	τ
iso	12400	4.465e-04	5.126e+00	1.800-03
ani	12007	7.621e-05	2.677e+02	7.000-03

Table 3.16: Comparison between the isotropic and anisotropic mesh adaptation for about the same number of elements.

Example 11

Now we solve again problem defined in Equation (3.99) by picking functional

$$J_5(\phi)_= \int_{\Omega_h} \nabla_\Gamma \phi \cdot \nabla_\Gamma e_h \, d\sigma \,, \tag{3.112}$$

where Ω_h is the portion of the surface highlighted in Figure 3.45 on the left.

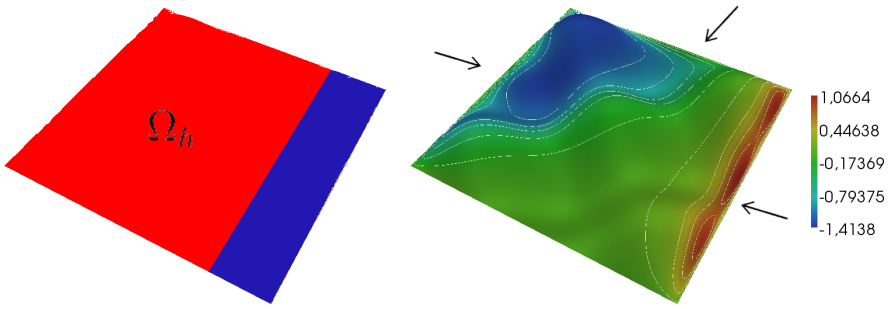


Figure 3.45: Portion Ω_h of the surface associated with the functional J_2 is defined, on the left; the reference solution u_{ref} , on the right.

Now, the region Ω_h contains two of the three boundary layers characterizing the solution u_{ref} . From Figure 3.46 and from the details in Figure 3.47 and 3.48, we see that these two boundary layers are properly refined by the proposed mesh adaptation and, from Figure 3.48, we see that the triangles are aligned in the right direction. Finally Figure 3.49 shows that the other boundary layer is not refined in both adapted meshes.

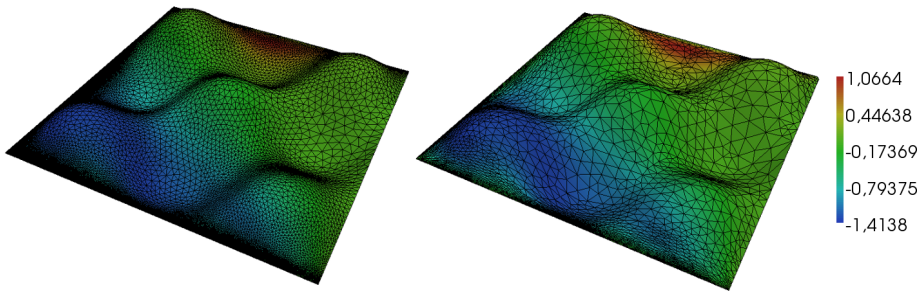


Figure 3.46: Isotropic adapted mesh, on the left, and anisotropic adapted mesh, on the right.

Then, Table 3.17 compare the performance of the isotropic with the anisotropic mesh adaptation procedure. We fix about the accuracy on the functional J_5, and we notice that the anisotropic mesh guarantees the target accuracy with a reduced number of elements than the isotropic grid.

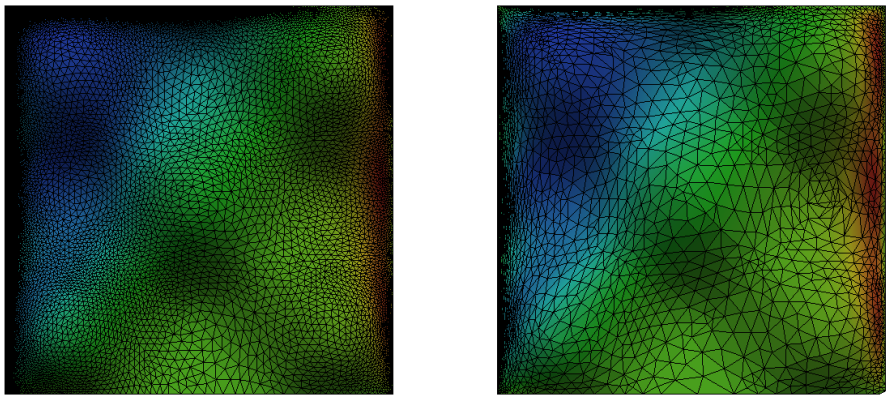


Figure 3.47: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

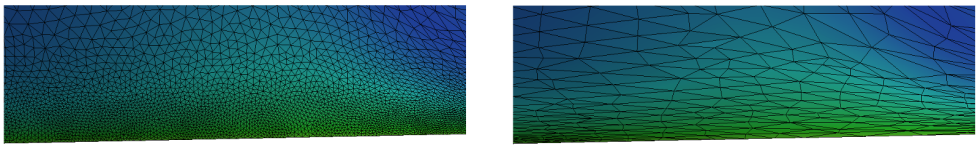


Figure 3.48: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

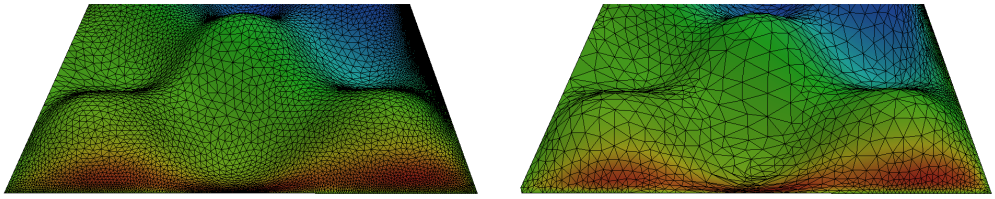


Figure 3.49: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

	Ele.	e_{J_5}	$s_{\max}$	τ
iso	47219	1.421e-03	3.821e+00	5.000e-03
ani	10495	1.314e-03	6.358e+01	7.500e-03

Table 3.17: Comparison between the isotropic and anisotropic mesh adaptation for about the same accuracy on e_{J_5}.

3.6 An Anisotropic Recovery Based Error Analysis

3.6.1 Motivation

The error estimation technique introduced by O. C. Zienkiewicz and J. Z. Zhu in [114, 115, 113] is one of the most popular strategy used in literature. The idea behind this procedure is very simple. We consider the generic Partial Differential Equation:

$$\mathcal{L}(u) = f \ \text{ in } \Omega\,, \tag{3.113}$$

completed with suitable boundary conditions on $\partial\Omega$, where $\mathcal{L}$ is a generic differential operator, f is a sufficiently regular function, Ω is a generic two-dimensional or three-dimensional domain and $\partial\Omega$ denotes the corresponding boundary.

Suppose that we have a finite element approximation u_h of the exact solution u. Moving from u_h and the corresponding gradient ∇u_h, we build an approximation $\nabla^* u_h$ of the gradient of u, more accurate than the gradient ∇u_h of the finite element solution. Then, the Zienkiewicz-Zhu error estimator exploits the discrepancy

$$||\nabla^* u_h - \nabla u_h||_{L^2(\Omega)}\,, \tag{3.114}$$

to estimate the H^1-semi-norm of the discretization error, i.e.,

$$|u - u_h|_{H^1(\Omega)} \simeq ||\nabla^* u_h - \nabla u_h||_{L^2(\Omega)}\,. \tag{3.115}$$

One of the most interesting aspects of this kind of estimators concerns the independence from the problem at hand, except for the selected finite element space. In practice, we only need:

1. the finite element solution, u_h, or just its gradient ∇u_h;
2. a gradient recovery technique to build $\nabla^* u_h$.

In the next sections we will extend this technique with respect to two aspects:

i) we extend this approach to Partial Differential Equations defined on surfaces [31] by resorting to the gradient recovery procedures detailed in Subsection 2.6.2;

ii) we enrich the standard Zienkiewicz-Zhu estimator with directional and orientation information by generalizing what performed for the planar case in [89, 36].

This new error estimator allows us to deal with an anisotropic a-posteriori mesh adaptation procedure for Partial Differential Equation defined on surface embedded in the three-dimensional space.

3.6.2 A Zienkiewicz-Zhu like Error Estimator

Driven by Proposition 3.2.1 we devise an anisotropic a-posteriori error estimator for a function defined on a surface. Let

$$e_{ZZ}^{T} := (\mathcal{GT}_h(\nabla_{\Gamma_h} u_h) - \nabla_{\Gamma_h} u^E)|_T \,, \tag{3.116}$$

be the so called recovered approximation error for the gradient over a generic triangle T of the polyhedral surface Γ_h.

Remark 3.6.1 *Since Γ_h is not necessary a subset of Γ, in Equation (3.116) we have to consider the extension u^E of the exact solution u.*

Generalizing what done in [36], we define the anisotropic Zienkiewicz-Zhu local error estimator for the H^1-semi-norm of the discretization error as

$$[\eta_{T,A}]^2 := \frac{3}{s_{1,T} s_{2,T}} \left(s_{1,T}^2 (\mathbf{r}_{1,T})^t \, \mathbf{G}_T(e_{ZZ}^T) \, \mathbf{r}_{1,T} + s_{2,T}^2 (\mathbf{r}_{2,T})^t \, \mathbf{G}_T(e_{ZZ}^T) \, \mathbf{r}_{2,T} \right) \,, \tag{3.117}$$

where $\mathbf{G}_T$ is the matrix defined in Equation (3.7). Then, we obtain the global associated error estimator as

$$[\eta_A]^2 := \sum_{T \in \Gamma_h} [\eta_{T,A}]^2 \,. \tag{3.118}$$

We do not have a rigorous derivation of this estimator. It is essentially heuristic even though it is inspired by the procedure followed in the proof of Proposition 3.2.2, where we choose $\psi = u - u_h^E$ and we substitute the exact tangential gradient, $\nabla_\Gamma u$, with the recovered one, $\mathcal{GT}_h(\nabla_{\Gamma_h} u_h)$.

However some rationale can be provided. First of all the factor $s_{1,T}s_{2,T}$ in Equation (3.6) guarantees a sort of consistency with respect to the isotropic case. In fact, if we consider $s_{1,T} = s_{2,T}$, the estimators defined in Equation (3.117) and (3.118) become isotropic:

$$[\eta_{T,I}]^2 := 3 \int_T |e^T_{ZZ}| d\sigma_h \quad \text{and} \quad [\eta_I]^2 := \sum_{T \in \Gamma_h} [\eta_{T,I}]^2 . \tag{3.119}$$

Then, the factor 3 in Equation (3.117) can be justified in such a way: the recovery procedure $\mathcal{GT}_h(\nabla_{\Gamma_h} u_h)$ involves three patches to get the recovered gradient on T, see Figure 3.50.

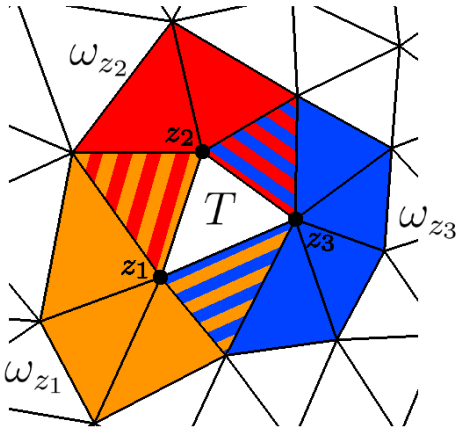


Figure 3.50: Patches influencing triangle T, i.e., $\omega_{\mathbf{z}_1}$, $\omega_{\mathbf{z}_2}$ and $\omega_{\mathbf{z}_3}$.

Finally, Lemma 3.3.4 states a sort of equivalence between $[\eta_A]^2$ and $|u^E - u_h|^2_{H^1(T)}$.

3.6.3 From the Estimator to an Anisotropic Metric

To define a piecewise constant metric field starting from estimator (3.117) and (3.118), we first re-write the local estimator as follows:

$$[\eta_{A,T}]^2 \quad = \quad \frac{3}{s_{1,T}s_{2,T}} \left(s^2_{1,T} (\mathbf{r}_{1,T})^t \, \mathbf{G}_T(e^T_{ZZ}) \, \mathbf{r}_{1,T} + s^2_{2,T} (\mathbf{r}_{2,T})^t \, \mathbf{G}_T(e^T_{ZZ}) \, \mathbf{r}_{2,T} \right)$$

$$
\begin{aligned}
&= 3\underbrace{\left(s_T(\mathbf{r}_{1,T})^t\,\mathbf{G}_T(e^T_{ZZ})\,\mathbf{r}_{1,T} + \frac{1}{s_T}(\mathbf{r}_{2,T})^t\,\mathbf{G}_T(e^T_{ZZ})\,\mathbf{r}_{2,T}\right)}_{(I)}\\
&= \frac{3}{2}|T|\left(s_T(\mathbf{r}_{1,T})^t\,\overline{\mathbf{G}}_T(e^T_{ZZ})\,\mathbf{r}_{1,T} + \frac{1}{s_T}(\mathbf{r}_{2,T})^t\,\overline{\mathbf{G}}_T(e^T_{ZZ})\,\mathbf{r}_{2,T}\right)\\
&= \frac{3}{2}|T|\nu_T(s_T,\mathbf{r}_1)\,,
\end{aligned}
$$

where we have introduced the stretching factor $s_T = s_{1,T}/s_{2,T}$ together with the scaled matrix $\overline{\mathbf{G}}_T$ to adimensionalize the quantity (I), and with

$$
\nu_T(s_T,\mathbf{r}_{1,T}) := \left(s_T(r_{1,T})^t\,\overline{\mathbf{G}}_T(e^T_{ZZ})\,\mathbf{r}_{1,T} + \frac{1}{s_T}(\mathbf{r}_{2,T})^t\,\overline{\mathbf{G}}_T(e^T_{ZZ})\,\mathbf{r}_{2,T}\right)\,,
$$

Since $\mathbf{r}_{1,T}$ and $\mathbf{r}_{2,T}$ are orthonormal, the dependence of $\nu_T(s_T,\mathbf{r}_{1,T})$ on $\mathbf{r}_{2,T}$ is implicit.

Then, given a positive tolerance τ, we impose an equidistribution criterion of the predicted error:

$$
\eta_{A,T} = \frac{\tau}{\#\mathcal{T}}\,, \tag{3.120}
$$

where $\#\mathcal{T}$ is the number of elements of the mesh. Equation (3.120) is not enough to uniquely determine $s_{1,T}$, $s_{2,T}$ and $\mathbf{r}_{1,T}$. To achieve this goal we proceed in the same way as for the derivation of the anisotropic a-priori metric, see Subsection 3.2.2. We impose condition (3.120) by contemporary looking for the most economical adapted mesh. This corresponds to satisfy condition (3.120) with the maximum possible value of the area for the triangle T, i.e., under the constraint that ν_T is minimal. Also in this case we may derive an explicit formula for the optimal solution we are looking for, thus we have

$$
\mathbf{r}_{1,T} = \mathbf{w}_2\,, \quad \mathbf{r}_{2,T} = \mathbf{w}_1\,, \quad s_T = \sqrt{\frac{\mu_1}{\mu_2}}
$$

where $\mathbf{w}_1$, $\mathbf{w}_2$ and μ_1, μ_2 are the eigenvectors and eigenvalues of the matrix $\overline{\mathbf{G}}_T(e^T_{ZZ})$ with $\mu_1 \geq \mu_2$.

Moving from these relations and imposing condition (3.120), we may identify the piecewise constant metric with

$$
\sigma_1^2 = \frac{4\tau^2}{3(\#\mathcal{T})^2\nu_{\min}}\sqrt{\frac{\mu_1}{\mu_2}}\,, \quad \sigma_2^2 = \frac{4\tau^2}{3(\#\mathcal{T})^2\nu_{\min}}\sqrt{\frac{\mu_2}{\mu_1}}\,, \quad \mathbf{u}_1 = \mathbf{w}_2\,, \quad \mathbf{u}_2 = \mathbf{w}_1\,,
$$

in Equation (1.14), where

$$\nu_{\min} := \sqrt{\frac{\mu_1}{\mu_2}}\mu_2 + \sqrt{\frac{\mu_2}{\mu_1}}\mu_1 \, .$$

3.6.4 Numerical Results

In this section we numerically check the reliability of the proposed Zienkiewicz-Zhu error estimator and of the corresponding mesh adaptation procedure. In more details, in Example 12 we compare the behaviour of both the estimators $\eta_{A,T}$ and $\eta_{I,T}$. Then, in Examples 13, 14 and 15, we perform a more quantitative analysis following what we have done in Subsections 3.3.2 and 3.4.2. In particular, we take into account the following quantities:

a) the *number of the elements*;

b) the *stretching factor*: the same quantity defined in Equation (3.37);

c) the *evaluation of the error*: since we proceed with an adaptation procedure driven by an anisotropic estimate for the $H^1(\Gamma_h)$-seminorm of the error and we know the exact function u, we can compute the exact value

$$|u^E - u_h|_{H^1(\Gamma_h)} \, ,$$

where u^E is the extension of u, see Equation (2.9);

d) the *isotropic case*: we evaluate the quantity above also in the case of an isotropic mesh adaptation to compare the performances of these two mesh adaptation processes.

The tolerances, τ, driving the adaptation procedure are specified in the tables below.

Example 12

In this example we evaluate the reliability of both the estimator η_A and η_I. To perform this comparison we also compute the so-called effectivity index associated with the error indicators η_A, and η_I, given by

$$E.I._* := \frac{\eta_*}{|u^E - u_h|_{H^1(\Gamma_h)}} \tag{3.121}$$

where $*$ refers to A or I according to the desired estimator, while u^E is the extension of u and u_h is the discrete approximation for the exact function u. In this example, we consider

$$u(x, y, z) = \tanh(10x), \tag{3.122}$$

while the surface is given by the zero level set of the function d_4 defined in Equation (3.79).

We recall that an error indicator is asymptotically exact if the corresponding effectivity index converges to one as the mesh size goes to zero. This kind of analysis is a standard practice in engineering applications [114, 79, 36]. The theoretical analysis has been carried out under the hypothesis of uniform triangulation, [67, 68], or of fully-structured or globally-structured meshes, [30, 73, 112]. As far as we know, a generalization of this analysis is currently not available in the literature.

In Figure 3.51 we show the isotropic and the anisotropic meshes. Then, in Table 3.18 and 3.19, we provide the values of η_*, $|u^E - u_h|_{H^1(\Gamma_h)}$ and $E.I._*$ for both the isotropic and the anisotropic error estimator on a sequence of adapted meshes with an increasing number of elements.

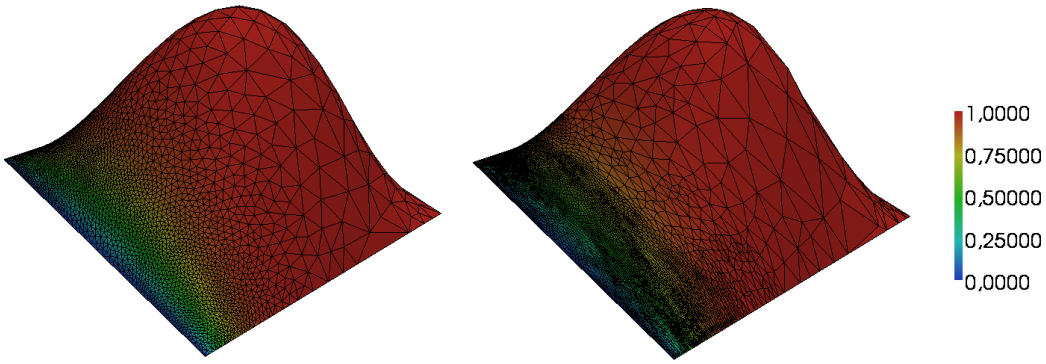


Figure 3.51: Isotropic adapted mesh, on the left, and anisotropic adapted mesh, on the right.

Ele.	η_A	$\lvert u^E - u_h\rvert_{H^1(\Gamma_h)}$	$E.I._A$
271	5.318e-01	3.757e-01	1.431e+00
561	1.935e-01	1.687e-01	1.147e+00
1050	1.277e-01	1.356e-01	9.419e-01
2119	9.361e-02	9.991e-02	9.396e-01
4023	7.277e-02	7.451e-02	9.767e-01
9154	4.668e-02	5.521e-02	8.453e-01

Table 3.18: Behaviour of the anisotropic estimator (3.117).

Ele.	η_I	$\lvert u^E - u_h\rvert_{H^1(\Gamma_h)}$	$E.I._I$
342	6.735e-01	3.983e-01	1.690e+00
799	2.317e-01	1.496e-01	1.548e+00
1601	1.474e-01	9.617e-02	1.533e+00
3765	9.164e-02	5.886e-02	1.577e+00
9041	5.876e-02	3.833e-02	1.533e+00

Table 3.19: Behaviour of the isotropic estimator (3.119).

Example 13

In this example we consider the exact function

$$u(x, y, z) = \tanh(10y) - \tanh(10(x - y) - 5), \tag{3.123}$$

defined on the surface Γ_4 given by the zero level set of the function

$$d_7 : [-2, 2] \times [-2, 2] \times [-0.2, 0.2] \to \mathbb{R},$$

such as

$$d_7(x, y, z) := 0.2 \sin(0.5\pi x) \sin(0.5\pi y) - z. \tag{3.124}$$

From Figure 3.52, we detect that u exhibits two internal layers crossing each other. We apply both an isotropic and anisotropic mesh adaptation driven by the metric field associated with the Zienkiewicz-Zhuerror estimator, η_I and η_A. If we consider the anisotropic adapted mesh, Figure 3.53 on the right together with the details in Figure 3.54 and 3.55 on the right, we may appreciate that the triangles are suitably aligned with the layers, while the isotropic adapted mesh simply refines the regions where layers occur.

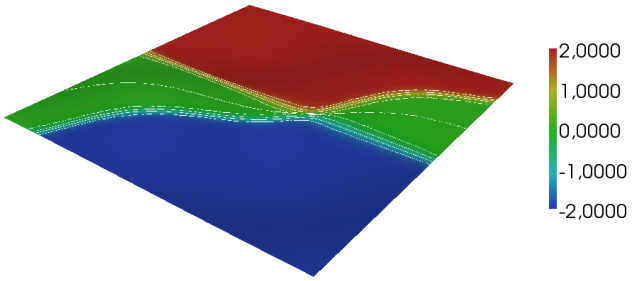


Figure 3.52: The exact function u of example 13, the iso-lines highlight the presence of the boundary layer.

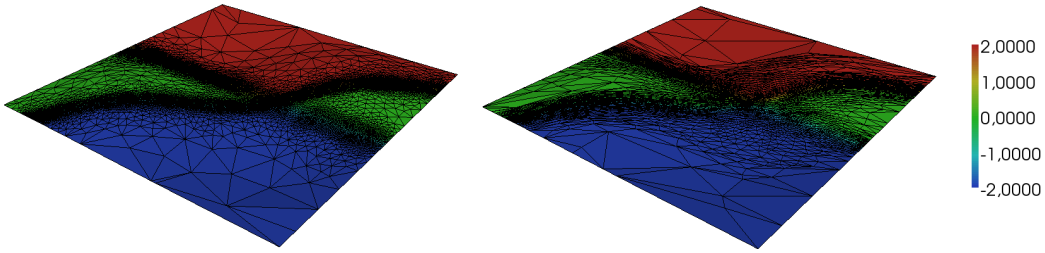


Figure 3.53: Isotropic adapted mesh, on the left, and anisotropic adapted mesh, on the right.

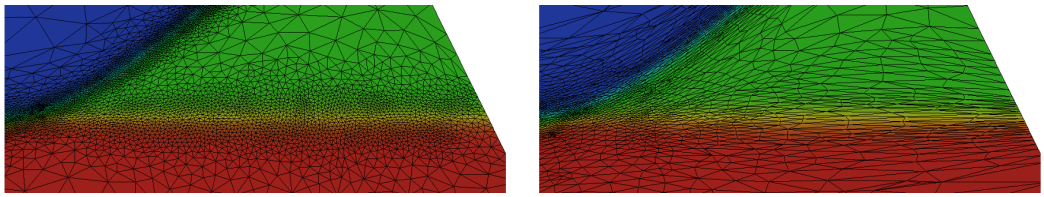


Figure 3.54: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

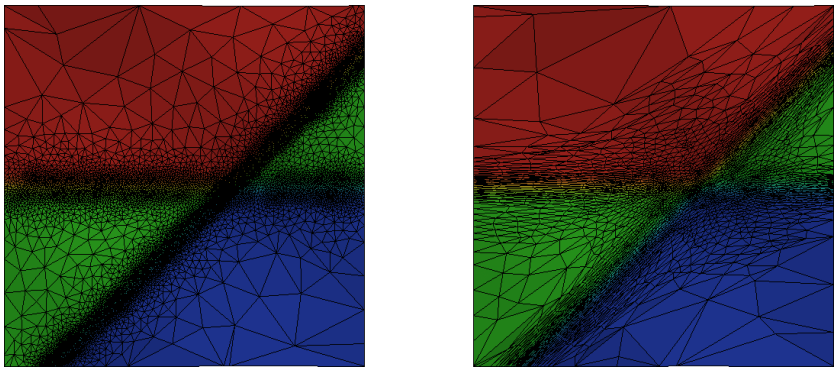


Figure 3.55: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

In Table 3.20 and 3.21 we furnish a more quantitative analysis of the mesh adaptation procedures. Via the coefficient $s_{\max}$, we realize that the triangles in the anisotropic mesh are really distorted. Moreover, these tables confirm also the better "error-vs-number of elements" behaviour of the anisotropic meshes.

	Ele.	η_*	$\lvert u^E - u_h \rvert_{H^1(\Gamma_h)}$	$s_{\max}$	τ
iso	5057	1.402e+00	7.140e-01	6.187e+00	5.000e-01
ani	2206	9.467e-01	7.195e-01	1.780e+01	5.000e-01

Table 3.20: Comparison between the isotropic and anisotropic mesh adaptation for about the same global accuracy.

	Ele.	η_*	$\lvert u^E - u_h \rvert_{H^1(\Gamma_h)}$	$s_{\max}$	τ
iso	5057	1.402e+00	7.140e-01	6.187e+00	5.000e-01
ani	5035	4.147e-01	4.112e-01	2.439e+01	2.200e-01

Table 3.21: Comparison between the isotropic and anisotropic mesh adaptation for about the same number of elements.

Example 14

Let us consider the function

$$u(x, y, z) = \tanh(5x) - \tanh(5(x - y) - 2.5), \tag{3.125}$$

defined on the surface Γ_5 given by the zero level set of the the function

$$d_8 : [-2, 2] \times [-2, 2] \times [-0.4, 0.4] \to \mathbb{R},$$

such as

$$d_8(x, y, z) := 0.4 \sin(0.25\pi x) \sin(0.25\pi y) - z. \tag{3.126}$$

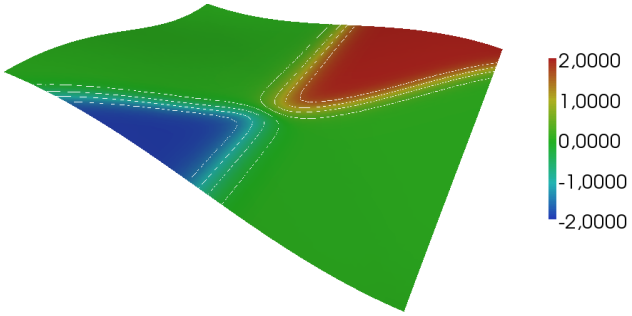


Figure 3.56: The exact function u of example 14, the iso-lines highlight the presence of the boundary layer.

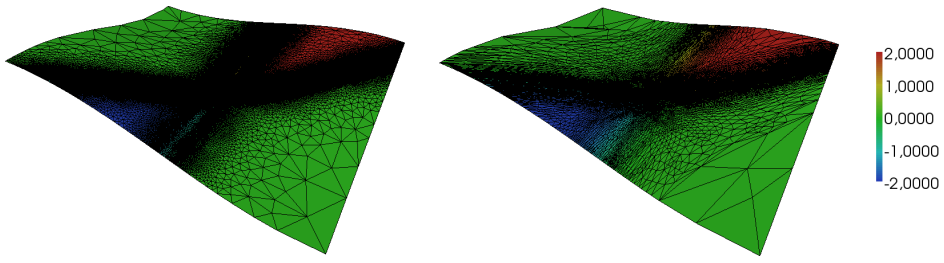


Figure 3.57: Isotropic adapted mesh, on the left, and anisotropic adapted mesh, on the right.

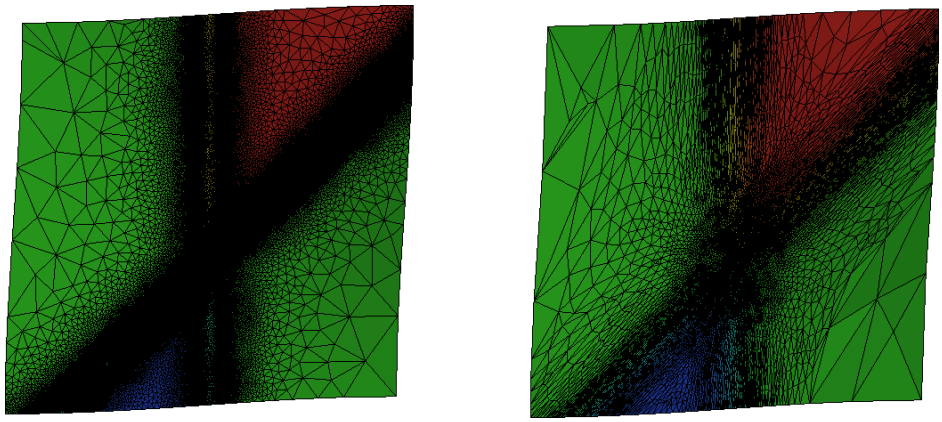


Figure 3.58: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

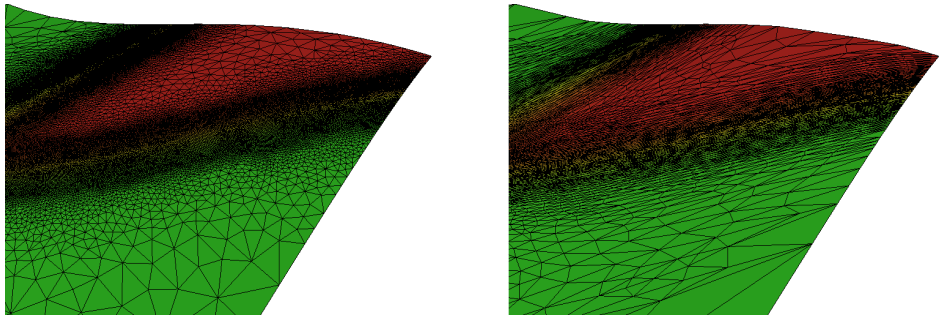


Figure 3.59: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

In Figure 3.56 we provide the exact solution while Figure 3.57 shows the corresponding isotropic and anisotropic adapted mesh. From this figure and the details in Figure 3.58 and 3.59, we see that in the anisotropic mesh the triangles are stretched in the correct direction, in order to capture the layers as well as possible.

Finally, from Table 3.22 and 3.23, we numerically verify that the anisotropic adapted meshes offer better results than the isotropic ones. In fact, if we fix about the same number of elements, we get lower error in the anisotropic case, Table 3.23. Vice-versa, we get the same error with fewer elements when we resort to the anisotropic adapted mesh, see Table 3.22.

	Ele.	η_*	$\lvert u^E - u_h\rvert_{H^1(\Gamma_h)}$	$s_{\max}$	τ
iso	4876	1.413e+00	7.157e-01	5.455e+00	1.200e+00
ani	2742	1.177e+00	7.578e-01	2.040e+01	1.300e+00

Table 3.22: Comparison between the isotropic and anisotropic mesh adaptation for about the same global accuracy.

	Ele.	η_*	$\lvert u^E - u_h\rvert_{H^1(\Gamma_h)}$	$s_{\max}$	τ
iso	4876	1.413e+00	7.157e-01	5.455e+00	1.200e+00
ani	4882	5.713e-01	4.387e-01	3.320e+01	7.100e-01

Table 3.23: Comparison between the isotropic and anisotropic mesh adaptation for about the same number of elements.

Example 15

In this example we consider the function

$$u(x, y, z) = \tanh(20x) + \tanh(20(x - y)) + \tanh(20y) + 2\,, \tag{3.127}$$

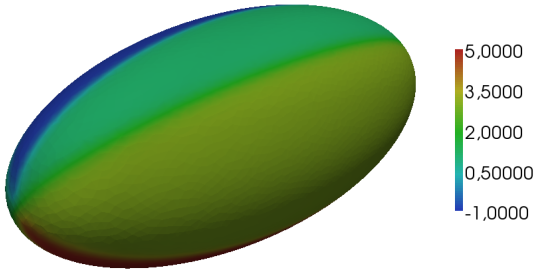


Figure 3.60: The exact function u of example 15.

defined on the surface Γ_6. This surface does not have the boundary and it is defined via the zero level set of the following function

$$d_8 \,:\, [-0.8, 0.8] \,\times\, [-0.8, 0.8] \,\times\, [-2., 2.] \to\, \mathbb{R}\,,$$

such as

$$d_8(x,\, y,\, z) := \frac{x^2}{0.8^2} + \frac{y^2}{0.8^2} + \frac{z^2}{2.0^2}\,. \tag{3.128}$$

As shown in Figure 3.60, u presents a series of internal layers. In Figure 3.60, 3.62 and 3.63 we furnish the whole adapted meshes and some details. In the anisotropic adapted mesh the triangles are properly aligned with the internal layers. Their shape, size and orientation meet the trend of the function u. A more quantitative analysis is given in Table 3.24 and 3.25, where we numerically show the better "error-vs-number of elements" behaviour of the anisotropic meshes.

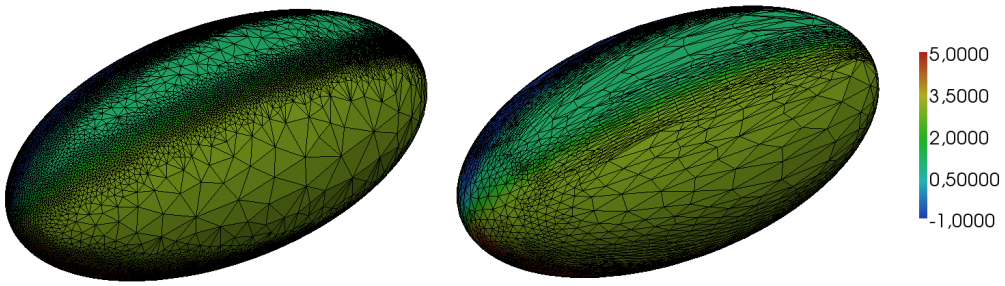


Figure 3.61: Isotropic adapted mesh, on the left, and anisotropic adapted mesh, on the right.

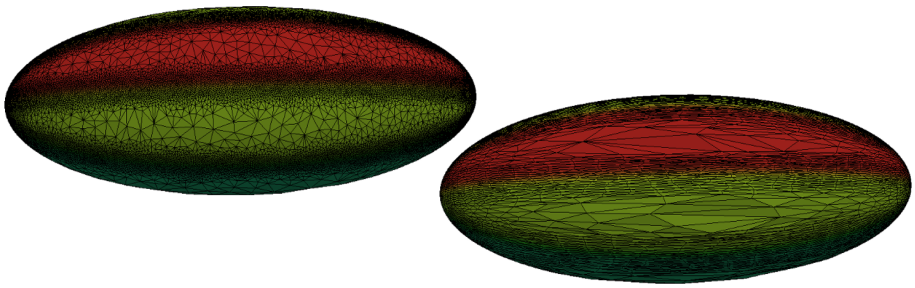


Figure 3.62: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

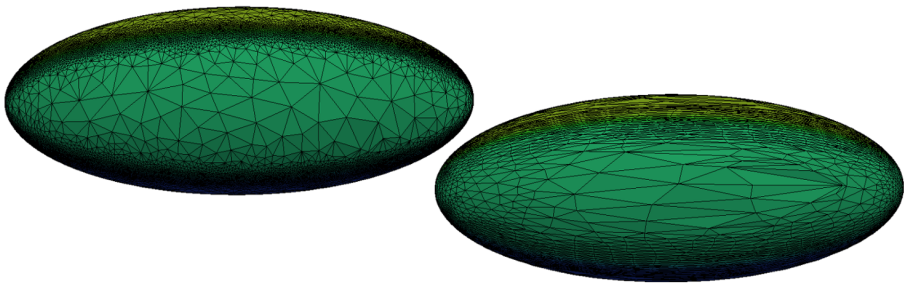


Figure 3.63: Details of the isotropic adapted mesh, on the left, and the anisotropic adapted mesh, on the right.

	Ele.	η_*	$\lvert u^E - u_h \rvert_{H^1(\Gamma_h)}$	$s_{\max}$	τ
iso	13190	1.313e+01	6.161e+00	2.924e+00	3.000e+00
ani	4666	8.515e+00	6.041e+00	2.675e+01	2.700e+00

Table 3.24: Comparison between the isotropic and anisotropic mesh adaptation for about the same global accuracy.

	Ele.	η_*	$\lvert u^E - u_h \rvert_{H^1(\Gamma_h)}$	$s_{\max}$	τ
iso	4834	2.105e+01	1.127e+01	2.281e+00	2.000e+00
ani	4666	8.515e+00	6.041e+00	2.675e+01	2.700e+00

Table 3.25: Comparison between the isotropic and anisotropic mesh adaptation for about the same number of elements.

Chapter 4

A Higher Dimensional Re-Meshing Algorithm

Many computational applications involve triangulation of a complex surface geometry. The main challenge is to automatically generate a surface mesh which satisfies various criteria with respect to the geometry approximation, such as the mesh size and the mesh quality.

In this chapter we propose a novel method to generate a surface mesh which well approximates a surface of interest and with a number of elements as small as possible. For this purpose, we aim at modifying an initial mesh in order to get a new mesh whose elements are adapted according to the curvature of the surface at hand. Intuitively, the most curved regions of the surface will contain small elements and a dense vertex sampling, while the almost flat regions will have large elements with more sparse vertices. Using only isotropic elements may be far from optimal.

However, an anisotropic mesh could offer a better "number of elements vs geometry fitting" behaviour. Figure 4.1 shows an example of an anisotropic mesh.

We propose a new method for re-meshing 3d surfaces based on the idea of an higher dimensional embedding [14, 71, 75]. The idea behind this method is the following: an anisotropic mesh corresponds to an isotropic mesh in a higher dimension. For instance, a surface in $\mathbb{R}^3$ will correspond to an anisotropic mesh in the flat two-dimensional space, see Figure 4.2. In [75], B. Lévy and N. Bonneel proposed a mesh generation strategy based on the computation of the Voronoi diagram in $\mathbb{R}^6$. The results provided in [75] are really impressive, but their algorithm does not preserve sharp features. To overcome this issue, we propose a new method that directly optimizes the triangular surface mesh in the embedded space

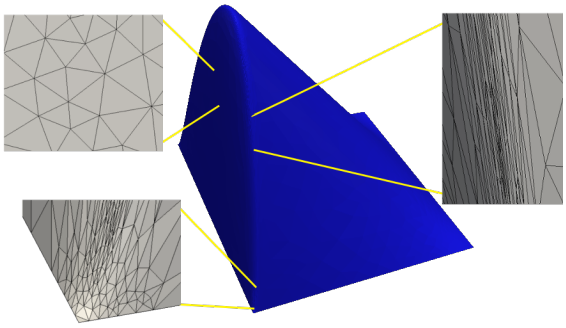


Figure 4.1: The best approximation of a surface, shown in the middle, consists of mesh elements of different size, shape and orientation, matching the principle curvatures of the surface itself.

in such a way that the triangles are as uniform as possible in $\mathbb{R}^6$.

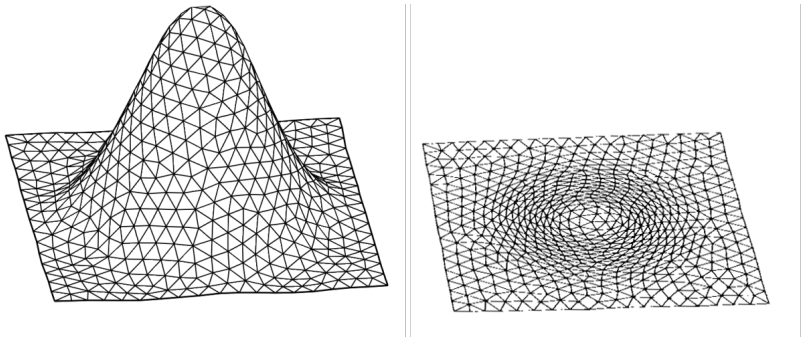


Figure 4.2: On the right the anisotropic mesh in $\mathbb{R}^2$, on the left the corresponding mesh in $\mathbb{R}^3$.

This method has the following properties:

i) the core operation is a uniform re-meshing of a surface; it fits the well-developed mesh adaptation strategies and it is possible to use any optimization-based isotropic surface re-meshing method;

ii) it can handle arbitrary complex geometries and topologies, as well as a very strong anisotropy;

iii) despite the theory proposed in [75], it automatically preserves sharp features, corner and edge singularities;

iv) it is robust; in fact, the initial mesh can provide a very coarse or crude approximation of the actual surface.

4.1 Surface Embedding in $\mathbb{R}^6$

The re-meshing method proposed in this chapter is inspired by the method of B. Lévy and N. Bonneel [75]. The basic idea is pioneered by G. D. Cañas and S. J. Gortler and Y. K. Lai et al. [14, 71] and it is originated in the application of feature characterization [90] from image processing [65].

In the proposed method we get an anisotropic mesh by increasing the dimensions such that we could consider the mesh isotropic in a higher dimensional space, see Figure 4.2:

an isotropic mesh in a higher dimensional space will correspond to an anisotropic mesh in the lower dimensional space.

This concept has been successfully applied in generating curvature-adapted surface meshes [66, 75].

For a smooth surface, it is natural to consider the unit normals defined at the surface points as the components of the co-dimension, i.e., to use the components of the Gaussian map of the surface, [49].

Given a surface Γ in $\mathbb{R}^3$, one can embed it into $\mathbb{R}^6$ by using the one-to-one embedding function: $\Phi : \Gamma \to \mathbb{R}^6$ defined as

$$\Phi(\mathbf{x}) = \begin{bmatrix} x \\ y \\ z \\ s\,n_x \\ s\,n_y \\ s\,n_z \end{bmatrix}, \tag{4.1}$$

where $(n_x, n_y, n_z)^t$ denotes the unit outward normal to Γ at $\mathbf{x}$, and $s \in [0, +\infty)$ is a user-specified parameter.

Remark 4.1.1 *The embedding Φ allows us to approximate the geodesic edge lengths in Γ by the Euclidean edge lengths in $\Phi(\Gamma)$. In fact, each edge length in $\Phi(\Gamma)$ is determined by two parts:*

- *its Euclidean length in $\mathbb{R}^3$;*
- *the variation of the normals at its endpoints, scaled by the parameter s.*

Remark 4.1.2 *Via the embedding Φ, the length of the edges remains the same in the embedded space on flat regions. While, in regions which have high curvatures, the length of edges in $\Phi(\Gamma)$ is much larger than the corresponding length in $\mathbb{R}^3$, see Figure 4.3.*

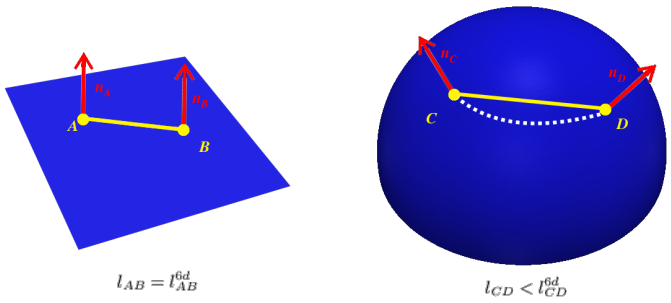


Figure 4.3: Different 6d-lengths with different type of surfaces: on the left a plane where the 6d and the 3d-lengths coincide, $l^{6d}_{AB} = l_{AB}$; on the right a sphere where the two lengths do not coincide, $l^{6d}_{CD} > l_{CD}$.

As it is pointed out in [71], the $\mathbb{R}^6$ embedding Φ is mainly suited for a simple transfer from non-isotropic to isotropic configurations. As will be clear from the developments below, we can still explain everything in $\mathbb{R}^3$ via an appropriate combined processing of points and normals. Indeed, the use of the embedding in $\mathbb{R}^6$ does not result in any computational overhead over working in 3d.

By embedding a surface in a higher dimensional space motivates the new problem: *how to generate an isotropic good quality surface mesh in this embedding space?* In principle, a direct generalization of available methods in 3d is possible, but this will be impractical due to the $d!$ memory cost requirement.

B. Lévy and N. Bonneel [75] overcome this difficulty by using their *Vorpaline* (Voronoi Parallel Linear Enumeration) technique to compute a restricted centroidal Voronoi diagram (CVT) embedded in 6d. It directly computes the 6d

Voronoi cells by iterative half-space clipping. It requires only the nearest neighbour information for a point set in $\mathbb{R}^6$.

It is proved by B. Lévy and N. Bonneel [75] that this method may produce flipped (self-intersected) triangles, in particular in regions where the anisotropy varies too fast. This fact implies that, if the normals between the neighbour vertices vary too quickly, this method may not work correctly. One possible way to get end of this problem is to insert new vertices between neighbouring vertices. However, the method by B. Lévy and N. Bonneel [75] does not support inserting new vertices dynamically.

Another well-known problem in RVD- and CVT-based methods is that sharp features or details of the surfaces may be smoothed or missed in the resulting mesh, see the examples in [75]. Although a theoretical solution has been proposed in [16], its efficiency is still a challenge in practice.

Due to these problems, we propose a new method in the next section. In particular, we show that a common mesh optimization framework for isotropic surface re-meshing may be applied in re-meshing surfaces embedded in a higher dimensional space.

4.2 The Re-meshing Approach

Consider a surface Γ in $\mathbb{R}^3$. For simplicity, we assume that Γ is a smooth surface, i.e., it contains no corner and edge singularities. Then, we will consider the non-smooth case. In this section, we propose an optimization-based method for re-meshing Γ.

4.2.1 Preliminaries

The proposed method assumes that the following two functions, η_1 and η_2, are provided:

(1) given a point $\mathbf{p} \in \Gamma$, η_1 returns the normal to Γ at $\mathbf{p}$;

(2) given a point $\mathbf{p} \in \mathbb{R}^3$, η_2 returns the closest point on Γ.

If Γ is represented by an implicit function or it is a parametrized surface, e.g., a CAD model, the exact normals and the closest points of Γ are provided. In the case that Γ is given as a polygonal mesh, these two functions must be approximated from the input data, see, e.g., [46, 3, 13, 87].

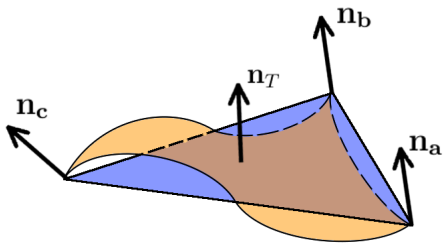


Figure 4.4: An example of inverted triangle.

Moreover, we assume to have an initial surface mesh of Γ, Γ_{init}. It is required that all triangles of this initial mesh Γ_{init} are oriented consistently. In particular, if $\mathbf{ab}$ is the common edge of two faces $\mathbf{abc}$ and $\mathbf{bad}$, the normals to $\mathbf{abc}$ and $\mathbf{bad}$ are pointing outwards to the surface.

Moreover, if a triangle is used to approximate a patch of a surface, the normal of the triangle, $\mathbf{n}_T$, should not vary too much with respect to the normals of the real surface, $\mathbf{n_a}$, $\mathbf{n_b}$ and $\mathbf{n_c}$, see Figure 4.4. We use a heuristic condition to justify the geometric approximation.

Definition 4.2.1 *Let T be a face in the surface mesh. We say that T is* **inverted** *if the maximum angle between the normals to Γ at each triangle vertex, $\mathbf{n_a}$, $\mathbf{n_b}$ and $\mathbf{n_c}$, and the normal to the triangle, $\mathbf{n}_T$, is greater than a given threshold.*

An inverted face is considered a bad approximation of the geometry. Figure 4.4 shows an example of inverted triangle. Thus, it is crucial to use an appropriate threshold in order to achieve the best mesh quality. In our experiments we fix the threshold equal to $90°$.

4.2.2 The parameter s

We found it is better to analyse more in the details the role of the parameter s in Φ. We consider a surface Γ and two points $A, B \in \Gamma$, we apply the map Φ and we have:

$$\begin{aligned} \Phi(A) &= (x_A,\, y_A,\, z_A,\, sn_A,\, sv_A,\, sw_A)^t\,, \\ \Phi(B) &= (x_B,\, y_B,\, z_B,\, sn_B,\, sv_B,\, sw_B)^t\,, \end{aligned}$$

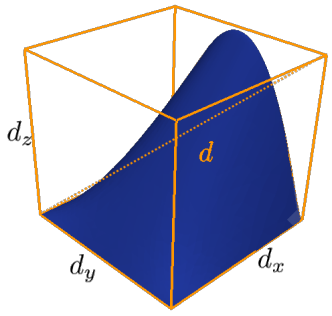


Figure 4.5: Bounding box of the surface Γ.

where x_A, y_A, z_A and x_B, y_B, z_B are the $\mathbb{R}^3$ coordinates of A and B and n_A, v_A, w_A and n_B, v_B, w_B are the components of the normal vector to Γ at A and B. The scalar product in $\mathbb{R}^6$ between these two points is

$$(A,B)_{6d} = \underbrace{x_A x_B + y_A y_B + z_A z_B}_{I} + s^2 \big(\underbrace{n_A n_B + v_A v_B + w_A w_B}_{II} \big) .$$

Since the coordinates of both A and B vary in the bounding box of the surface Γ, we can say that $I \in [-d^2, d^2]$ where d is the diagonal of the bounding box, see Figure 4.5. Moreover, we can state that the quantity $II \in [-1, 1]$, because the normals at A and B are such that $n_A^2 + v_A^2 + w_A^2 = 1$ and $n_B^2 + v_B^2 + w_B^2 = 1$. Indeed, the parameter s is introduced to give more or less importance to the normals, II, on the whole value of $(A,B)_{6d}$. But, since $I \in [-d^2, d^2]$ and $II \in [-1, 1]$, the contribution of I and II on $(A,B)_{6d}$ is unbalanced, because it depends on the dimension of the bounding box.

To make the quantities I and II almost comparable, we modify the scalar product in $\mathbb{R}^6$ in such a way:

$$(A,B)_{6d} = x_A x_B + y_A y_B + z_A z_B + (h_\Gamma s)^2 \big(n_A n_B + v_A v_B + w_A w_B \big) , \quad (4.2)$$

where

$$h_\Gamma = \frac{d_x + d_y + d_z}{3} ,$$

and d_x, d_y and d_z are the dimensions of the bounding box of Γ, see Figure 4.5. In this way, the quantity I and II are at most comparable and the parameter s has the effect to give more or less importance to the normals. In the following part of the

chapter, we use the scalar product defined in Equation (4.2) and we numerically analyse the influence of s in the re-meshing procedure, see Subsection 4.3.1.

4.2.3 Overview of the Approach

The inputs of the algorithm are:

1. an initial triangular mesh Γ_{init} of the surface Γ;
2. a user-specified edge length L_{des}, in $\mathbb{R}^6$;
3. a user-specified minimum face angle ϑ_{min};
4. a parameter s that specifies the desired amount of anisotropy.

We initialize a mesh $\Gamma_h := \Gamma_{\text{init}}$. Then, we use the map Φ to transform Γ_h into a surface mesh $\Gamma_{h,\Phi}$ in $\mathbb{R}^6$.

The proposed method directly optimizes a two-dimensional triangular mesh of the surface embedded in $\mathbb{R}^6$, so way that its triangles are as uniform as possible in the embedded space $\mathbb{R}^6$. During this process, we still work in $\mathbb{R}^3$, **but** we measure the edge length and the angle in $\mathbb{R}^6$. Consequently, once we map back the triangular surface mesh in $\mathbb{R}^3$, we get an anisotropic mesh, where the triangular elements are stretched according to the curvature of the surface Γ.

In the proposed re-meshing method we may distinguish two main phases:

i) *sampling*, we split the edges in Γ_h whose lengths, measured in $\mathbb{R}^6$, are too long or we contract the edges that are too short with respect to the given parameter L_{des}, Subsection 4.2.5;

ii) *optimizing*, we maximize the smallest face angle such that they are not smaller than ϑ_{min}, measured in $\mathbb{R}^6$, Section 4.2.6.

The result of this adaptation procedure is a curvature-adapted anisotropic triangular mesh of the surface Γ. The following subsections describe how we have modified the standard local mesh operations, in Section 1.2 to tackle this new framework.

A sketch of the whole embedding procedure is provided in Figure 4.6.

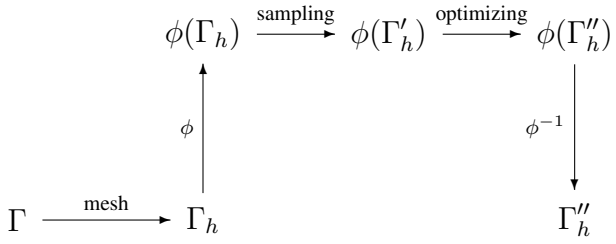


Figure 4.6: Scheme of the re-meshing process.

4.2.4 Local Mesh Modifications

The proposed algorithm applies a series of local surface mesh modifications directly to the mesh Γ_h. The most well-known and commonly used local modifications are: edge-flip, edge-collapse, vertex insertion, and vertex smoothing.

An Edge-flip Algorithm

Given an edge $\mathbf{ab}$ in Γ_h, it needs to be flipped if one of the following two conditions are met:

(a) (geometric approximation) either face $\mathbf{abc}$ or face $\mathbf{bad}$ is inverted;

(b) (mesh quality) both $\mathbf{abc}$ and $\mathbf{bad}$ are not inverted and the smallest 6d-angle of the two new faces ($\mathbf{cdb}$ and $\mathbf{dca}$) is larger than the smallest 6d-angle of $\mathbf{abc}$ and $\mathbf{bad}$.

If an edge $\mathbf{ab}$ satisfies either (a) or (b), it implies that either face $\mathbf{abc}$ or face $\mathbf{bad}$ is bad, or both of them. In these cases, we have to locally improve the mesh. However, as we have seen in Subsection 1.2.1, since we are dealing with surface mesh, the edge $\mathbf{ab}$ is flippable if it meets the following conditions:

(i) the edge **cd** does not belong to the mesh;

(ii) any of the angles adjacent to the edge **ab** has to be obtuse;

(iii) the angle between the normals to triangles **abc** and **bad** has to be lower than a threshold value, in this work we take $15°$.

Then the edge-flip algorithm we propose for this kind of re-meshing procedure has a structure similar to the FLIPEDGES procedure shown in Algorithm 1.

Edge Collapse

Edge collapse is a common operation for simplifying meshes. Consider an edge $\mathbf{ab}$, this operation unifies the two endpoints of the edge, $\mathbf{a}$ and $\mathbf{b}$, so that two adjacent faces vanish, $\mathbf{abx}$ and $\mathbf{bay}$. The endpoints may be contracted in one of them or in a new vertex inside the cavity of the faces adjacent to the edge $\mathbf{ab}$.

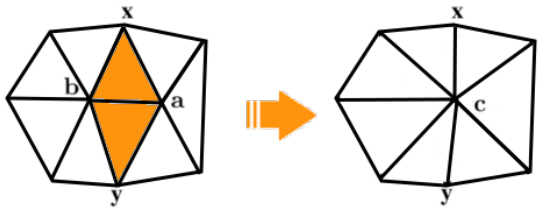


Figure 4.7: Collapsing of the edge $\mathbf{ab}$.

Since we are dealing with a surface re-meshing, if we decide to unify the endpoints of the edge into a new vertex inside this cavity $\mathbf{xayb}$, we have to project the new point on the real surface Γ and evaluate its normal to the surface. We avoid this time consuming computation, by contracting the edge $\mathbf{ab}$ into one of its endpoints. More in details, we choose the endpoint that modifies the mesh in order to have the smallest angle as large as possible.

After a contraction is performed, we push all the edges connected to this new vertex into a stack, and then the routine FLIPEDGES runs, to locally improve the mesh.

Vertex insertion

Let $\mathbf{v}$ be a new vertex in the surface, to be inserted on the edge $\mathbf{ab}$. The splitting of this edge replaces two faces $\mathbf{abc}, \mathbf{bad}$ by four faces $\mathbf{avc}, \mathbf{vbc}, \mathbf{bvd}, \mathbf{vad}$, see Figure 4.8.

Since the surface may be curved, the middle point of the edge $\mathbf{ab}$ is not necessary on the surface; so the resulting mesh does not fit the real surface. We avoid this problem by projecting this new inserted point onto the real surface.

After this insertion takes place, we put all the edges connected to $\mathbf{v}$ into a stack and run the routine FLIPEDGES to locally improve the mesh.

However, the insertion of a new vertex on a 3d surface mesh may considerably deform the surface itself. In fact, when the initial surface mesh is only a crude

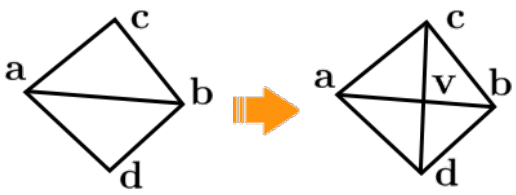


Figure 4.8: Vertex insertion.

approximation of the original geometry, the insertion of a new vertex may create inverted elements, we show an example in Figure 4.9.

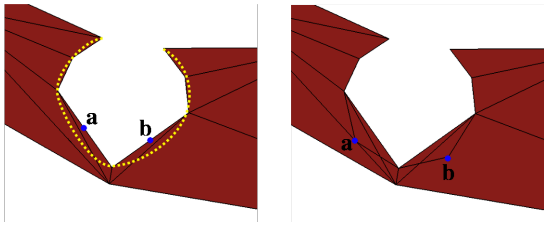


Figure 4.9: On the left the middle points of the edges that we want to add and the real geometry is dashed. On the right the points, a and b, are projected on the real geometry but this yields an undesired configuration.

To correct this undesired features, we use the FLIPEDGES routine. In fact, in addition to improving the neighbourhood of the vertex v, the routine FLIPEDGES can **automatically** fix this kind of problem. Numerical experiments show that this edge flip algorithm is very effective to overcome this issue, but, unfortunately, its convergence is not mathematically proved yet.

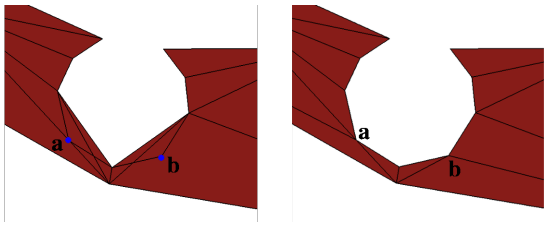


Figure 4.10: On the left the undesired configuration, on the right the configuration is fixed.

Vertex smoothing

For a given vertex $\mathbf{v}$, the smoothing operation consists in finding a new location for this vertex such that the local mesh quality is improved **without** changing the mesh topology.

A generic smoothing method moves a point $\mathbf{v}$ in a new location $\mathbf{v}'$ given by the formula

$$\mathbf{v}' = \mathbf{v} + \alpha \sum_{\mathbf{v}_i \in \omega_\mathbf{v}} f(d(\mathbf{v}\,, \mathbf{v}_i))\mathbf{u}_i\,, \tag{4.3}$$

where α is a constant, $f : \mathbb{R} \to \mathbb{R}$ is a function, $\omega_\mathbf{v}$ is the set of vertices that are connected to $\mathbf{v}$ and $\mathbf{u}_i$ are the unit vectors that identify the direction from $\mathbf{v}_i$ to $\mathbf{v}$, see Figure 4.11. Finally, $d(\mathbf{v},\, \mathbf{v}_i)$ is the distance between $\mathbf{v}$ and $\mathbf{v}_i$, normalized with respect to a desired edge length of the mesh, i.e.,

$$d(\mathbf{v},\, \mathbf{v}_i) = ||\mathbf{v} - \mathbf{v}_i||/L_{\text{des}}\,,$$

where $||\cdot||$ is the standard Euclidean norm in $\mathbb{R}^3$ and L_{des} is the desired edge length.

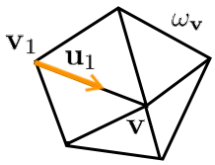


Figure 4.11: Patch $\omega_\mathbf{v}$, where we highlight the direction $\mathbf{u}_1$, identified by the couple of vectors $\mathbf{v}$ and $\mathbf{v}_1$.

Different smoothing methods are characterized by different choices of the parameter α and of the function f in Equation (4.3). For instance, the classical Laplacian smoothing, [37], is defined by

$$\alpha = \frac{1}{\#\omega_\mathbf{v}} \quad \text{and} \quad f(d) = -d\,.$$

where $\#\omega_\mathbf{v}$ is the cardinality of the set $\omega_\mathbf{v}$, i.e., the number of vertices connected to $\mathbf{v}$.

Remark 4.2.1 *According to the sign of the function f, the vertex $\boldsymbol{v}$ is attracted or repelled by the vertex $\boldsymbol{v}_i$, i.e.,*

- *if $f(d(\boldsymbol{v}, \boldsymbol{v}_i)) > 0$, $\boldsymbol{v}$ is repelled by $\boldsymbol{v}_i$;*
- *if $f(d(\boldsymbol{v}, \boldsymbol{v}_i)) < 0$ it is attracted by $\boldsymbol{v}_i$.*

A standard Laplacian smoothing does not use this feature on the function f, in fact $f(d) = -d \geq 0$. However in [10], F. J. Bossen and P. S. Heckbert implemented a vertex smoothing method that exploits this attraction/repulsion behaviour via the function

$$f_{\text{BH}}(d) := (1 - d^4)e^{-d^4}.$$

Since their mesh generation procedure is metric-based, their desired edge length is set to 1. Thus, if the vertex $\mathbf{v}$ is too close to $\mathbf{v}_i$, i.e., $0 < d < 1$, it is repelled by $\mathbf{v}_i$; vice-versa, if it is far from $\mathbf{v}_i$, i.e., $1 < d < 1.7$, it is attracted. Finally, if it is too far, $d \geq 1.7$, or if it is exactly at the right distance, $d = 1$, $\mathbf{v}_i$ does **not** influence the new position of $\mathbf{v}$. The trend of $f_{\text{BH}}(d)$ is shown in Figure 4.12.

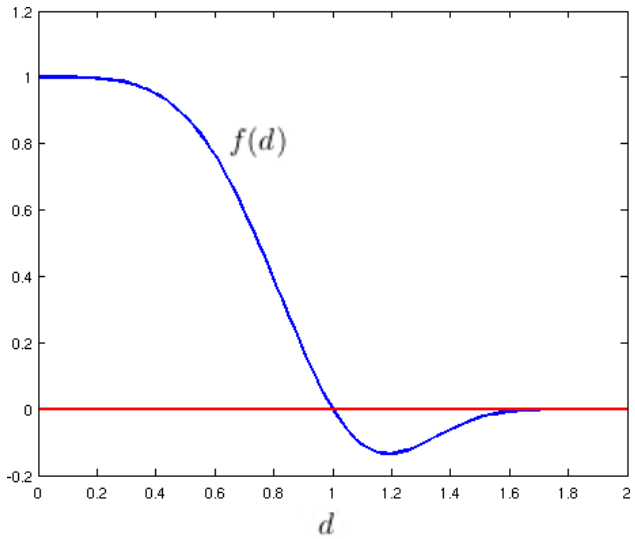


Figure 4.12: Smoothing function $f_{\text{BH}}(d) = (1 - d^4)e^{-d^4}$ proposed by Bossen and Heckbert.

In this re-meshing procedure we use a vertex smoothing method inspired by the one proposed by F. J. Bossen and P. S. Heckbert in [10].

Consider the set of vertices connected to $\mathbf{v}$, $\omega_{\mathbf{v}} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_n\}$. Since we are trying to make the mesh as uniform as possible in the embedded space, the best location of the vertex verifies this property:

$$||\mathbf{v} - \mathbf{v}_i||_{6d} = L_{\text{des}} \quad \forall i = 1, 2, \ldots n\,.$$

To move the point in this optimal location, we use the following normalized distance function

$$d_{6d}(\mathbf{v}, \mathbf{v}_i) = \frac{||\mathbf{v} - \mathbf{v}_i||_{6d}}{L_{\text{mean}}}\,,$$

where L_{mean} is the mean 6d-length of the edges $\mathbf{v}\mathbf{v}_i$ in the patch $\omega_{\mathbf{v}}$. Finally, the smoothing formula we propose is:

$$\mathbf{v}' = \mathbf{v} + \alpha \frac{\mathbf{w}}{||\mathbf{w}||}\,, \tag{4.4}$$

where

$$\mathbf{w} = \sum_{\mathbf{v}_i \in \omega_{\mathbf{v}}} f_{\text{BH}}\left(d_{6d}(\mathbf{v}, \mathbf{v}_i)\right) \mathbf{u}_i\,,$$

and the parameter α is chosen as

$$\alpha := \max\left\{c_1 L_{\omega_{\mathbf{v}}}, c_2 L_{\text{min}}\right\},$$

where L_{min} is the minimum valid edge length of the mesh,

$$L_{\omega_{\mathbf{v}}} := \min_{\mathbf{v}_i \in \omega_{\mathbf{v}}} ||\mathbf{v} - \mathbf{v}_i||\,,$$

while c_1 and c_2 are constants, in this work we have chosen $c_1 = 0.01$ and $c_2 = 10$.

After finding the location $\mathbf{v}'$, we project the point on the surface Γ, we push all the edges connected to $\mathbf{v}$ into a stack and we run the routine FLIPEDGES to locally improve the quality of the mesh.

The vertex smoothing plays a key role in the proposed surface re-meshing method. In fact, it is called in the most internal loop of the optimization algorithm, line 4 in Algorithm 4. Due to this wide employment, it has to be sufficiently fast. To increase the speed of this process, we decide to move not **all** the vertices of the mesh, but only the ones that are more far away from their optimal position. The magnitude of $\mathbf{w}$ suggests us which vertex has to be moved. In fact, we may infer that vertices associated with high magnitude of $\mathbf{w}$ are far from their optimal position, on the contrary, vertices with low values of $||\mathbf{w}||$ are close to their optimal position.

4.2.5 Sampling

The purpose of sampling is to achieve the desired mesh size with respect to the given 6d-length parameter L_{des}. Our strategy is straightforward, splitting and contracting the edges that are too long or too short compared to the desired length.

Algorithm 3 The sampling algorithm

SAMPLING(Γ, $\mathcal{T}$, Q, L_{des})
Data:Q is a queue of triangles in $\mathcal{T}$.

1: **while** Q is non-empty **do**
2: pop a face f from Q;
3: Let e be the longest edge of f;
4: **if** $\|e\|_{6d} > 1.5\ L_{des}$, **then**
5: split e by adding $v \in \Omega$ into $\mathcal{T}$;
6: update Q;
7: **end if**
8: **end while**
9: put all the triangles in Q;
10: **while** Q is non-empty **do**
11: pop a face f from Q;
12: Let e be the shortest edge of f;
13: **if** $\|e\|_{6d} < 0.5\ L_{des}$, **then**
14: contract e;
15: update Q;
16: **end if**
17: **end while**

The sampling algorithm is shown in Algorithm 3. It initializes a stack Q which contains all the triangles in Γ_h. Then, it works in a loop until Q is empty. On each face T popped from Q, it checks the longest edge e of T and split it if the 6d-length of e is too long, lines $4-7$. Once all the too long edges are splitted, the stack Q is refilled with all the triangles in Γ_h, line 9. Then, it loops until Q is empty; T is popped from Q, the algorithm look for the shortest edge e of T and perform a contraction if the 6d-length of e is too short, lines $13-16$. Then, Q is updated by removing old faces and adding new faces, line 15.

4.2.6 Optimizing

Since the sampling phase has removed too long and too short edges, the goal of the optimizing phase is to maximize the smallest 6d-angle of the triangles in the mesh Γ_h. We follow the sequence of operations reported in Figure 4.

Mesh optimization is performed by iteratively combining a series of local operations:

- edge flipping;
- vertex smoothing that iteratively moves the positions of vertices on surface so that we have a uniform edge length of the edges connected to **v**;
- edge collapsing that is used to remove small angles, i.e,. $\theta < \theta_{\text{min}}$; this operation iteratively removes the edges opposite to small angles;
- edge splitting that is used to remove large angles, i.e., $\theta > 180^{\circ} - 2\theta_{\text{min}}$; this operation iteratively splits the edges opposite to large angles.

We call the routine FLIPEDGES within each on the last three operations to locally improve the mesh quality.

Algorithm 4 The optimizing algorithm

OPTIMIZING(Ω, $\mathcal{T}$, L_{des}, θ_{min}, I, J, K)
Data: I , J and K are integers that specify the number of iterations.

```
1:  θ_max := 180^o − 2 ∗ θ_min;
2:  Collapse too short edges with respect to L_des;
3:  for i ∈ {1, ..., I} do
4:     for j ∈ {1, ..., J} do
5:        for k ∈ {1, ..., K} do
6:           Smooth all vertices;
7:        end for
8:        Collapse edges to remove angles < θ_min;
9:     end for
10:    Split edges to remove angles >= θ_max;
11: end for
```

4.2.7 Sharp Features

This method can be easily adapted to mesh non-smooth surfaces, i.e., surfaces containing edges, corner singularities and sharp features. These features are commonly present in complex geometries, such as CAD geometries.

We consider a surface Γ, for example a geometry coming from a CAD model. The whole geometry is the sum of a finite set of patches that are smoothly joint together along their common boundaries, i.e.,

$$\Gamma = \bigcup_{i=1}^{n} \Gamma_i \,, \quad \text{and} \quad \Gamma_i \cap \Gamma_j = \begin{cases} \emptyset \\ \gamma_{ij} \end{cases} , \tag{4.5}$$

where γ_{ij} is the common line between Γ_i and Γ_j, if it exists, see Figure 4.13.

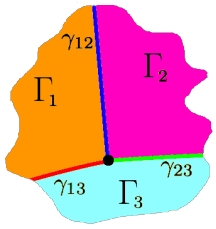


Figure 4.13: Example of a domain with sharp features.

The proposed method re-meshes each sub-surface Γ_i, separately. Then, when we operate on the common boundaries γ_{ij}, the meshes that represent the surfaces Γ_i and Γ_j will be suitably updated.

The presence of sharp features gives a further constraint on the flip algorithm. In fact, if we decide to flip an edge that lies on the common line between two different patches Γ_i and Γ_j, we lose the exact shape of these patches, see Figure 4.14. To preserve sharp features, we add this condition on the flipping of a generic edge $\mathbf{ab}$:

(iv) the edge $\mathbf{ab}$ does not belong to a sharp feature.

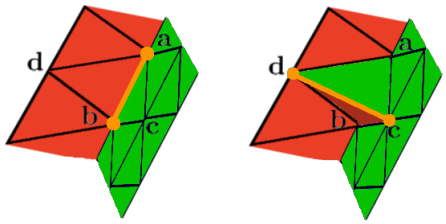


Figure 4.14: Flipping of the edge $\mathbf{ab}$, that belongs to a sharp features.

4.3 Examples

In this section, several examples are presented to demonstrate the reliability of the proposed method and they have been done with a laptop with a 2.26 GHz processor. Firstly, we re-mesh some very simple surfaces to ensure that this method works. In particular, we consider surfaces such that we may predict where and how the triangles will be stretched, and we experimentally verify the right behaviour of the proposed re-meshing method.

Finally, in Example 4, we consider a very simple geometry in order to verify that the proposed re-meshing procedure **does** preserve sharp features.

We report the statistical information of these examples and the CPU times in Table 4.1. To evaluate the level of anisotropy of the resulting mesh, we consider the so-called **aspect ratio**, see Definition 1.1.2:

$$Q(T) := \frac{2r_T}{R_T}\,, \tag{4.6}$$

here T is a generic triangle of the mesh, r_T and R_T are the radii of the inscribed and circumscribed circles, respectively. If $Q(T) \approx 0$, the triangle T is stretched, while, for triangles close to the equilateral one, we have $Q(T) \approx 1$.

From Table 4.1, we observe that the meshes present really stretched elements, in fact the minimum value of the aspect ratio is close to 0. Moreover, we see that in Example 1 and 2 the final anisotropic meshes have fewer elements and nodes than the initial grid. This fact numerically proves the better "number of elements vs geometry fitting" behaviour of anisotropic meshes. In Example 3 and 4, we do not have the same gain, but in both these examples the initial mesh is a very rough approximation of the real surface, so we need a more extensive sampling.

	Examples	1	2	3	4
1.	L_{des}	0.1	0.1	0.1	0.1
2.	# vertices in Γ_h output	11518	49761	12985	2357
3.	# triangles in Γ_h output	23032	98560	25543	4710
4.	# vertices in Γ_h input	3697	20659	4937	5246
5.	# triangles in Γ_h input	7390	40790	9582	10488
6.	Sampling Time (sec.)	7	28	8	2
7.	Optimizing Time (sec.)	20	141	85	49
8.	Minimum Aspect Ratio	3.841e-02	2.383e-01	8.414e-04	3.249e-01

Table 4.1: Statistics of the Examples. The Optimizing time (in row 7) is the time for one outer loop in the optimizing algorithm (Figure 4), i.e., for $I = 1$. The number of inner loops are: $J = K = 4$.

Example 1

We consider the disk, defined by the zero level set of the following function

$$f_1 \,:\, [-0.8, 0.8] \,\times\, [-0.8, 0.8] \,\times\, [-0.4, 0.4] \to \mathbb{R}\,,$$

such as

$$f_1(x\,,y\,,z) := \left(\frac{x}{0.8}\right)^2 + \left(\frac{y}{0.8}\right)^2 + \left(\frac{z}{0.4}\right)^2 - 1\,. \tag{4.7}$$

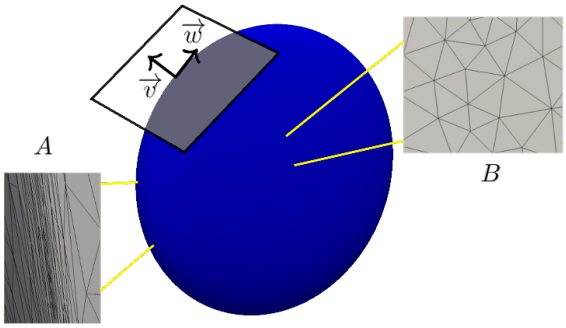


Figure 4.15: The geometry of **Example** 1. The zero level set of the function f_1; some zones are highlighted to show some details of the adapted mesh.

Since there is a very big change of curvature along one direction $\overrightarrow{v}$, see zone A in Figure 4.15, we expect that the triangles to be stretched along $\overrightarrow{w}$, i.e., the perpendicular direction that lies on the tangent plane to the surface. Vice-versa, we expect equilateral triangles in the zone B, where the surface is smoother and flat. The resulting mesh is shown in Figure 4.16: the shape of triangles behaves as expected.

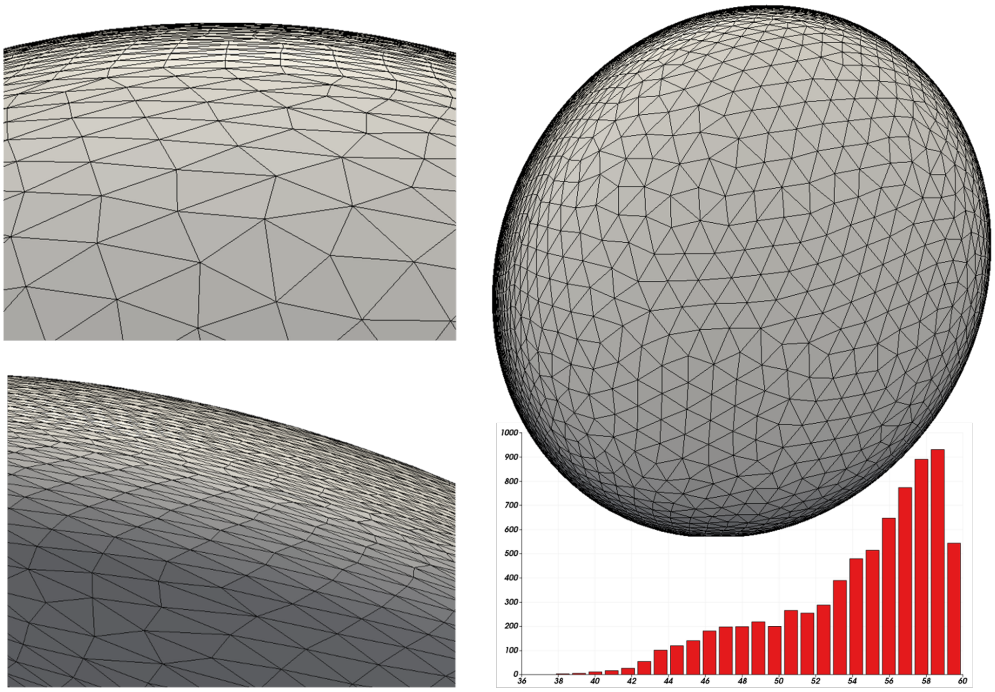


Figure 4.16: The optimized mesh for **Example** 1. The mesh quality (the smallest 6d-angles in triangles) histogram is shown bottom-right.

In the histogram of Figure 4.16 bottom-right, we provide the smallest 6d-angles for each triangle of the mesh. We can appreciate that they are close to the optimal one, $60°$, and that the smallest one is far from the $0°$, in fact the minimum is $38°$.

Example 2

We consider the sinusoidal surface, defined by the zero level set of the following function

$$f_2 \,:\, [-1.,1.] \,\times\, [-1.,1.] \,\times\, [-0.2,0.2] \,\rightarrow\, \mathbb{R}\,,$$

such as

$$f_2(x\,,y\,,z) := 0.2\sin(\pi x)\sin(\pi y) - z\,. \tag{4.8}$$

In Figure 4.17 we show the surface and some zones of interest. In particular, we notice that there are a lot of regions where the mesh should be isotropic, see the zones A and B in Figure 4.17. The resulting mesh is shown in Figure 4.18.

Even in this case we experimentally provide that the proposed re-meshing method behaves as expected. Moreover, from the histogram in Figure 4.18 bottom-right, we recognize that the smallest 6d-angle is close to $60°$.

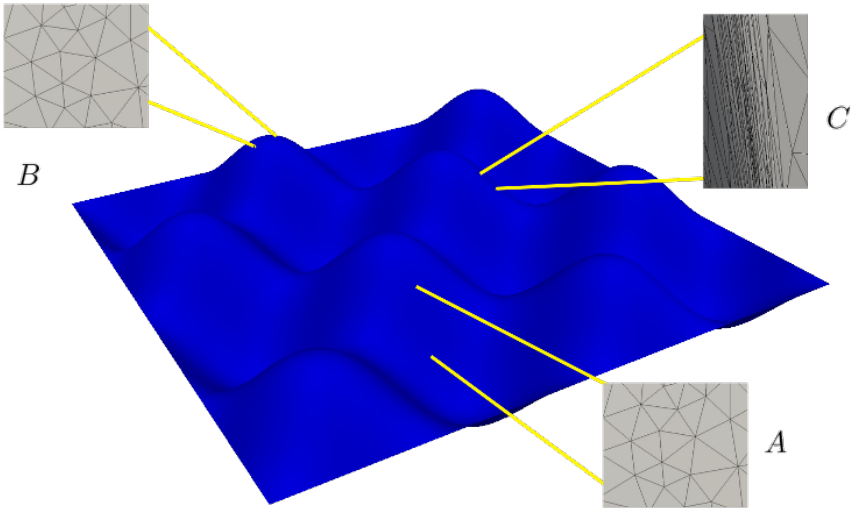


Figure 4.17: The geometry of the **Example** 2. The zero level set of the function f_2; some zones are highlighted to show how the adapted mesh is expected to be.

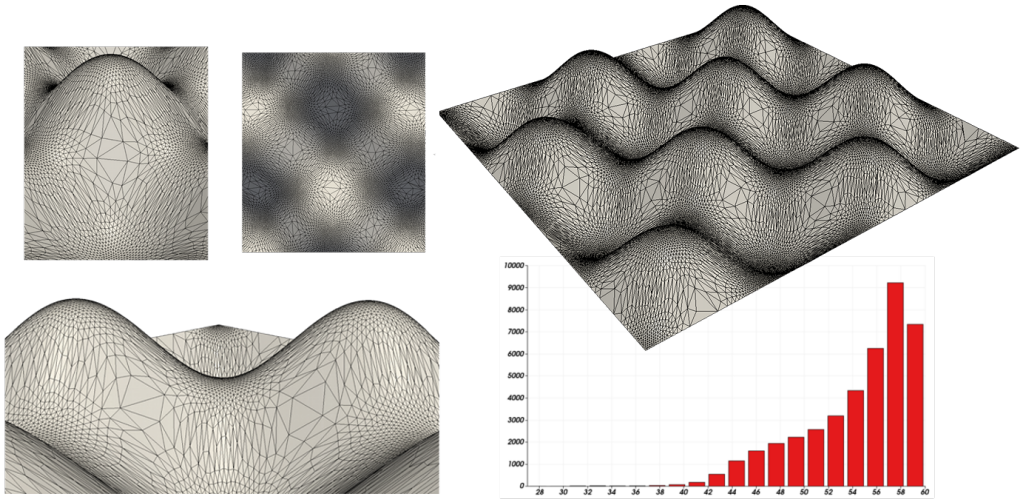


Figure 4.18: The resulting mesh for **Example** 2. The mesh quality (the smallest 6d-angles in triangles) histogram is shown bottom-right.

Example 3

In this example, we consider the function

$$f_3 \,:\, [-1.,1.] \,\times\, [-1.,1.] \,\times\, [-2.,2.] \,\rightarrow\, \mathbb{R}\,,$$

such as

$$f_3(x\,,y\,,z) := \tanh\left(20x\right) - \tanh\left(20(x-y)-10\right) - z\,. \tag{4.9}$$

The zero level set of such a function is a surface that exhibits a smart change of curvature, see zones A and B in Figure 4.19, and it is flat in others, see zones C and D in Figure 4.19. This behaviour allows the re-meshing procedure to create very stretched elements in A and B, and isotropic triangles in C and D.

We start from a very rough mesh obtained by a marching cube procedure, see Figure 4.20: the triangles are not oriented in the right way, there is an over sampling, zones C and D, and the triangles are somewhere really far away form the real geometry.

Figure 4.21 shows the resulting mesh of this example. The geometry is really well fitted by the resulting mesh. In particular, the initial mesh has been entirely

changed. Furthermore, we observe that both the **shape** and **orientation** of the elements help to fit the geometry as well as possible.

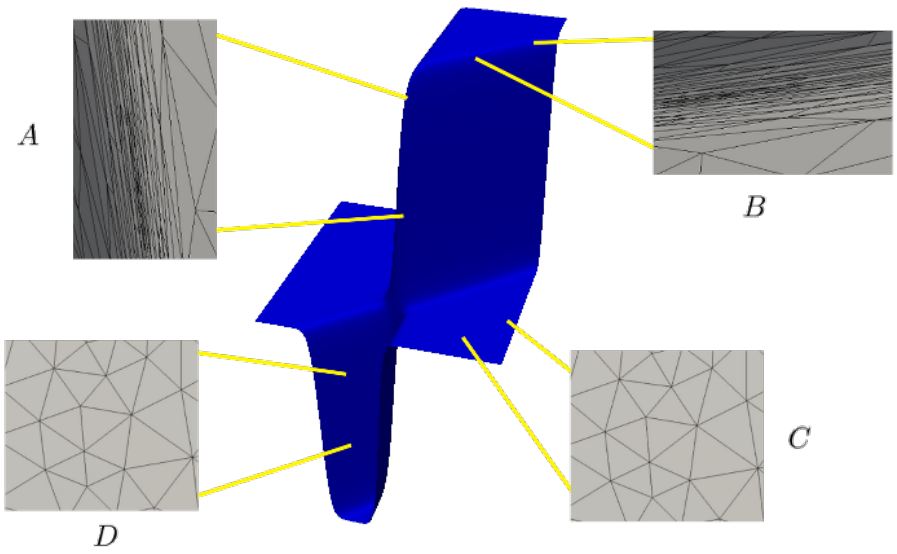


Figure 4.19: The geometry of **Example** 3. The zero level set of the function f_3; some zones are highlighted to show how the adapted mesh will be.

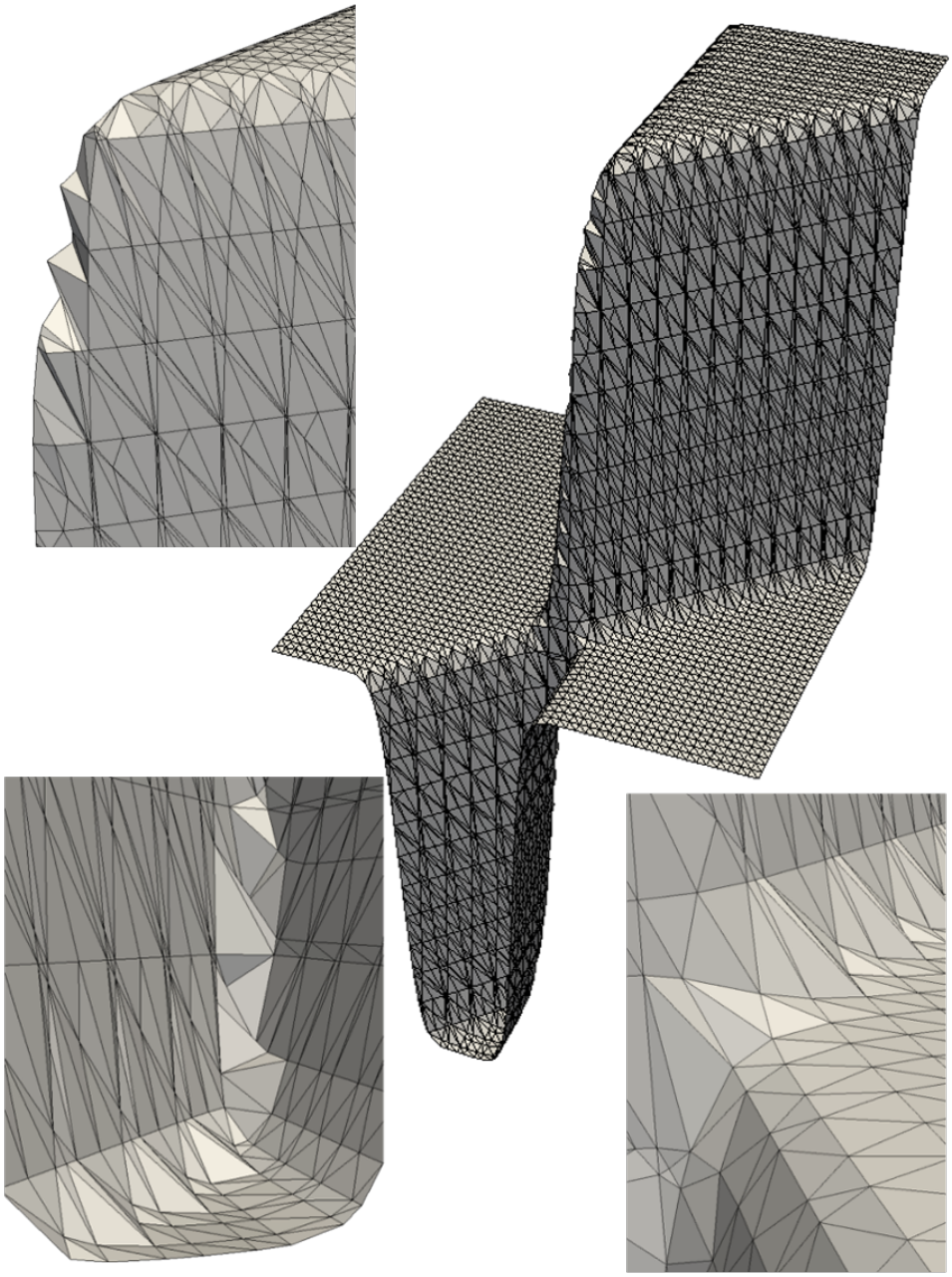


Figure 4.20: The initial mesh for **Example** 3.

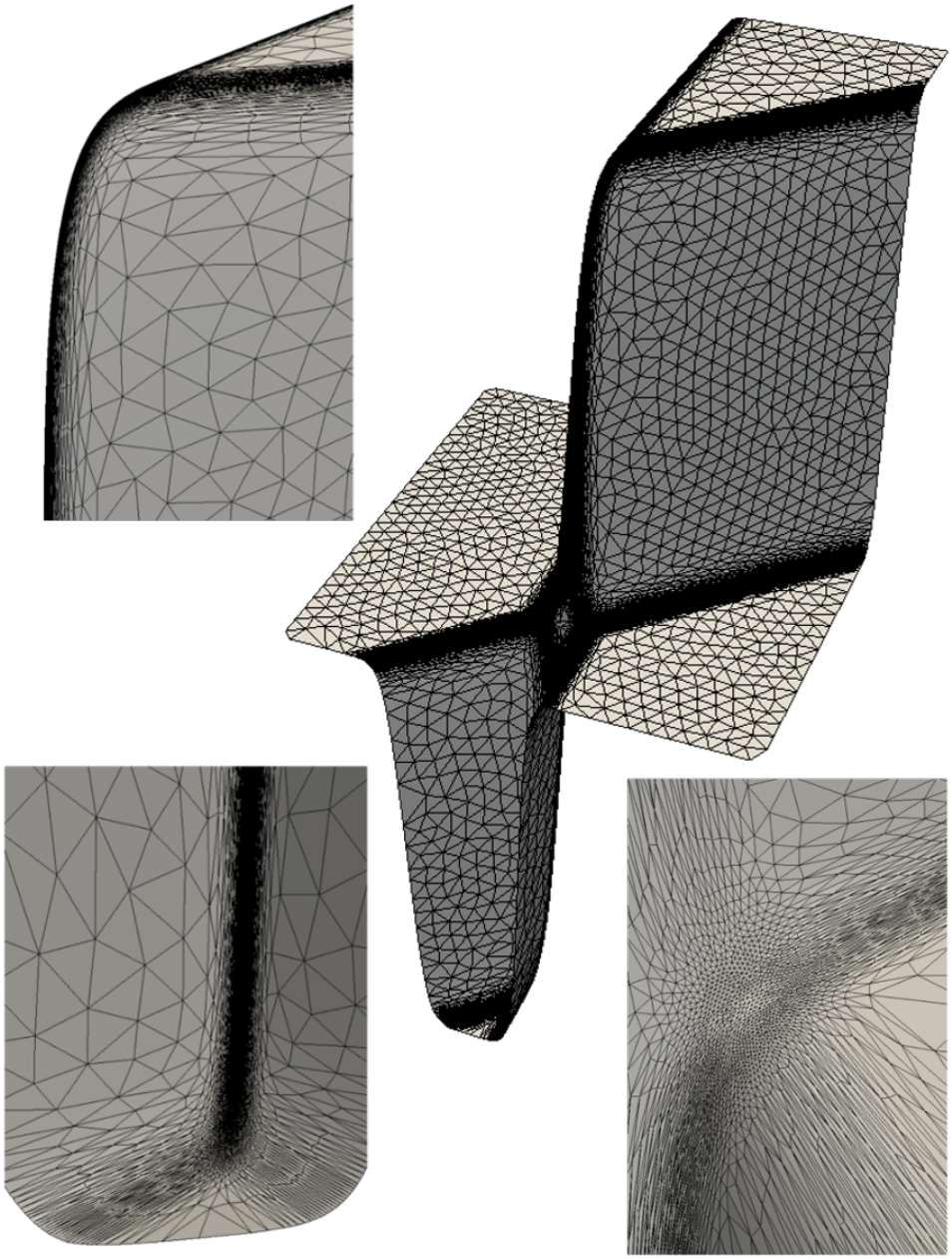


Figure 4.21: The resulting mesh for **Example** 3.

Example 4

In this example, we construct a surface containing sharp features. The purpose is to illustrate that the sharp features are well preserved by the proposed re-meshing method. The input of this example consists in three different patches, i.e., three different smooth surfaces, that are joined together along their common boundaries, see Figure 4.22.

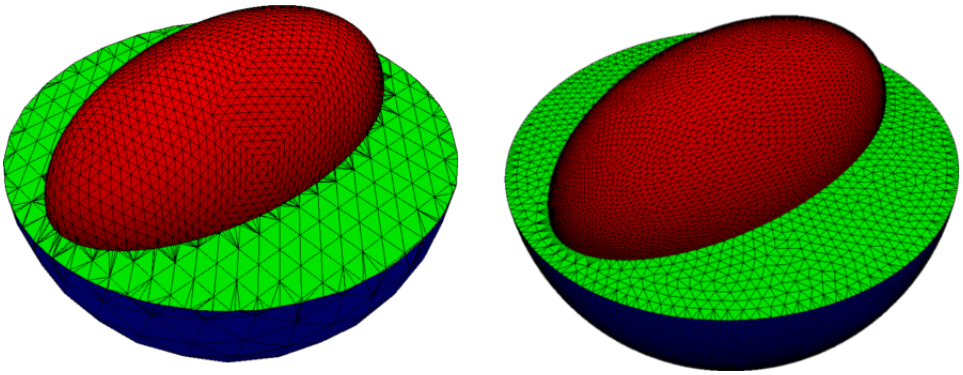


Figure 4.22: **Example** 4: the initial mesh on the left, the resulting one on the right.

From Figure 4.23, we observe that the proposed re-meshing method **does** preserve the sharp features.

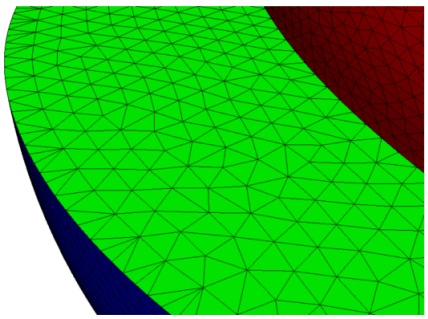


Figure 4.23: A detail of the mesh in **Example** 4.

4.3.1 Choice of the parameter s

In all the previous examples we have considered the same factor $s = 1$ in the re-meshing algorithm. Now, we change this value to understand the effect on the resulting mesh.

We define the function

$$f_4 : [-1., 1.] \times [-1., 1.] \times [-2., 2.] \to \mathbb{R},$$

such as

$$f_4(x, y, z) := \tanh(20y) - \tanh(20(x - y) - 10) - z, \tag{4.10}$$

whose zero level set is the surface represented in Figure 4.24. This surface presents some flat regions, like the zone A in Figure 4.24, and a series of very deep jumps, see the arrows in Figure 4.24.

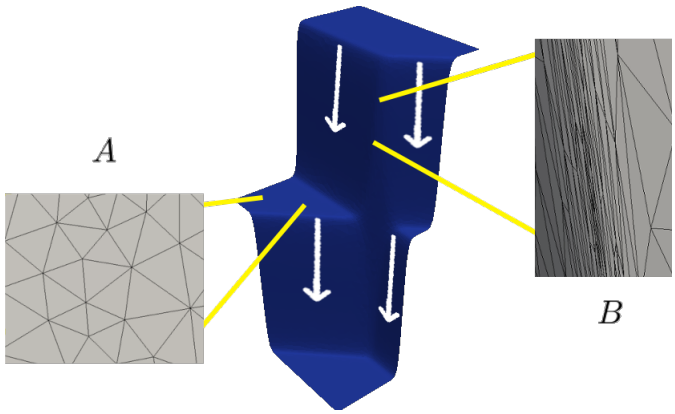


Figure 4.24: The geometry used for the **Choice of the parameter** s. The zero level set of the function f_4; some zones are highlighted to show how the adapted mesh will correctly works.

We fix a desired 6d-length and we consider these values for the parameters:

$$s = [0.1,\ 1.,\ 5.,\ 25.].$$

If we increase the factor s, the 6d-length of the edges of the mesh will increase, **but** this fact it is not completely true. The **actual** effect of increasing the parameter

s is to **emphasize** the variation of the normal, i.e., the variation of the curvature. In fact, where the surface is flat, the size of the mesh elements is the same for each values of s. On the contrary, where the surface exhibits a variation of the curvature, it is more and more refined for higher values of s. When we are dealing with big values of s, a small variation of the normals corresponds to large variation of the edge length, so the sampling procedure may refine these edges. The meshes obtained with different values of s, are shown in Figure 4.25 and we propose some details in Figure 4.26. In Table 4.2 we provide the minimum value of the aspect ratio for each mesh. From these data we observe that higher values of s increase the stretching of triangles and the number of elements.

Remark 4.3.1 *High values of s emphasize the variation of the normal in the mesh. This fact is exploited in [71], where the authors use very small values for s in order to find the most significant sharp features of the input surface Γ.*

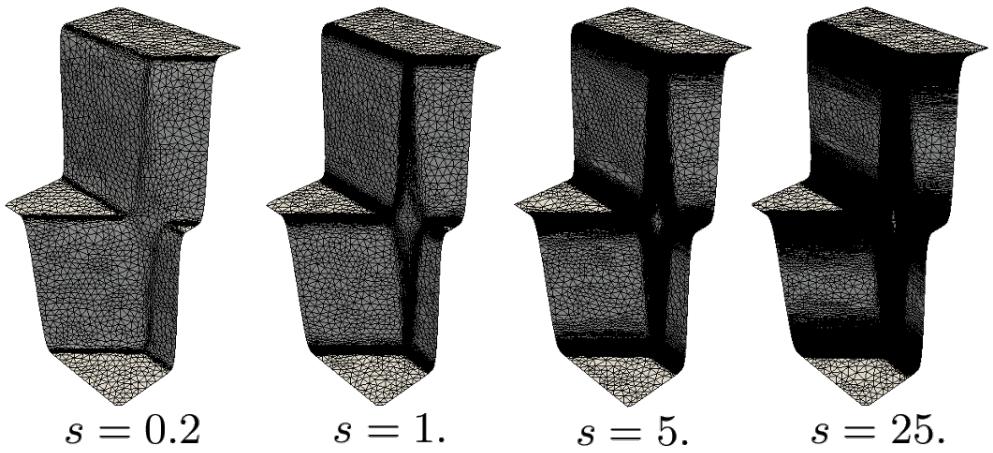


Figure 4.25: Resulting meshes for different values of the parameter s.

	$s = 0.2$	$s = 1.$	$s = 5.$	$s = 25.$
Minimum Aspect Ratio	1.472e-01	3.173e-04	8.414e-05	4.162e-07
# vertices in $\mathcal{T}_{output}$	2652	8583	54001	409444
# triangles in $\mathcal{T}_{output}$	5067	16696	107239	817605

Table 4.2: Minimum value of the aspect ratio with different value for s and the number of vertices and elements.

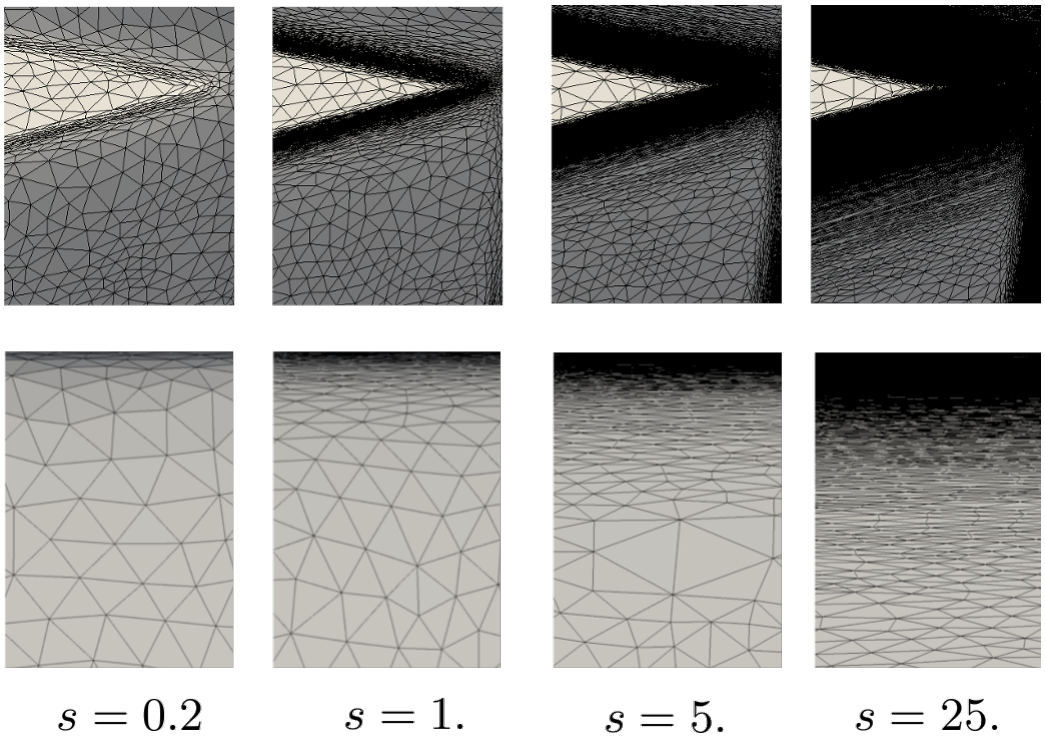


Figure 4.26: Details of the optimized meshes in Figure 4.25.

4.3.2 CAD Models

We consider CAD models to show that the proposed method is able to preserve sharp features of very complex geometries. We use `Gmsh` library [50] to get the initial mesh and to get the functions η_1 and η_2, see Subsection 4.2.1 for more details.

Example 5

The method of B. Lévy and N. Bonneel [75] does not preserve the sharp features, i.e., they are oversampled and smoothed. We consider one of the examples provided in [75] and we apply the proposed re-meshing procedure to make a comparison between these two re-meshing algorithms.

In Figure 4.27 on the left, we show the initial rough mesh, then in Figure 4.27 on the right, we provide the final adapted mesh. Even in the case of complex CAD geometries, the proposed re-meshing method is able to reconstruct the complex geometries moving from a really rough initial mesh.

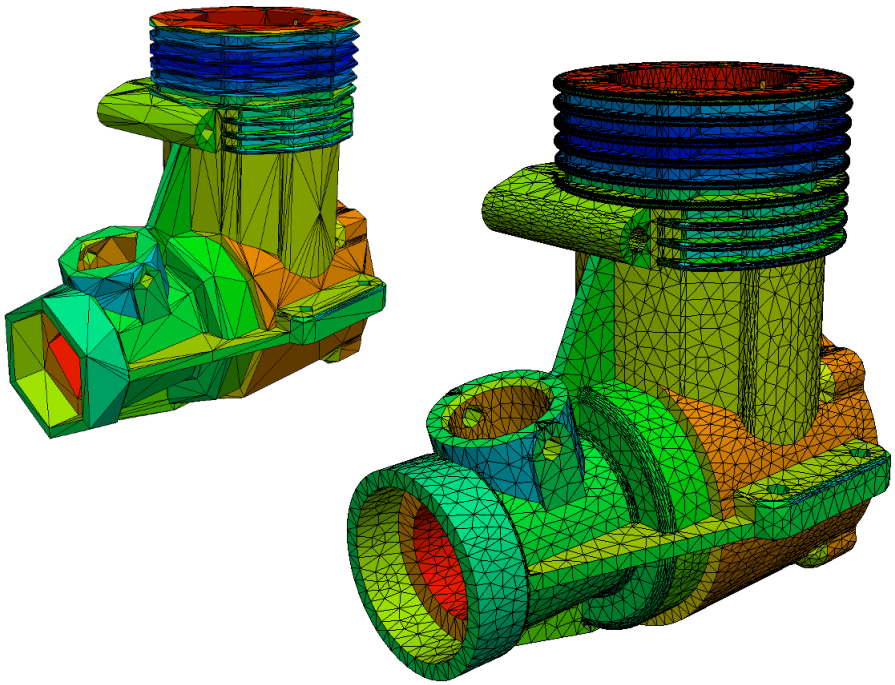


Figure 4.27: The initial mesh, on the left, and the optimized mesh, on the right.

In Figure 4.30, 4.29 and 4.28, we compare the mesh provided by B. Lévy and N. Bonneel in [75] with the same detail obtained by the proposed re-meshing method. We can see that sharp features are **preserved** and they are neither smoothed nor oversampled.

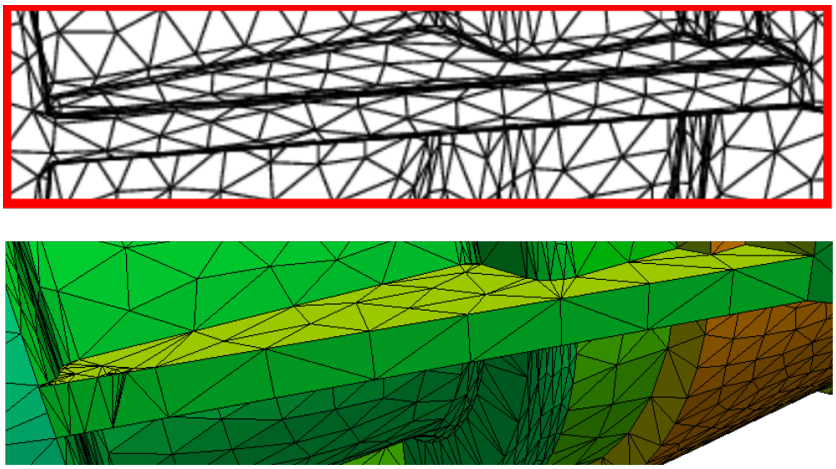


Figure 4.28: A detail of the mesh proposed in [75], top, and the same detail of the mesh obtained by the proposed method, bottom.

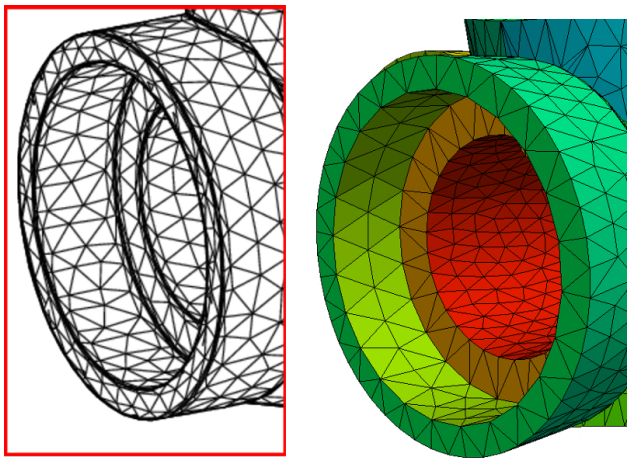


Figure 4.29: A detail of the mesh proposed in [75], on the left, and the same detail of the mesh obtained by the proposed method, on the right.

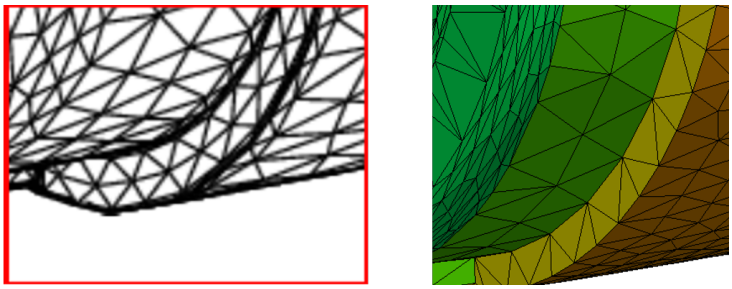


Figure 4.30: A detail of the mesh proposed in [75], on the left, and the same detail of the mesh obtained by the proposed method, on the right.

Example 6

This is another example in [75]. Also in this case, we observe that the proposed method does **not** smooth the sharp features, but they are preserved. Even in this case the geometry is complex and the initial surface mesh is a really rough approximation of the actual geometry. Nevertheless, the proposed re-meshing method gives a very good final mesh.

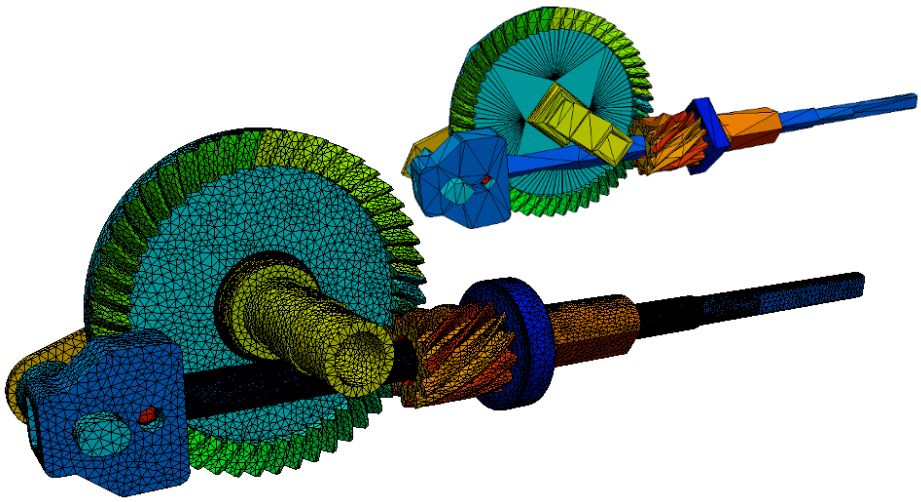


Figure 4.31: The initial mesh, top, and the optimized mesh, bottom.

In Figure 4.32, 4.33, 4.34 and 4.35, we compare the mesh provided by B. Lévy and N. Bonneel in [75] with the same detail obtained by this new re-meshing method. Even in this case, we can see that the re-meshing algorithm **does** preserve sharp features.

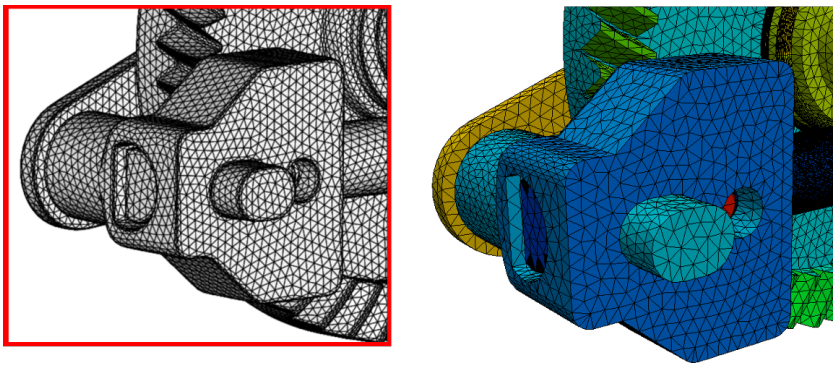


Figure 4.32: A detail of the mesh proposed in [75], on the left, and the same detail of the mesh obtained by the proposed method, on the right.

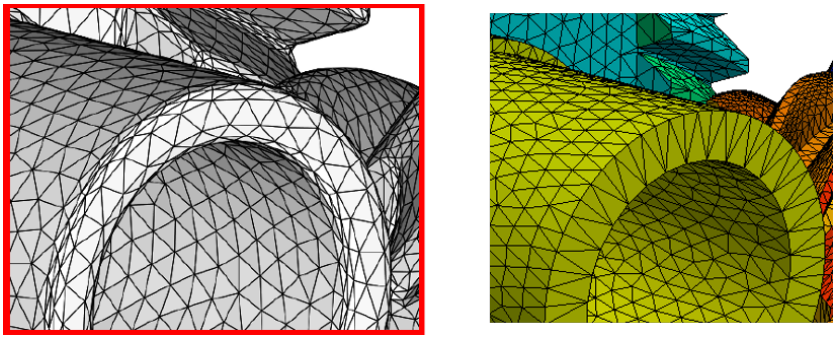


Figure 4.33: A detail of the mesh proposed in [75], on the left, and the same detail of the mesh obtained by the proposed method, on the right.

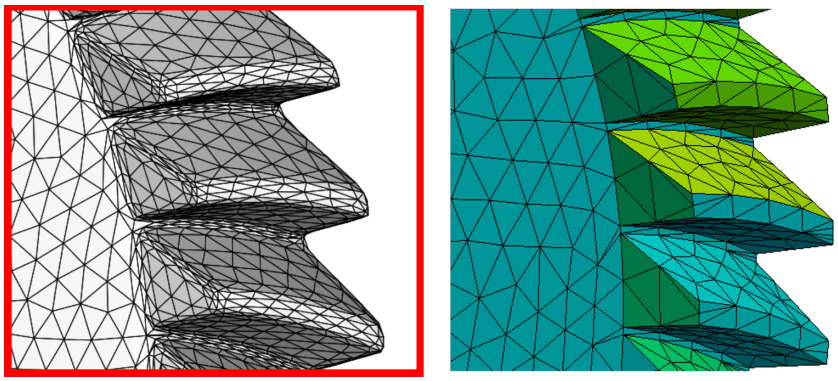


Figure 4.34: A detail of the mesh proposed in [75], on the left, and the same detail of the mesh obtained by the proposed method, on the right.

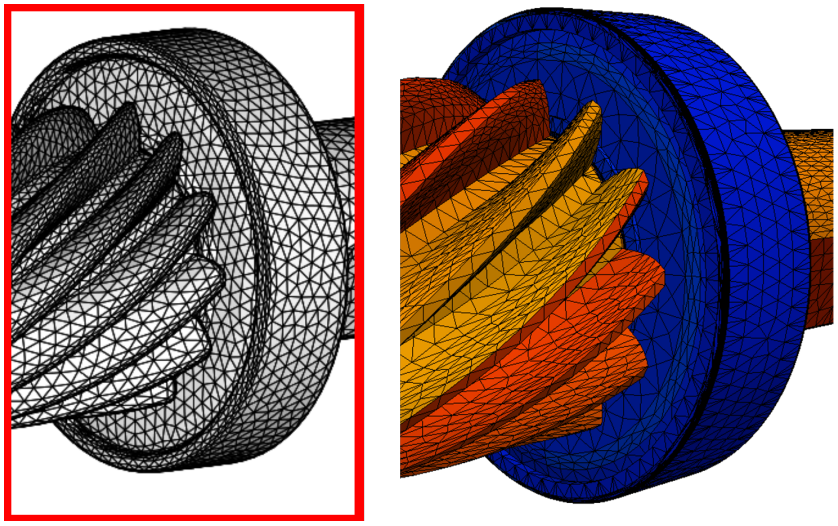


Figure 4.35: A detail of the mesh proposed in [75], on the left, and the same detail of the mesh obtained by the proposed method, on the right.

Chapter 5

Surface Mesh Simplification

There is a large variety of computational applications that involve complex and highly detailed meshes. To preserve this good level of realism, they may require a huge number of nodes and elements, so they are computationally too expensive. One of the possible strategies to overcome this issue is to consider simplified models with a lower number of elements and nodes. This is a common issue in computer graphics, [61, 48], and it is recurrent also in the finite element framework, [39, 42, 81].

In the first part of this chapter we introduce a mesh simplification algorithm that reduces the number of elements of a surface mesh simply moving from geometrical considerations. The proposal of this algorithm becomes clear in the second part of the chapter where we provide a new mesh simplification algorithm to deal with a statistical analysis of data distributed on complex geometries. The geometric criterion is properly enriched to get a simplified mesh where data are re-distributed without loosing the desired good inferential properties.

A challenging and important application, the analysis of brain cortex thickness data, drives the proposal of this new contraction algorithm.

5.1 Mesh Simplification

Consider an initial surface mesh Γ_h with n nodes, that approximates a geometry defined by a surface Γ. A general simplification process aims at building a new mesh Γ'_h with m nodes, where $m \ll n$, that preserves the surface as well as possible.

Several different strategies have been presented in the literature to achieve this

goal. They can be classified as follows:

- vertex decimation;
- vertex clustering;
- iterative edge contraction.

Vertex Decimation

In [102], W. J. Schroeder et al. introduce an algorithm that iteratively removes a vertex from a polyhedral mesh, in order to get a simplified model. Once a vertex **v** is removed, all the faces that share this node are removed and the resulting hole is re-triangulated. In [109], M. Soucy et al. use this simplification strategy. They improve the efficiency of this approach in [102] to get an higher fidelity to the original model. Unfortunately, both these mesh simplification methods use vertex classification and re-triangulation schemes that are limited to manifold surfaces.

Figure 5.1: Example of vertex decimation.

Iterative Edge Contraction

There is a large variety of algorithms that simplify a model via an iterative edge contraction. The main difference among them is how an edge of the mesh is chosen for the contraction, see Figure 5.2 and Subsection 1.2.3. The most common and notable algorithm is provided by Hoppe [62], R. Ronfard and J. Rossignac [94] and A. Guéziec [54]. All these algorithms have been designed for manifold surfaces, but, since they use an iterative edge contraction, they may be generalized to non-manifold surfaces.

In [47], M. Garland and P. S. Heckbet propose an extension of the iterative edge contraction. They do not contract an edge, but generic pairs of vertices of the mesh, i.e., a pair of vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ that are the end-points of an edge or too

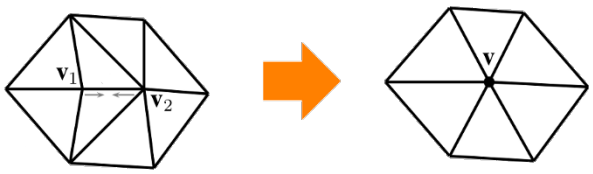


Figure 5.2: Example of edge contraction.

close each other. Indeed, they fix a tolerance t. Then, a generic pair $\mathbf{v}_1$ and $\mathbf{v}_2$ can be contracted if:

(a) $\mathbf{v}_1\mathbf{v}_2$, is an edge of the mesh, see Figure 5.2;

(b) $||\mathbf{v}_1 - \mathbf{v}_2|| < t$, i.e., the vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ are too close, see Figure 5.3,

where $||\cdot||$ is the standard Euclidean norm.

By contracting arbitrary vertex pairs, i.e., not **only** edges, this algorithm may join unconnected regions. So, it could facilitate much better approximations, both visually and with respect to geometric error. Moreover, since it allows topological joining, the method proposed by Garland and Heckbet supports non-manifold surface models.

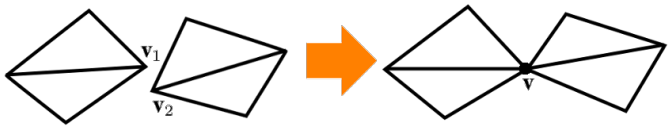


Figure 5.3: Pair contraction of the type (b) proposed by Garland and Heckbert.

Vertex Clustering

One of the most well-known mesh simplification algorithms based on vertex clustering is the one proposed by J. Rossignac and P. Borrel, [95]. Conversely to edge contraction and vertex decimation, this simplification algorithm is not iterative, but it creates the simplified mesh in a single step. A bounding box is placed around the original mesh and it is divided into a grid. Then, in each cell of the grid the vertices are clustered together into a single vertex and the faces of the mesh are suitably updated, see Figure 5.4.

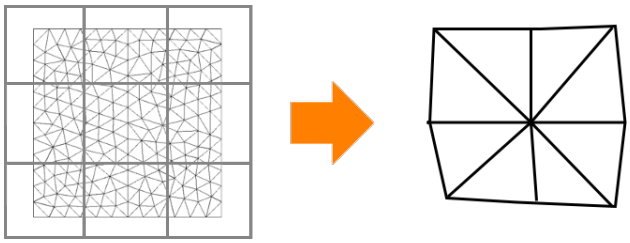


Figure 5.4: Example of vertex clustering.

Despite its speed, this method presents some drawbacks. Firstly, the resulting mesh may lose important features of the real geometry. Moreover, the number of faces is indirectly determined by the specified grid dimension and it depends on the position and orientation of the original model with respect to the surrounding grid, so it may be difficult to construct a simplified mesh with a desired number of elements. In [77], a generalization to a non uniform grid is proposed.

In this Chapter we present two simplification procedures based on the iterative edge contraction. In particular, we will assume that the geometry consists in triangles, i.e., we do not have an exact representation of the real geometry.

5.2 Geometric Mesh Simplification

In this section we explain in more details a mesh simplification strategy based on an iterative edge contraction. The algorithms presented here are not new, but they are functional to better understand the novel mesh simplification method based on statistical considerations described in Section 5.3.

An iterative edge contraction procedure may be formalized in this way: given an initial triangular mesh Γ_h, with n vertices, we contract the edges of the mesh until we get a new mesh Γ'_h with a desired number of vertices m, $m \ll n$. But, this issue remains: how can we choose the edges to be contracted?

We overcome this problem via the notion of **contraction cost**: we associate a positive real number c with each edge e of the mesh. This number represents the loss of geometric accuracy due to the contraction of e. Then, if we iteratively contract the edges associated with the lowest cost, the important geometric features of the mesh should **not** be lost in the final mesh.

5.2.1 Geometric Cost

Before dealing with the definition of the contraction cost, we need the following results.

Remark 5.2.1 *A generic plane in* $\mathbb{R}^3$ $\pi : ax+by+cz+d=0$ *can be represented by a vector in* $\mathbb{R}^4$*:*

$$\boldsymbol{p} = (a,\, b,\, c,\, d)^t\,, \tag{5.1}$$

where a*,* b*,* c *and* d *are real numbers such that* $a^2+b^2+c^2=1$*,*

Proposition 5.2.1 *Consider a generic point* $\boldsymbol{v} \in \mathbb{R}^3$ *and a plane* π *in* $\mathbb{R}^3$*; the signed distance of the point* $\boldsymbol{v}$ *to the plane* π *can be computed as the scalar product*

$$\mathbf{w}^t\,\boldsymbol{p}\,, \tag{5.2}$$

where $\boldsymbol{p} \in \mathbb{R}^4$ *is the vector that represents the plane* π*, see Remark 5.2.1, and* $\mathbf{w} \in \mathbb{R}^4$ *is defined as*

$$\mathbf{w} = (v_x,\, v_y,\, v_z,\, 1)^t\,,$$

with v_x*,* v_y *and* v_z *the components of* $\boldsymbol{v}$ *in* $\mathbb{R}^3$*.*

Proof. Consider a generic point $\mathbf{v} = (v_x,\, v_y,\, v_z)^t$, a plane π and a generic point $\mathbf{u} = (u_x,\, u_y,\, u_z)^t$ that lies on the plane π. We choose the coefficients a, b and c of the implicit form of the plane $ax+by+cz+d=0$, such that $a^2+b^2+c^2=1$. These three coefficients represent the components of the normal $\mathbf{n}$ to the plane π. Then, via $\mathbf{u}$, we can compute the last coefficient of the plane π, $d=-\mathbf{u}^t\mathbf{n}$.

We introduce the vectors $\mathbf{w}, \mathbf{p} \in \mathbb{R}^4$, with $\mathbf{w} = (v_x,\, v_y,\, v_z,\, 1)^t$ and $\mathbf{p} = (a,\, b,\, c,\, d)^t$, see Remark 5.2.1. Thus, the scalar product in $\mathbb{R}^4$ could be written in terms of the implicit form of the plane and we have

$$\mathbf{w}^t\,\mathbf{p} = av_x + bv_y + cv_z + d = (\mathbf{x}-\mathbf{u})^t\mathbf{n}\,.$$

Since $\mathbf{n}$ is a unitary vector, from the chain of equalities below it is clear that we get the signed distance of $\mathbf{v}$ from the plane π, see Figure 5.5,

$$(\mathbf{v}-\mathbf{u})^t\mathbf{n} = ||\mathbf{v}-\mathbf{u}||\,||\mathbf{n}||\,\cos\vartheta = ||\mathbf{v}-\mathbf{u}||\,\cos\vartheta = l\,,$$

with $||\cdot||$ the standard Euclidean norm.

□

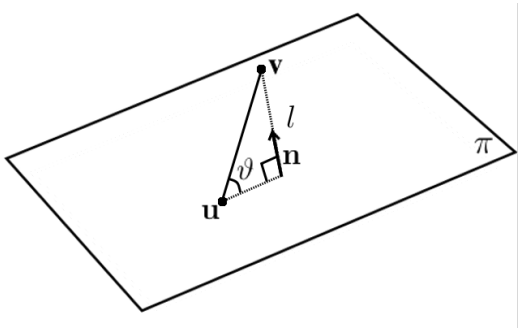


Figure 5.5: The plane π, a generic point $\mathbf{v}$; we highlight the signed distance l and the angle ϑ between the vector $\mathbf{uv}$ and $\mathbf{n}$.

Now we are ready to define the geometrical cost. More precisely we define this cost via the error in the neighborhood of each vertex by a quadric form, see [34, 94].

In the original geometry each vertex $\mathbf{v}$ of the triangular mesh is the intersection of the planes defined by the triangles that share $\mathbf{v}$. Consider a point $\mathbf{v}$ of the initial mesh and the set of triangles sharing it. For each of these triangles, we can define a plane and, consequently, a point $\mathbf{p} \in \mathbb{R}^4$, see Remark 5.2.1 and we denote by $\omega_{\mathbf{v}}$ the set of these vectors. Moving from these planes and Proposition 5.2.1, we define the quadric error at a vertex $\mathbf{v}$ as the sum of the squared distances from the point $\mathbf{v}$ to the planes identified by the triangles, [34, 94]:

$$\begin{aligned}
\Delta(\mathbf{v}) &= \sum_{T \in \omega_{\mathbf{v}}} (\mathbf{w}^t \, \mathbf{p})^2 \\
&= \sum_{T \in \omega_{\mathbf{v}}} (\mathbf{w}^t \, \mathbf{p})(\mathbf{p}^t \mathbf{w}) \\
&= \mathbf{w}^t \left(\sum_{T \in \omega_{\mathbf{v}}} \mathbf{p}\mathbf{p}^t \right) \mathbf{w} \\
&= \mathbf{w}^t \mathbf{Q}_{\mathbf{v}} \mathbf{w} \, ,
\end{aligned}$$

where $\mathbf{w} = (v_x, \, v_y, \, v_z, \, 1)^t$ and $\mathbf{v} = (v_x, \, v_y, \, v_z)^t$, see Proposition 5.2.1. We associate the matrix $\mathbf{Q}_{\mathbf{v}}$ with each vertex $\mathbf{v}$ of the initial mesh. Then, we define the contraction cost of a given edge e into the node $\mathbf{v}$ as:

$$c_{\text{geo}}(e, \mathbf{v}) := \mathbf{w}^t \, \overline{\mathbf{Q}} \, \mathbf{w} \, , \tag{5.3}$$

where

$$\overline{\mathbf{Q}} := \mathbf{Q}_{\mathbf{v}_1} + \mathbf{Q}_{\mathbf{v}_2}\,, \tag{5.4}$$

with $\mathbf{Q}_{\mathbf{v}_1}$ and $\mathbf{Q}_{\mathbf{v}_2}$ are the matrices associated with the endpoints of the segment e.

In a standard edge contraction, the edge e is contracted into one of its endpoints, $\mathbf{v}_1$ and $\mathbf{v}_2$, or into the middle point of the segment $\mathbf{v}_1\mathbf{v}_2$. Since we are dealing with a cost function c_{geo}, that represents the loss of geometric accuracy, if we fix the edge to be contracted, it is possible to find the position $\mathbf{v}_{\text{opt}}$ that minimizes c_{geo}.

Proposition 5.2.2 *Consider the edge e, whose endpoints are $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. The point $\boldsymbol{v}_{opt}$ that minimizes the cost function c_{geo}, defined in Equation (5.3), is given by:*

$$\boldsymbol{v}_{opt} = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix}^{-1} \begin{bmatrix} -q_{14} \\ -q_{24} \\ -q_{34} \end{bmatrix}\,, \tag{5.5}$$

where q_{ij} are the components of the matrix $\overline{\boldsymbol{Q}}$, Equation(5.4).

Proof. We recall that the matrix $\overline{\mathbf{Q}}$ is symmetric, so $q_{i,j} = q_{j,i}\ \forall i, j = 1, 2, 3$. We consider a generic vector $\mathbf{v} = (x,\, y,\, z)^t$ in $\mathbb{R}^3$ and its corresponding vector in $\mathbb{R}^4$, $\mathbf{w} = (x,\, y,\, z,\, 1)^t$ and the edge e. The cost function c_{geo} is a quadratic form in the variables x, y and z. In fact we have

$$\begin{aligned} \Delta(x,\, y,\, z) = \mathbf{w}^t\, \overline{\mathbf{Q}}\, \mathbf{w} \;&=\; q_{11}\, x^2 + q_{22}\, y^2 + q_{33}\, z^2 + \\ &+\; 2q_{12}\, xy + 2q_{13}\, xz + 2q_{23}\, yz + \\ &+\; q_{14}\, x + q_{24}\, y + q_{34}\, z + q_{44}\,. \end{aligned}$$

To find the minimum of this quadratic form, we compute the partial derivatives of $\Delta(x,\, y,\, z)$ and we look for $(\overline{x},\, \overline{y},\, \overline{z})^t$ such that

$$\frac{\partial \Delta(\overline{x},\, \overline{y},\, \overline{z})}{\partial x} = \frac{\partial \Delta(\overline{x},\, \overline{y},\, \overline{z})}{\partial y} = \frac{\partial \Delta(\overline{x},\, \overline{y},\, \overline{z})}{\partial z} = 0\,.$$

This is equivalent to solve the following linear system:

$$\begin{cases} q_{11}\, \overline{x} + q_{12}\, \overline{y} + q_{13}\, \overline{z} \;=\; -q_{14} \\ q_{12}\, \overline{x} + q_{22}\, \overline{y} + q_{23}\, \overline{z} \;=\; -q_{24} \\ q_{13}\, \overline{x} + q_{23}\, \overline{y} + q_{33}\, \overline{z} \;=\; -q_{34} \end{cases}\,,$$

and this completes the proof.

□

Remark 5.2.2 *Since it is not guaranteed that the matrix*

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix}$$

is invertible, we can ***not*** *always find the optimal position* $\boldsymbol{v}_{opt}$*, see Equation* (5.5)*. When this occurs, the edge* e *is contracted into another point. We choose this new point among the endpoints and the middle point of the segment, according to the lowest value of* c_{geo}*.*

Remark 5.2.3 *In the initial mesh for each vertex,* $\boldsymbol{v}$*, we can define the matrix* $\boldsymbol{Q}_{\boldsymbol{v}}$*, but once we contract an edge* $\boldsymbol{v}_1\boldsymbol{v}_2$ *into a vertex* $\boldsymbol{v}^*$*, we have to create a matrix to be associated with the new point* $\boldsymbol{v}^*$*. We decide to associate with* $\boldsymbol{v}^*$ *the matrix* $\overline{\boldsymbol{Q}}$ *in Equation* (5.4)*.*

5.2.2 Summary of the Algorithm

In Algorithm 5, we summarize the adopted mesh simplification algorithm. The inputs for the algorithm are the initial mesh Γ_h with n vertices and a desired number m of vertices, such that $n \gg m$.

We have implemented a **dynamic** data structure that, for each triangle of the current mesh, stores a **valid edge**, i.e., the edge of the triangle that minimizes the cost function defined in Equation (5.3), and whose contraction does not produce an undesired topological configuration, see Subsection 1.2.3. Then, we iteratively contract the edge with the lowest cost, until we reach the desired number of vertices. This data structure is suitably updated after each edge contraction, line 6 in Algorithm 5.

As we will show in the following subsection, the SIMPL routine gives very good final meshes, i.e., meshes that, despite the lower number of vertices, offer a good approximation of the initial geometry, but the computational time to get this result is usually high.

To overcome this issue, we modify this algorithm. In particular, we propose the procedure represented in Algorithm 6. It has the same inputs as SIMPL, but we add the integer t. This new algorithm contracts all the first t edges in the list E without updating the data structure, line 4 in Algorithm 6. Then, for each iteration of the **while** cycle, it rebuilds the data structure with a lower number of elements, line 5 in Algorithm 6.

Algorithm 5 The simplification algorithm

SIMPL(Γ_h, m)

Data: E is the dynamic data structure for the triangles of Γ_h and n is the current number of vertices in Γ_h.

1: initialize n with number of vertices in Γ_h;
2: initialize E with all the triangles in Γ_h;
3: **while** $m <= n$ **do**
4: find the cheapest valid edge;
5: contract e;
6: update the data structure;
7: update n;
8: **end while**

Algorithm 6 New simplification algorithm with steps

SIMPLSTEP(Γ_h, m, t)

Data: E is the dynamic data structure for the triangles of Γ_h and n is the actual number of vertices in Γ_h.

1: initialize n with number of vertices in Γ_h;
2: initialize E for all the triangles in Γ_h;
3: **while** $m <= n$ **do**
4: do the first t contractions stored in E;
5: rebuild the data structure E;
6: update n;
7: **end while**

5.2.3 Numerical Examples

In this section, several examples are presented to demonstrate the reliability of the proposed simplification methods, i.e., SIMPL and SIMPLSTEP. First, we simplify the meshes with SIMPL. In Table 5.1 we provide the corresponding computational time for each of these examples. Then, we compare the performance of SIMPL and SIMPLSTEP when we consider a mesh with a huge number of vertices.

In all these examples we consider an initial triangular mesh Γ_h with n number of vertices. Then, we reduce these meshes up to 80%, 60%, 40%, 20% and 5% of the initial number of vertices.

All the geometries used in the following examples can be found in the Stanford 3D Scanning Repository, [1].

Examples	1	2	3
# vertices in Γ_h input	2412	2903	35613
Time (sec.) 80%	5	6	82
Time (sec.) 60%	8	10	140
Time (sec.) 40%	13	14	195
Time (sec.) 20%	16	17	252
Time (sec.) 5%	19	20	290

Table 5.1: Statistics of on the first three geometries by using the routine SIMPL.

Example 1: the pawn geometry

In this case we consider a very simple geometry: a pawn. In Figure 5.6, we show the initial mesh and the geometry at each level of the simplification.

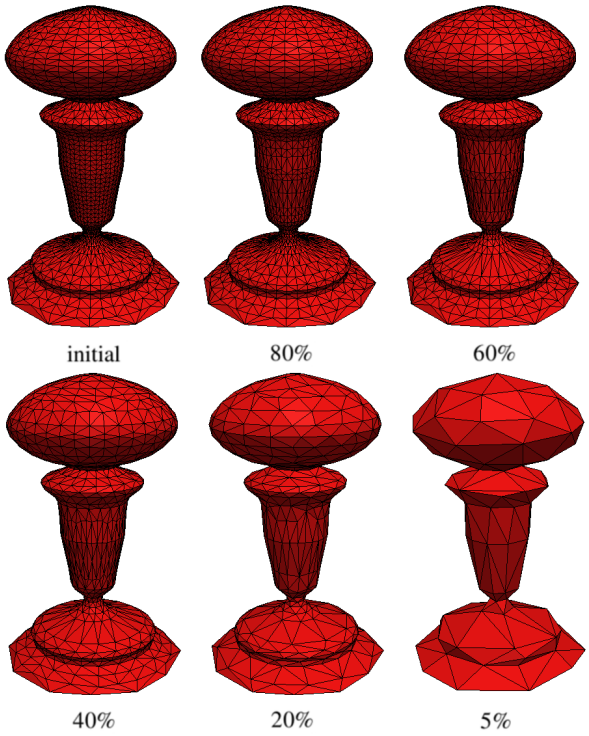


Figure 5.6: Initial mesh for different levels of simplification.

In this example we could appreciate that the simplification procedure starts to remove the edges that lie on the same plane. In fact, the bottom of the pawn is flat and the algorithm simplifies here as much as possible, see Figure 5.7. We see that, after the first steps of simplification, 80%, the bottom of the pawn remains unchanged.

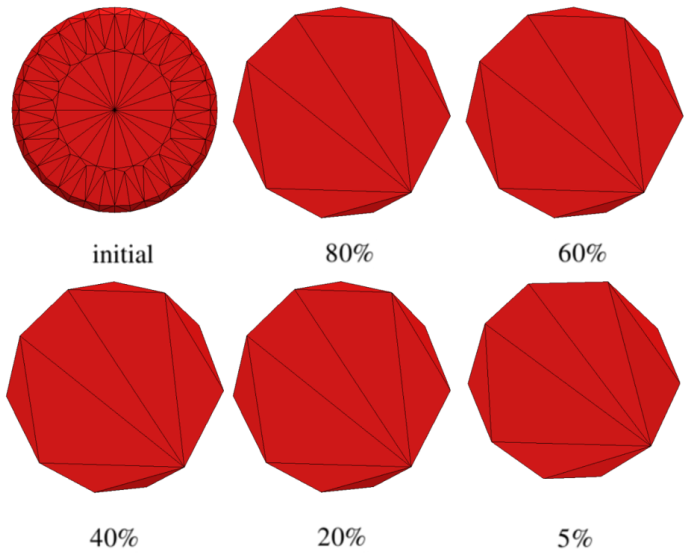


Figure 5.7: Detail of the initial mesh for different levels of simplification.

Example 2: the cow geometry

In this example, we consider a more complex geometry and we proceed with the simplification process SIMPL. From Figure 5.8, we see that the geometry is preserved as much as possible, even if we are dealing with a very small number of elements.

Since the cost function is essentially related to the curvature, the resulting mesh does not contain equilateral triangles, but the size, shape and orientation of the triangles depend on the curvature of the surface, see Figure 5.9.

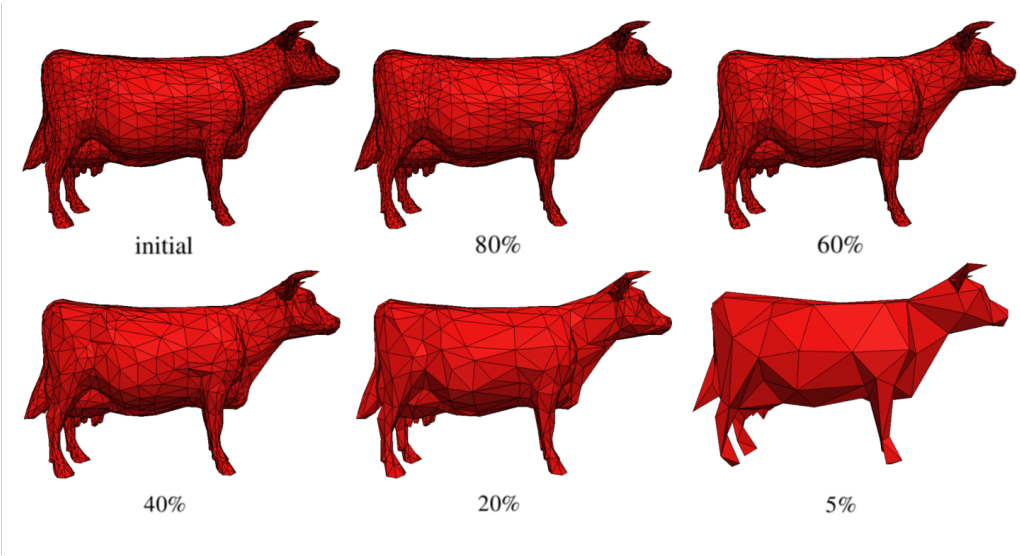


Figure 5.8: Initial mesh for different levels of simplification.

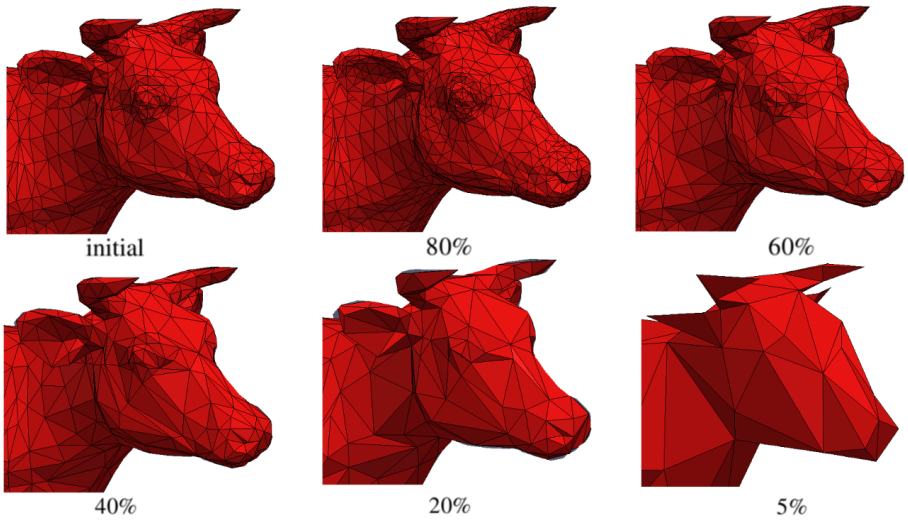


Figure 5.9: Detail of the initial mesh for different levels of simplification.

Example 3: the Stanford bunny geometry

In Figure 5.10 we show the whole geometry for different levels step of simplification. From the detail in Figure 5.11, we notice that the elements are oriented in the right direction in order to get the best approximation of the surface.

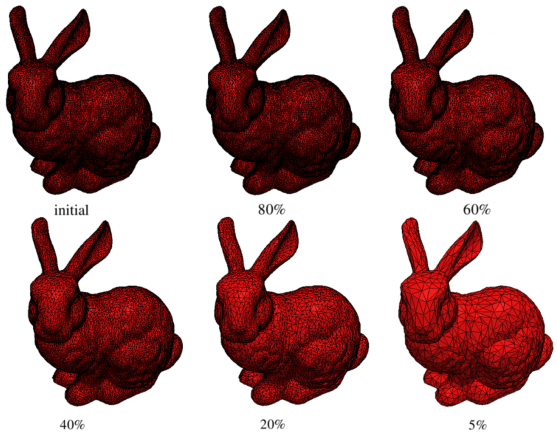


Figure 5.10: Initial mesh for different levels of simplification.

From Figure 5.10, we appreciate that the geometry of the bunny in well preserved even if we reduce the number of vertices to the 5% of the total vertices. This fact becomes more evident from the detail in Figure 5.11.

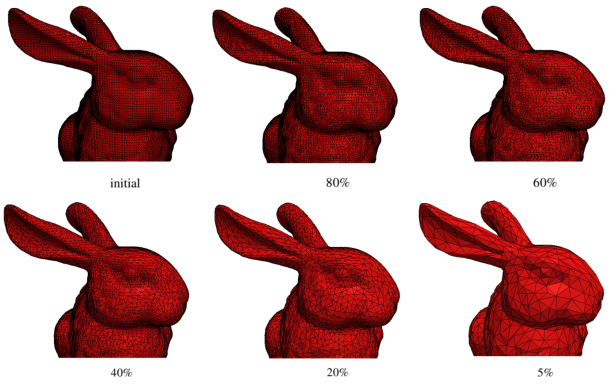


Figure 5.11: Detail of the initial mesh for different levels of simplification.

Numerical examples with a huge number of vertices

Now we make a comparison between the proposed simplification algorithms, SIMPL and SIMPLSTEP. The tests have been done with a laptop with a 2.26 GHz processor and the corresponding timings are shown in Table 5.2.

Examples	4		5	
Routine	SIMPL	SIMPLSTEP	SIMPL	SIMPLSTEP
# vertices in Γ_h input	165954	165954	514300	514300
Time (sec.) 80%	407	150	1354	808
Time (sec.) 60%	644	264	2246	1446
Time (sec.) 40%	925	267	3083	1885
Time (sec.) 20%	1211	314	3878	2023
Time (sec.) 5%	1349	319	4425	2158

Table 5.2: Statistics for examples 4 and 5 using both the routines SIMPL and SIMPLSTEP.

These data show that the simplification routine SIMPLSTEP is clearly more efficient than SIMPL in terms of computational time: for each simplification process the computational time for the SIMPLSTEP algorithm, is fewer than running-time of the SIMPL routine.

From the geometric point of view, the meshes simplified by SIMPLSTEP still offer a good approximation of the surface Γ, **but** the size, orientation and shape of the triangles do not reflect the curvature. In particular, we notice that the algorithm SIMPLSTEP contracts the edges in a specific region and then it moves to another one. On the contrary, SIMPL does not focus on particular zones of the mesh, but it contracts the edges over the whole geometry in order to reach the desired number of vertices.

Example 4: the armadillo geometry

We compare the final meshes obtained with SIMPL and SIMPLSTEP. The simplification process SIMPL produces a more uniform distribution of the triangles on the armadillo geometry. The sequence of meshes in Figure 5.12, 5.13, 5.14, 5.15 and 5.16 exemplifies this observation. Here, for each mesh vertex, $\mathbf{v}_i$, we provide the maximum length of the edges connected to $\mathbf{v}_i$. Moving from these figures, we easily understand that the algorithm SIMPLSTEP focuses the contraction in par-

ticular zones, see e.g., the chest and the ears of the armadillo, while the SIMPL produces a more uniform mesh.

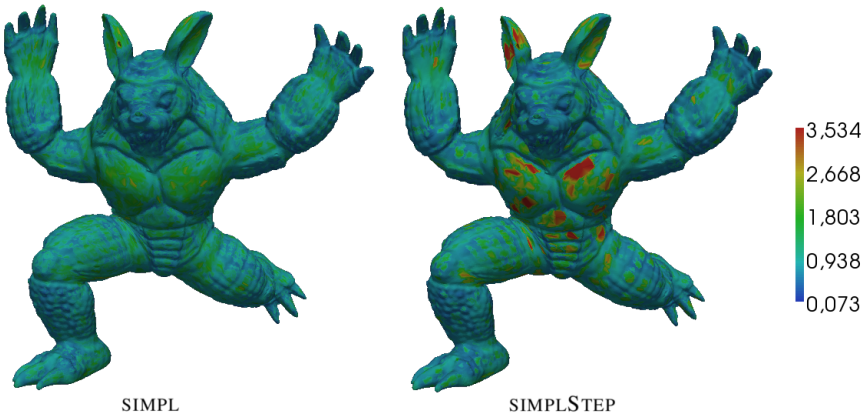


Figure 5.12: Simplified mesh up to 80% of the initial number of vertices, on the left the one obtained with SIMPL, on the right the one obtained with SIMPLSTEP, for each node $\mathbf{v}_i$ we provide the maximum length of the edges connected to $\mathbf{v}_i$.

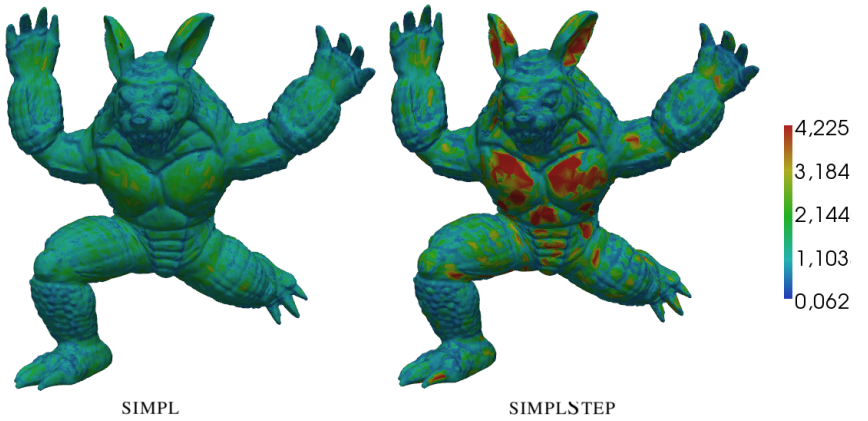


Figure 5.13: Simplified mesh up to 60% of the initial number of vertices, on the left the one obtained with SIMPL, on the right the one obtained with SIMPLSTEP, for each node $\mathbf{v}_i$ we provide the maximum length of the edges connected to $\mathbf{v}_i$.

Figure 5.14: Simplified mesh up to 40% of the initial number of vertices, on the left the one obtained with SIMPL, on the right the one obtained with SIMPLSTEP, for each node $\mathbf{v}_i$ we provide the maximum length of the edges connected to $\mathbf{v}_i$.

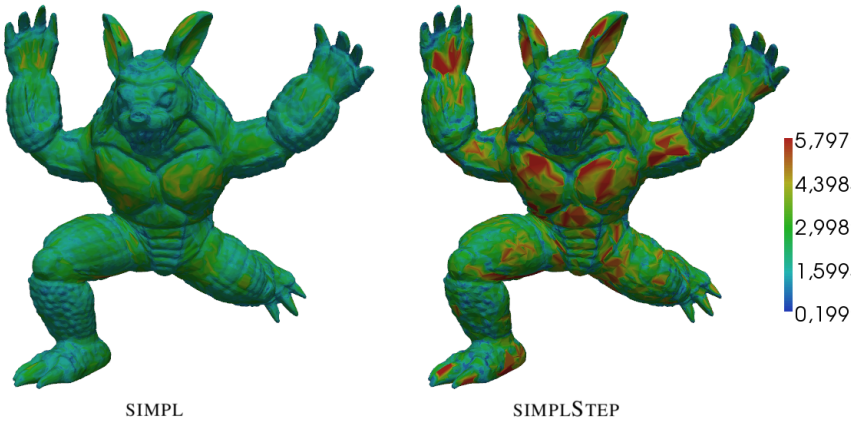


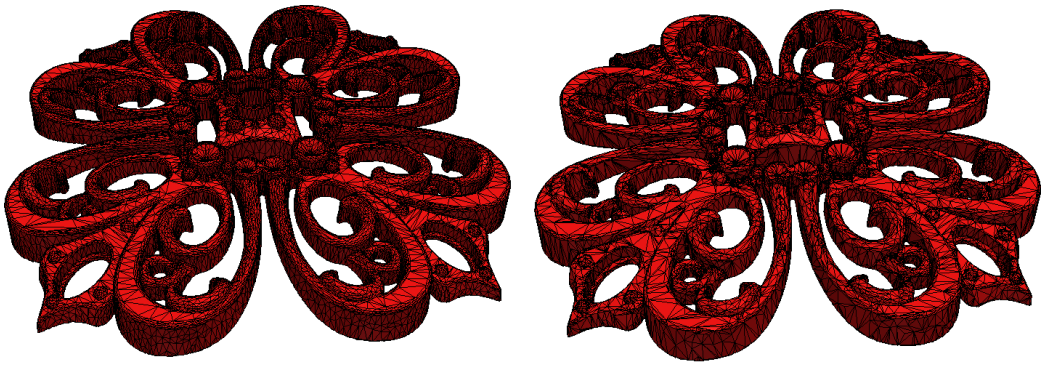
Figure 5.15: Simplified mesh up to 20% of the initial number of vertices, on the left the one obtained with SIMPL, on the right the one obtained with SIMPLSTEP, for each node $\mathbf{v}_i$ we provide the maximum length of the edges connected to $\mathbf{v}_i$.

Figure 5.16: Simplified mesh up to 5% of the initial number of vertices, on the left the one obtained with SIMPL, on the right the one obtained with SIMPLSTEP, for each node $\mathbf{v}_i$ we provide the maximum length of the edges connected to $\mathbf{v}_i$.

Example 5: the filigree geometry

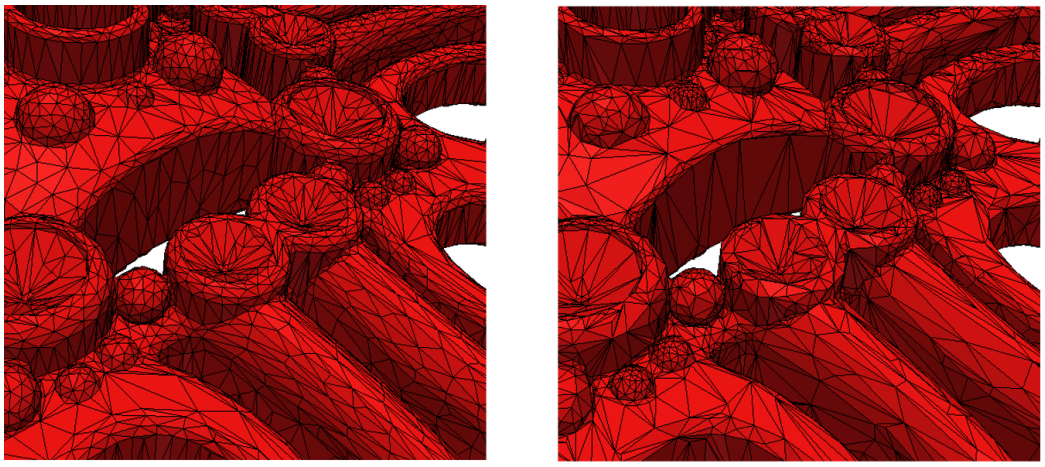
In this example we consider a very complex geometry with about half a million vertices. As we can see from, e.g., Figure 5.17, both simplification algorithms produce a mesh that matches the geometry of the surface.

From the detail on the left in Figure 5.18, 5.19 and 5.20, we observe that SIMPL generates triangles whose shape, orientation and size reflect the geometry of the input surface. On the contrary, in some regions of the mesh obtained by SIMPLSTEP, the triangles **do not** reflect the actual curvature of the surface. We can recognize this behavior from the details in Figure 5.18 and 5.19 on the right and this trend becomes more evident in Figure 5.20 on the right. In this last figure the mesh simplified with SIMPL has all the triangles oriented according to the curvature of the surface while in the mesh obtained with SIMPLSTEP, see Figure 5.19 on the right, we do not have the same well shaped triangles, even if the geometry is fitted.



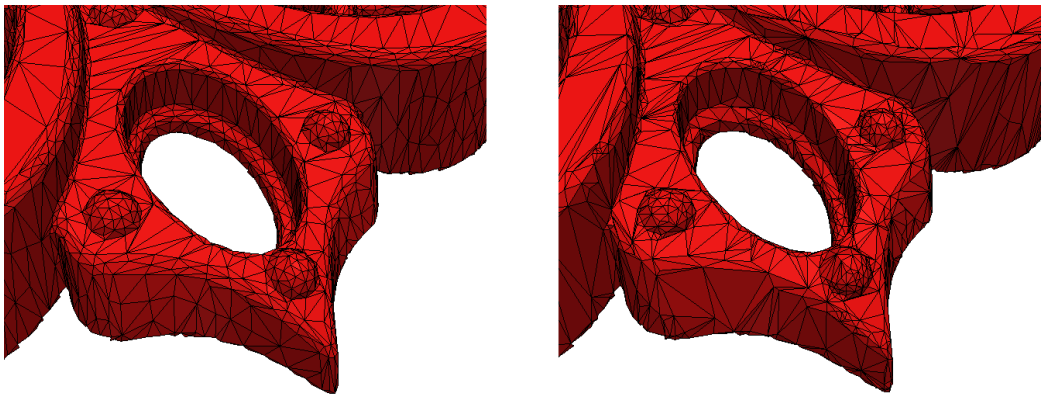
SIMPL SIMPLSTEP

Figure 5.17: Simplified mesh up to 5% of the initial vertices; on the left the one obtained with SIMPL; on the right the one obtained with SIMPLSTEP.



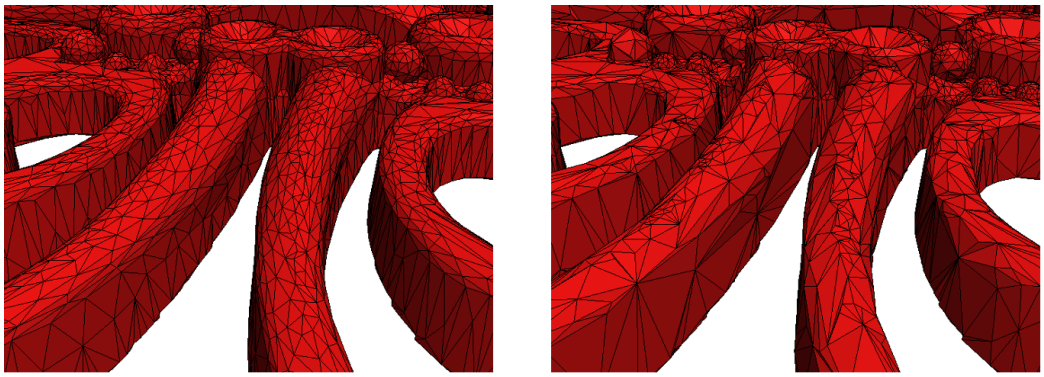
SIMPL SIMPLSTEP

Figure 5.18: Detail of the simplified mesh up to 5% of the initial vertices; on the left the one obtained with SIMPL; on the right the one obtained with SIMPLSTEP.



SIMPL SIMPLSTEP

Figure 5.19: Detail of the simplified mesh up to 5% of the initial vertices; on the left the one obtained with SIMPL; on the right the one obtained with SIMPLSTEP.



SIMPL SIMPLSTEP

Figure 5.20: Detail of the simplified mesh up to 5% of the initial vertices; on the left the one obtained with SIMPL; on the right the one obtained with SIMPLSTEP.

5.3 Overview on the Statistical Analysis

In the literature there is a large variety of works related to model simplification. In the previous sections, we have introduced a purely geometric simplification process, i.e., a mesh simplification strategy that reduces the number of elements of a polyhedral mesh and preserves the geometry as much as possible. In this section we present a new method that combines a simplification strategy with a statistical analysis. More precisely, we show an efficient technique to analyze a large noisy data set represented by the thickness of the brain.

This data and the corresponding mesh are obtained via magnetic resonance imaging, IMR. From these images, the brain tissue is classified into white matter, gray matter and cerebrospinal fluid. Via this subdivision, it is possible to generate the inner and the outer cortical surface meshes. In particular, the inner mesh is built by finding the boundaries between white and gray matter and the outer one by finding the boundaries between the gray matter and the cerebrospinal fluid, [21]. Then, the cortical thickness is defined to be the distance between the inner and the outer meshes. From the medical point of view, these data are really important. In fact, they are linked to the pathology of many neurological disorders such as autism, Alzheimer's disease and schizophrenia, [64].

Since the brain is twisted and presents a large number of sulci, the mesh of this surface is really involved. As a consequence the mesh generation process is a complex multi-step procedure that results in a very large data set, often more than 10^6 vertices, with a corresponding data observation.

There is a large variety of work in literature that treats this kind of data analysis, see, e.g., [17]. In [18], M. Chung et al. introduce a smoothing technique, called Iterative Heat Kernel, IHK, for neuro-imaging applications. This geodesic-distance-based-kernel smoothing method solves the Laplace-Beltrami eigenvalue problem on the surface to construct a basis for the IHK directly on the cortical surface. Then, only a finite number of these basis functions is used in the expansion of the IHK. Finally, via an iterative algorithm, the number of terms in the Fourier series expansion of the IHK is properly adjusted. Another approach is the so-called Spatial Regression model for Non-Planar domain, SR-NP, proposed by B. Ettinger et al. in [35]. This approach minimizes a sum of squared error functionals with a roughness penalty term involved the Laplace-Beltrami operator associated with the non-planar domain. The estimation problem on the surface is appropriately re-casted over a planar domain via a conformal map. In the planar domain, existing spatial smoothing techniques are suitably generalized by taking into account the flattening of the domain, [98].

All these statistical analyses have to deal with a large amount of data. In literature, there are different methods for containing the computational cost associated with the analysis of the cortical surface data. In [56], D. J. Hangler et al. develop a nearest neighbor averaging scheme to overcome the high computational cost. This method smooths the variable of interest observed at each vertex of the mesh by suitably averaging this value with the ones observed at the neighborhood.

In this section we present a new strategy to reduce the computational cost of the analysis of the cortical thickness data. The basic idea behind is to reduce the number of elements before applying the SR-NP approach resorting to a new mesh simplification algorithm which merges geometric with statistical information.

In appendix A we briefly describe the SR-NP approach.

5.3.1 Spatial Regression and Simplification Strategy

Here we describe in more details the proposed statistical-simplification process, see Figure 5.21. We consider a non planar surface $\Gamma \in \mathbb{R}^3$ and a measurement of the thickness of the brain. We approximate the surface Γ with a triangular surface mesh, Γ_h. On each vertex, $\mathbf{x}_j$ of Γ_h we consider the thickness of the cortical surface as a scalar value z_j. We assume the following model for the data

$$z_j = f(\mathbf{x}_j) + \epsilon_j\,, \quad \forall j = 1,\, 2, \ldots n\,, \tag{5.6}$$

where ϵ_j are independent observational errors with zero mean and constant variance, while f is a twice continuously differentiable real-valued function defined on Γ. Since the data z_j are affected by the measurement error, ϵ_j, we have to find a proper approximation of the function f moving from the data z_j. To achieve this goal, we simplify the mesh Γ_h to a mesh Γ_h' with a lower number of vertices. During this process, we carefully track the data z_j on the new mesh. Then we flatten the simplified mesh via a conformal map, following the theory provided by B. Ettinger et al. in [35]. Finally, we apply a standard spatial regression analysis that properly takes into account this transformation to find a suitable approximation $\tilde{f}$ of f and we map back $\tilde{f}$ to the real three-dimensional domain. Figure 5.21 gives an outline of the whole process.

5.3.2 Mesh Point vs Data Point

Consider a triangulated surface Γ_h embedded in $\mathbb{R}^3$, where a scalar quantity has been observed at each vertex of the mesh via the values z_j, for $j = 1, \ldots, n$. In this framework, there are two different kinds of points:

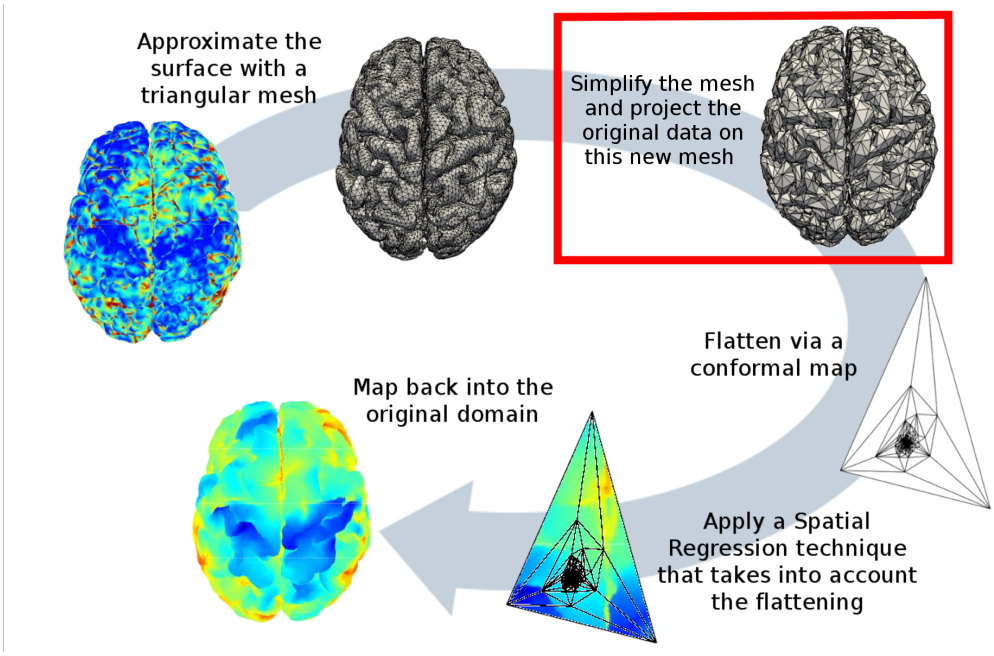


Figure 5.21: Scheme of the proposed statistical-simplification process, in the red box it is highlighted the simplification step.

- **mesh points**, i.e., the vertices of the mesh triangles, blue circles in Figure 5.22;
- **data points**, i.e., the points on the polyhedral surface that are related to a data, red squares in Figure 5.22.

Remark 5.3.1 *At the beginning of the statistical simplification procedure, the mesh points coincide with the data points.*

One of the possible strategies is to simplify the whole mesh, using the routine SIMPL and then to project all the data points on this new mesh. This operation could be not so straightforward due to the curvature of the surface Γ_h.

In fact, the closest point to the original data is **not** necessary the one where we have to project the data point. In Figure 5.23 we give an example of this issue, we show a cross-section of an original mesh, solid lines, and the new mesh, dashed line. Suppose that segment e_1 replaces the segments e_2 and e_3. The correct projection of the data point **a** is **b** and not in **c** even if **c** is the closest point to **a**.

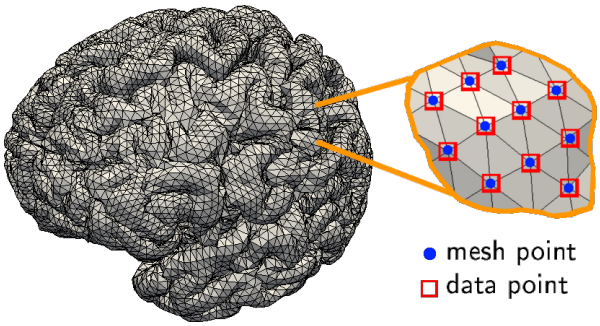


Figure 5.22: Example of surface with mesh points, blue circles, and data points, red squares.

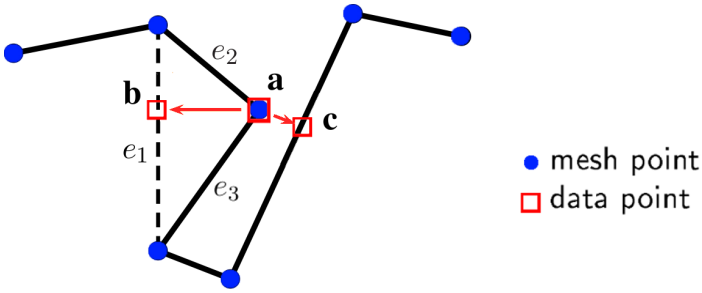


Figure 5.23: Cross-section of the original mesh, solid lines; the new mesh, dashed line, replaces segments e_2 and e_3 with the segment e_1.

The geometry of the brain is characterized by many sulci, see Figure 5.24. So, we cannot simplify the mesh and then project all the data, because we may have many configurations like the ones in Figure 5.23. To overcome this issue, we decide to carefully track the data associated with the mesh. In particular, once we have contracted an edge of the mesh, we properly project all the data points associated with the triangles involved in the contraction.

5.4 Edge Contraction Issues

We have seen in Subsection 1.2.3, that the contraction of a generic edge e may lead to topological undesired configurations. In this new simplification procedure, we are dealing with a complex geometry and a statistical analysis, so we have to add new constraints to the edge contraction.

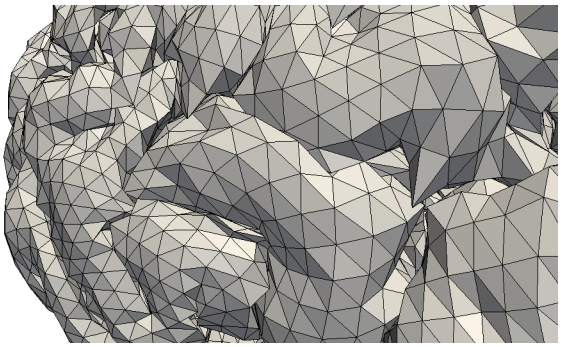


Figure 5.24: Example of sulci of the brain.

Geometry Considerations

In this mesh simplification procedure, we are dealing with the geometry of the brain. Besides its huge number of elements, this kind of geometry is really complex due to the high folded nature of the brain. As you can see from Figure 5.24, the triangles are really close each other, so that a contraction of an edge e may lead to an intersection between the new generated triangles with the neighboring elements, see Figure 5.25.

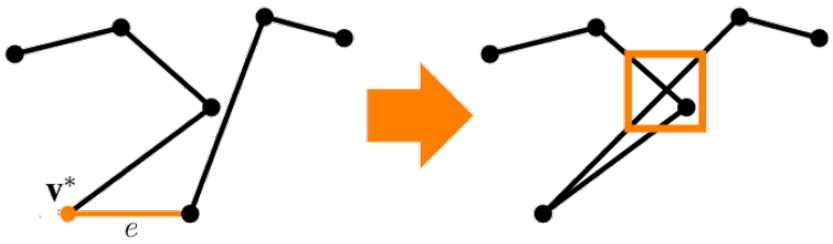


Figure 5.25: Example of self-intersection due to the nature of the sulci. The algorithm tries to contract the edge e into the node $\mathbf{v}^*$, left, but this operation yields a self-intersection, right.

We use the search data structure and the intersection test described in Chapter 6, to overcome this problem. In particular, once we have decided to contract an edge e, we check if the new configuration led by this contraction generates a self-intersected mesh, and then we decide to allow or not the contraction of e.

Statistical Considerations

To apply the flattening map, it is necessary to keep a triangle unchanged, the reasons of this demand are detailed in [35]. We consider a triangle T_{fix} and its vertices $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$. To maintain this triangle unchanged during the simplification procedure, we do not allow any contraction that involves an edge whose endpoints coincide with $\mathbf{v}_1$, $\mathbf{v}_2$ or $\mathbf{v}_3$, see Figure 5.26.

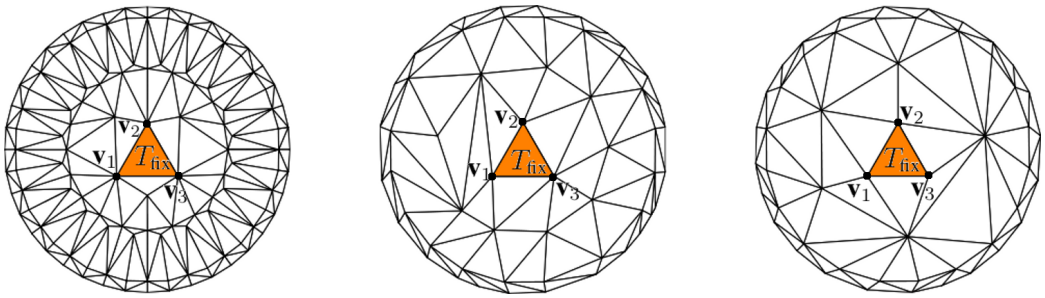


Figure 5.26: Example of different levels of simplification that preserve an element of the mesh, the orange triangle.

During the simplification process, it is not a priori guaranteed that, after any contraction all the triangles involved still have at least one data point associated with. An **empty** triangle, i.e., a triangle with no data point, is useless for the subsequent spatial regression analysis, since it does not give any further information about the function f we are estimating. For this reason, we do not allow any contraction of an edge e that generates empty triangles.

Final Considerations about Edge Contraction

To clarify the constrains on the contraction of a generic edge e of the mesh, we collect all the conditions that a contraction has not to verify:

(a) produce any topological undesired configuration, see Subsection 1.2.3;

(b) form any self mesh intersection;

(c) modify a fixed triangle T_{fix};

(d) bring to empty triangles.

To satisfy all these conditions, for each edge e of the mesh we consider several locations for the new vertex $\mathbf{v}^*$. In particular, we try to contract e onto: the optimal location defined in Subsection 5.2.1, the middle point of e and one of its endpoints.

Remark 5.4.1 *In this new framework a **valid edge** is an edge that does **not** verify all the conditions (a), (b), (c) and (d).*

5.5 A Mesh Simplification Strategy for Spatial Regression Analysis

In this framework the simplification procedure is the same as the one proposed in Section 5.2. We follow exactly the same routine shown in Algorithm 5, but now we take into consideration a different edge cost function. In fact, since we are interested both in the geometry of the surface and in the subsequent statistical analysis, the cost function has to take into account both the geometric approximation of the mesh and the data associated with the mesh. Hence, for a generic edge e of the mesh, we define this new contraction cost:

$$c(e, \mathbf{v}^*) := \alpha c_{\text{geo}}(e, \mathbf{v}^*) + \beta c_{\text{data}}(e, \mathbf{v}^*)\,, \tag{5.7}$$

where $\mathbf{v}^*$ is the vertex that replaces the edge e, while $c_{\text{geo}}(e, \mathbf{v}^*)$ and $c_{\text{data}}(e, \mathbf{v}^*)$ represents the geometric cost and the data cost, respectively. In particular, $c_{\text{geo}}(e, \mathbf{v}^*)$ is the function defined in Equation (5.3). Similarly, $c_{\text{data}}(e, \mathbf{v}^*)$ is defined to represent the loss of good properties for the subsequent statistical analysis in terms of the displacement and distribution of the data points over the new mesh. The weights $\alpha, \beta \in \mathbb{R}^+$ balance each function contribution to the overall contraction cost.

5.5.1 The Data Cost

The actual novelty of the proposed algorithm lies in incorporating the data points into the simplification process. Before dealing with this new feature of the mesh simplification process, let us make some further considerations about the data projection phase of the process. Each data point may be projected onto the new mesh Γ_h' in one of the following ways:

- on the face of a triangle of Γ_h', see data points 1 and 2 in Figure 5.27;

- on an edge between two triangles of Γ'_h, see data points 3 and 4 in Figure 5.27;

- on a vertex of Γ'_h, see data points 5 and 6 in Figure 5.27.

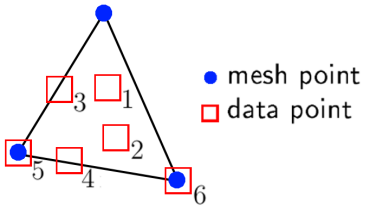


Figure 5.27: Triangle with all the possible association.

After the projection procedure, the data points are associated with their projection on Γ'_h. For the statistical analysis that follows the mesh simplification, it is crucial to properly take into account the data association with the mesh. To achieve this goal, we consider:

i) the displacement of the data points, i.e., the distance between the projected data locations and their original locations;

ii) the equidistribution of the data points over the triangles of the new mesh Γ'_h, i.e., each triangle of Γ'_h should be associated with about the same number of data points.

To take care of both these aspects, we introduce two suitable cost functions, one for each desired feature. Thus, the total data cost function is given by

$$c_{\text{data}}(e, \mathbf{v}^*) := \beta_1 c_{\text{disp}}(e, \mathbf{v}^*) + \beta_2 c_{\text{equi}}(e, \mathbf{v}^*), \tag{5.8}$$

where β_1 and β_2 are positive real numbers that properly weight the contributions of the data point displacement and of the data distribution.

Data displacement function

When the edge e is contracted into the point $\mathbf{v}^*$, we define the corresponding displacement cost function as

$$c_{\text{disp}}(e, \mathbf{v}^*) := \max_{(p,\, q) \in \mathcal{P}_{\text{new}} \times \mathcal{P}_{\text{orig}}} ||p - q||\,, \tag{5.9}$$

which essentially measures the maximum Euclidean distance between the projected locations of the data points, $\mathcal{P}_{\text{new}}$, and their original locations $\mathcal{P}_{\text{orig}}$. By minimizing the displacement of the data associations during the contraction process, we are able to reduce the error between the statistical estimates that use the original data points on Γ_h and the statistical estimates based on the data points associated with the simplified mesh Γ_h'.

Data distribution function

Our goal is to obtain, at the end of the simplification process, a new mesh whose elements contain about the same number of data points, i.e., to equidistribute the data points over the new mesh. For this purpose, during the simplification process, we define, for each triangle T, the quantity

$$N_T := n_f + \frac{1}{2} n_e + \frac{1}{\#\omega_{\mathbf{v}_1}} n_{\mathbf{v}_1} + \frac{1}{\#\omega_{\mathbf{v}_2}} n_{\mathbf{v}_2} + \frac{1}{\#\omega_{\mathbf{v}_3}} n_{\mathbf{v}_3} \,, \tag{5.10}$$

where n_f and n_e denotes the number of data points associated with the face and with the edges of the triangle T, respectively, n_j is the number of data points associated with the $j-$th vertex $\mathbf{v}_j$ of T, for $j = 1, 2, 3$, $\omega_{\mathbf{v}_j}$ is the patch of elements associated with $\mathbf{v}_j$ and $\#\omega_{\mathbf{v}_j}$ denotes its cardinality.

Via N_T, we compute the mean value $\overline{N}$ of N_T over the entire mesh for the current iteration of the simplification process. Then, when we contract the edge e into $\mathbf{v}^*$, we compute the quantity N_T for all the triangles involved in the contraction of e and we evaluate the equidistribution cost function

$$c_{\text{equi}}(e, \mathbf{v}^*) := \frac{1}{\#\mathcal{T}_e} \left(\sum_{T \in \mathcal{T}_e} \left(N_T - \overline{N} \right)^2 \right) \,, \tag{5.11}$$

where $\mathcal{T}_e$ is the set of triangles involved in the contraction of the edge e. This quantity measures the variation with respect to $\overline{N}$ in the distribution of the number of data points associated with the contraction of the edge e into the vertex $\mathbf{v}^*$.

5.5.2 Combination of the Geometric and of the Data Costs

One way to combine the cost functions defined in Equation (5.3), (5.9) and (5.11) is to consider the global cost

$$c(e, \mathbf{v}^*) := \alpha c_{\text{geo}}(e, \mathbf{v}^*) + \beta_1 c_{\text{disp}}(e, \mathbf{v}^*) + \beta_2 c_{\text{equi}}(e, \mathbf{v}^*) \,. \tag{5.12}$$

But, since these three functions may have different ranges, they have to be properly scaled. We decide to normalize these three cost functions with respect to their corresponding maximum, in order to consider values for $\alpha, \beta_1, \beta_2 \in [0, 1]$ such that $\alpha + \beta_1 + \beta_2 = 1$. For simplicity, we do not change the notation of these cost functions and from now on, we refer to the normalized quantities as $c_{\text{geo}}(e, \mathbf{v}^*)$ $c_{\text{disp}}(e, \mathbf{v}^*)$, and $c_{\text{equi}}(e, \mathbf{v}^*)$.

A low value of $c(e, \mathbf{v}^*)$ means that the contraction will yield a good geometric approximation to the original geometry, where the data points are close to their original locations and evenly distributed throughout the triangles of the new mesh. On the contrary, a high value of $c(e, \mathbf{v}^*)$ means that the contraction will produce a bad approximation of the original surface, or that the projected data points are too far from their original locations or that there might be triangles with too many or too few data points associated with.

As the mesh simplification algorithm described before, if we iteratively remove the edge of the mesh characterized by the lowest cost, we obtain a new mesh that would satisfy all the desired properties.

5.5.3 Numerical results

In this subsection, we numerically check the performances of the spatial regression analysis, see Figure 5.21. In particular, the goal is to verify that the mesh simplification described in this section produces good statistical estimates. For this purpose, we compare the SR-NR combined with mesh simplification this approach with the Iterative Heat Kernel (IHK) smoothing proposed in [18] and the Spatial Regression for Non-Planar domain (SR-NP) in [35]

We use two mesh simplification strategies with several levels of simplification. In particular, we consider these different choices of the parameters α, β_1 and β_2:

- Data+Geo (D+G): the simplification is obtained by equally weighting the geometric, the displacement and the distribution cost functions, i.e., we choose $\alpha = \beta_1 = \beta_2 = 1/3$, in Equation (5.12);
- OnlyGeo (O/G): the simplification is driven only by the geometric information, this is equivalent to $\alpha = 1$ and $\beta_1 = \beta_2 = 0$ in Equation (5.12).

For each example we generate simulated data on the starting mesh. We consider fifty test functions of the form

$$f_i(x,\, y,\, z) := a_i \sin(2\pi x) + b_i \sin(2\pi y) + c_i \sin(2\pi z) + 1 \quad \forall i = 1, \ldots 50\,. \tag{5.13}$$

with coefficients a_i, b_i and c_i randomly generated from independent normal distribution with mean 1 and standard deviation 1, where the data point locations coincide with the vertices of the original mesh. To get a noisy data, $z_{i,j}$, we add independent normally distributed errors with mean 0 and standard deviation 0.5 to each f_i, i.e., we consider

$$z_{i,j} = f_i(x_j, y_j, z_j) + \epsilon_{i,j}\,, \quad \forall j = 1, 2, \dots n\,, \tag{5.14}$$

where n is the number of vertices of the initial mesh Γ_h and $\epsilon_{i,j}$ is a random number generated by independent normal distribution with mean 0 and standard deviation 0.5.

Moving from the data $z_{i,j}$, the proposed strategy provides a smooth approximation of f_i, i.e., it creates a new function $\tilde{f}_i$ which is not affected by noise. Since we know each function f_i, we may proceed with an a-priori error analysis, so we evaluate the effectiveness of the proposed strategy looking at the mean squared error between the exact functions, f_i, and the corresponding estimates, $\tilde{f}_i$.

Example 1: the pawn test case

In Figure 5.28 we show a simulation example. In particular, we show a function f_i, Figure 5.28 (a), the noisy function Figure 5.28 (b), the IHK estimate computed on the original mesh, Figure 5.28 (c), the SR-NP estimate using the mesh with 1000 vertices yielded by the Data+Geo simplification, Figure 5.28 (d) and, finally, the SR-NP estimate using the mesh with 1000 vertices with the simplification OnlyGeo, in Figure 5.28 (e).

We observe that, despite using less than an half of the initial vertices, the SR-NP method is able to detect the variation of the function; on the base of the pawn this behavior is more evident. Furthermore, the Data+Geo simplification maintains an even level of smoothing over the pawn more than the OnlyGeo simplification does. In fact, the mesh yielded by the OnlyGeo simplification is able to locate the high variation on the base of the pawn, but it over-smoothes on the top left hand side of the pawn.

The superior performance of the SR-NP method combined with the Data+Geo simplification is more evident in Table 5.3 and Figure 5.29, where more quantitative information can be inferred. In particular, for each mesh simplification and simulation replication, we compute the Mean Squared Error (MSE) of the estimate which measures the mean square distance between the true function f_i and its estimate $\tilde{f}_i$. A lower MSE means a more efficient estimate, characterized by lower bias, i.e., lower systematic errors and lower variance.

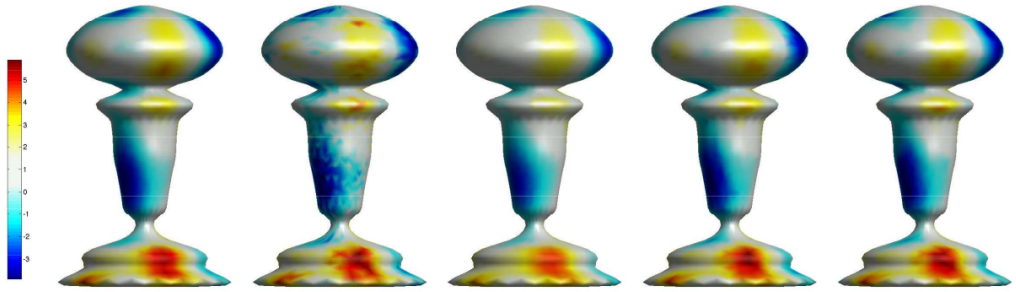


Figure 5.28: A simulation example on the pawn geometry: (a) example of one test function, (b) the function with noise, (c) the IHK estimate using the original mesh, (d) the SR-NP estimate using 1000 vertices with the Data+Geo simplification, (e) the SR-NP estimate using 1000 vertices with the OnlyGeo simplification.

Method	m	Simplification	MSE (IQRs)	SR-NP vs IHK
SR-NP	1000	Data+Geo	0.0662 (0.0281)	0.0447
		OnlyGeo	0.0903 (0.0494)	0.7316
SR-NP	1200	Data+Geo	0.0569 (0.0226)	0.0015
		OnlyGeo	0.0839 (0.0489)	0.4386
SR-NP	1400	Data+Geo	0.0453 (0.0206)	2.207e-8
		OnlyGeo	0.0698 (0.0483)	0.0963
SR-NP	1600	Data+Geo	0.0443 (0.0198)	3.100e-9
		OnlyGeo	0.0528 (0.0285)	1.801e-5
SR-NP	1800	Data+Geo	0.0447 (0.0207)	2.762e-9
		OnlyGeo	0.0467 (0.0190)	1.979e-8
SR-NP	2000	Data+Geo	0.0416 (0.0208)	4.139e-10
		OnlyGeo	0.0378 (0.0163)	3.895e-10
SR-NP	2527	none	0.0351 (0.0148)	3.895e-10
IHK	2527	none	0.0717 (0.0978)	

Table 5.3: Median MSEs, IQRs and p-values of pairwise Wilcoxon tests for different simplified meshes and different cost functions, comparing SR-NP with IHK.

In Table 5.3, we provide the median MSE computed over the fifty simulation replications and, within parentheses, the corresponding Inter Quartile Range (IQRs) that gives an idea about the spread of the MSEs over the fifty replicates. This information is also illustrated via box plots in Figure 5.29 which macro-

scopically simplify the comparison among the different methods. Table 5.3 and Figure 5.29 highlight that, as expected, the SR-NP method with the simplification based on the global cost function produces better results than the SR-NP method with the standard simplification driven by geometric information only.

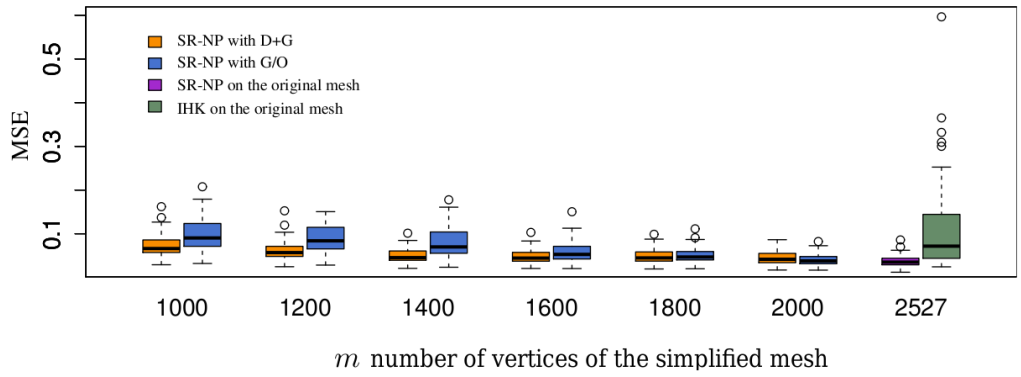


Figure 5.29: Box-plots of the mean square errors of the simulation results for fifty test functions.

In particular, for all levels of mesh simplification except for the case $m = 2000$, i.e., the most simplified mesh, both the median MSE and the corresponding IQRs associated with the D+G approach are lower, corresponding to more accurate and more reliable estimates. In the last row of Table 5.3 we compare the SR-NP approach with the IHK method on the original mesh. The SR-NP method combined with the simplification strategy driven by both data and geometry control produces better results in terms of estimates, with lower error, the MSE has a lower median, and more robustness, the MSE has a smaller IQR. On the other hand, we observe that if we combine the SR-NP method with the simplification strategy driven only by the geometric information, we need a mesh with at least 1400 vertices to get an estimate with a median MSE lower than one associated with the IHK method. This is a first confirmation of the importance to include the data information in the mesh simplification procedure. Now, to quantitatively verify these results, we use pairwise Wilcoxon tests, see [111]. The pairwise Wilcoxon test is a non-parametric statistical hypothesis test that is used here to assess whether the MSEs of SR-NP estimates are significantly lower than the MSEs of the IHK estimates. The p-value for this hypothesis test is the probability that the MSEs of the IHK estimates are lower than the MSEs of the SR-NP estimates. Thus, the lower the p-values for this test the stronger the statistical evidence that

the MSEs distribution for the SR-NP estimators are stochastically lower than the corresponding distribution for the IHK estimate. In the last column of Table 5.3 we provide the p-values for pairwise Wilcoxon tests. These p-values verify that the estimates obtained via the SR-NP method on the original mesh, consisting of 2527 vertices, and on all the simplified meshes generated via the Data+Geo simplification have significantly lower MSEs than the estimates obtained via the IHK method. Concerning the G/O simplified meshes, at least $m \geq 1600$ vertices are needed to produce significantly low MSEs.

Example 2: the brain test case

In this example, we apply the proposed approach to a cortical surface geometry. In this case, we are dealing with a original mesh with 40962 nodes. First, we apply the proposed method to a simulation study. Then, we apply the same approach to real cortical surface thickness data studied in M. Chung et al. in [17] and [18].

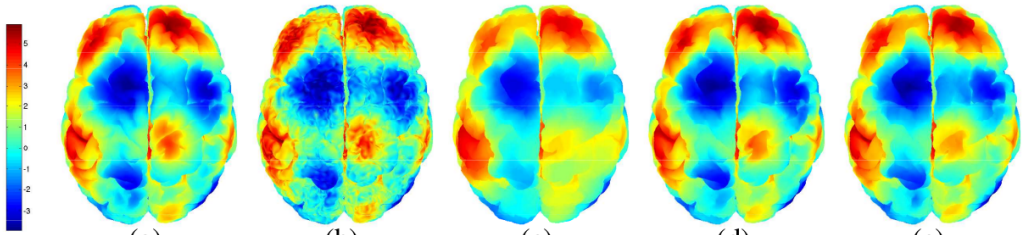


Figure 5.30: A simulation example on the brain geometry: (a) example of one test function, (b) the function with noise, (c) the IHK estimate using the original mesh, (d) the SR-NP estimate using 10000 vertices with the Data+Geo simplification, (e) the SR-NP estimate using 10000 vertices with the OnlyGeo simplification.

As in the previous example, we simulate noisy data on the cortical surface mesh by generating fifty test functions of the form described in Equation (5.13) and by adding independent normally distributed errors with mean zero and a standard deviation 0.5 to the function values at each of the data locations, see Equations (5.14). As before, we compare the SR-NP method based on the two mesh simplification strategies for different levels of simplification to the IHK results on the full mesh. For the SR-NP method, we consider three levels of mesh simplification: $m = 10000, 15000, 20000$ vertices for both the Data+Geo and the OnlyGeo simplifications. Here, we do not give the SR-NP estimate over the original cortical

Method	m	Simplification	MSE (IQRs)	SR-NP vs IHK
SR-NP	10000	Data+Geo	0.0383 (0.0427)	4.399e-10
		OnlyGeo	0.0501 (0.0614)	4.399e-10
SR-NP	15000	Data+Geo	0.0332 (0.0310)	5.275e-10
		OnlyGeo	0.0473 (0.0540)	4.674e-10
SR-NP	20000	Data+Geo	0.0328 (0.0281)	5.951e-10
		OnlyGeo	0.0432 (0.0476)	5.275e-10
IHK	40962	none	0.1349 (0.2662)	

Table 5.4: Median MSEs, IQRs and p-values of pairwise Wilcoxon tests for different simplified meshes and different cost functions comparing SR-NP with IHK.

surface mesh because it is too computationally expensive.

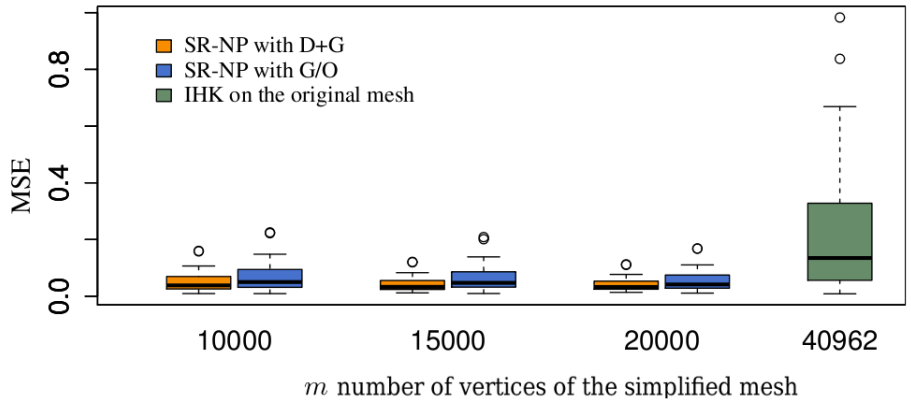


Figure 5.31: Box-plots of the mean square errors of the cortical surface simulation results of fifty test functions.

Table 5.4 collects the median MSEs and the corresponding IQRs over the fifty simulation replicates. As expected, the SR-NP method produces better results than the IHK method, while using less than half of the original nodes. Again, the SR-NP estimates associated with the Data+Geo simplification are better than the estimates given by the OnlyGeo simplification. The low p-values of pairwise Wilcoxon tests verify that the distribution of MSEs for the SR-NP estimators are stochastically lower than the corresponding distribution for the IHK estimate.

Figure 5.31 displays the box plots of the MSE values in Table 5.4. We recognize the same trend as in Figure 5.29, where the Data+Geo simplification produces

excellent results using fewer nodes. Figure 5.30 shows an example of a simulation replicate: an example of a test function generated via Equation (5.13) in (a), the corresponding level of noise in (b), the IHK estimate obtained on the original mesh in (c), the SR-NP estimate computed on a mesh with 10000 nodes and via the Data+Geo simplification in (d) and the SR-NP estimate on a mesh with 10000 nodes generated via the OnlyGeo simplification in (e). The SR-NP method is better at picking up variation in the data. This is most evident in the right hemisphere of the cortical surface.

In Figure 5.32 we show the initial mesh and the simplified grid up to a number of nodes equal to 10000. From this figure and the details in Figure 5.33 and 5.34, we can recognize that the involved geometry of the brain is well preserved even if it is composed by only quarter of the initial nodes.

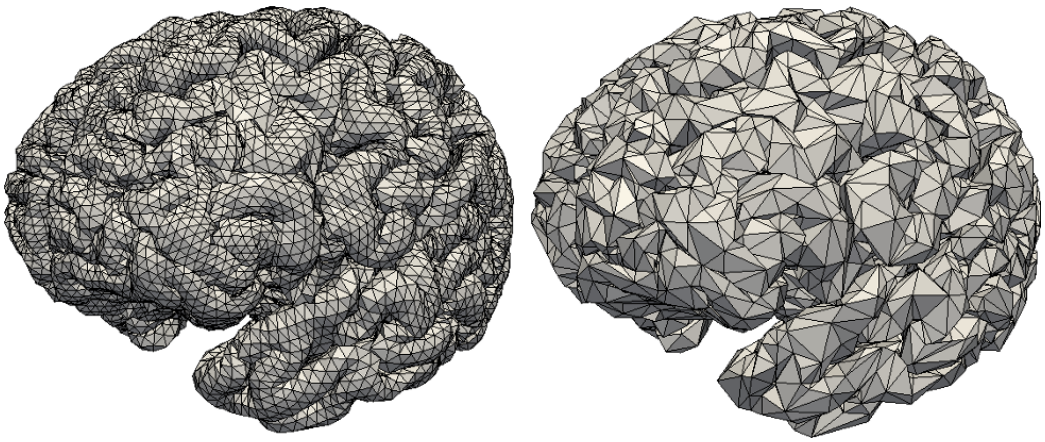


Figure 5.32: On the right the initial mesh, on the left the simplified mesh characterized by 10000 nodes.

Example 3: the brain test case on real data

Now, let us consider real cortical surface thickness data. Notice that the SR-NP method with the Data+Geo simplification is able to identify an additional area of the low thickness, circled in Figure 5.35 (c), with respect to what is detected by the IHK approach and by the same SR-NP method with the OnlyGeo simplification. This low thickness area is recognizable in the original thickness data, see Figure 5.35 (a).

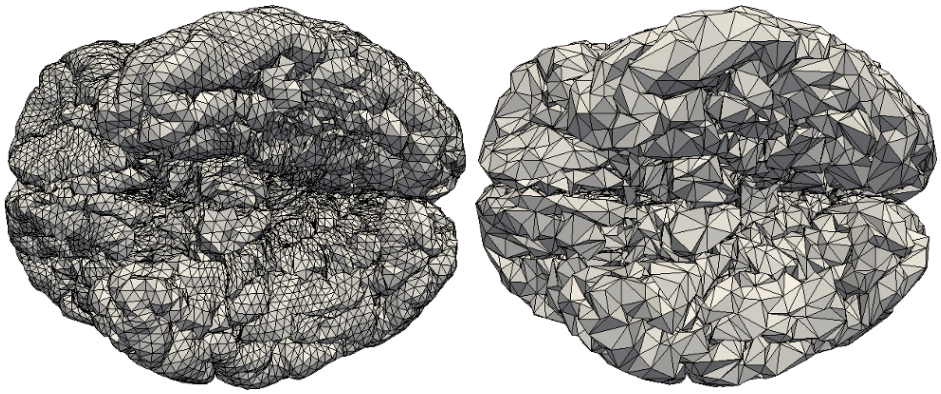


Figure 5.33: On the right a detail of the initial mesh, on the left the same detail for the simplified mesh constructed by 10000 nodes.

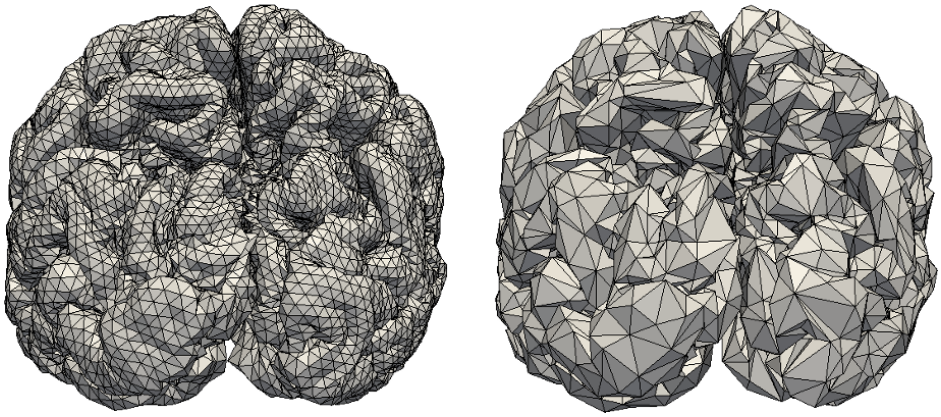


Figure 5.34: On the right a detail of the initial mesh, on the left the same detail for the simplified mesh constructed by 10000 nodes.

5.5.4 Sensitivity Analysis

In the previous examples we have considered two different choices of the parameters α, β_1 and β_2 in Equation (5.7). In particular, we have fixed $\alpha = \beta_1 = \beta_2 = 1/3$ to get a simplification procedure that put the same emphasis on all the desired aspects, i.e., the geometric approximation of the surface, the data displacement and the data distribution. Then, we have considered the alternative choice $\alpha = 1$ and $\beta_1 = \beta_2 = 0$, to obtain a simplification procedure that considers only the geometric approximation of the surface.

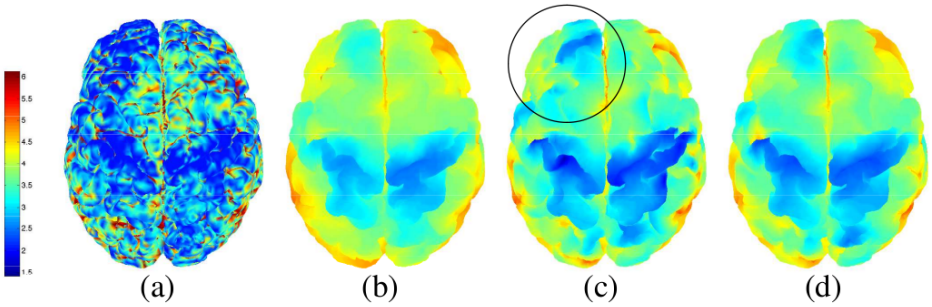


Figure 5.35: Thickness data on the original cortical surface mesh, in (a); the IHK estimate computed on the original mesh, (b); the SR-NP estimate using the 10000 node Data+Geo simplification, (c); the SR-NP estimate using the 10000 node OnlyGeo simplification, (d).

This preliminary study has numerically shown the importance of the data terms in the cost function (5.7) to get a good approximation of function f_i via the observations $z_{i,j}$. To improve this approximation, it could be interesting to make a more rigorous analysis on the possible values for α, β_1 and β_2.

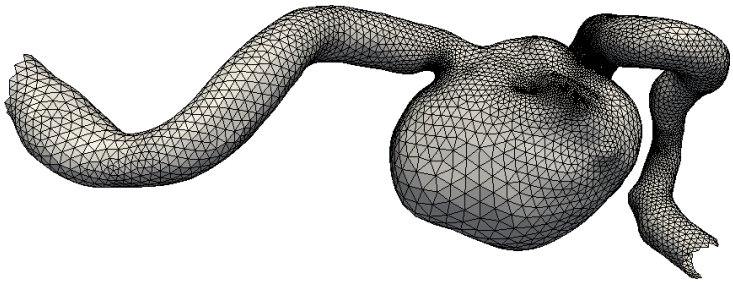


Figure 5.36: Mesh of the carotid artery with an aneurysm used for the sensitivity analysis of α, β_1 and β_2.

First, we apply the proposed procedure on a more simple model with respect to the brain test case: the carotid artery shown in Figure 5.36. As before we consider a set of fifty noisy functions given by Equation (5.14), characterized by the presence of a big aneurysm. The initial mesh is composed by 8303 vertices. We simplify it up to 3800 vertices and then we proceed with the SR-NP spatial regression analysis. To evaluate the best values for the coefficients α, β_1 and β_2, we consider these quantities:

- the maximum number of data points associated with a triangle T of the simplified mesh, i.e.,
$$N_{T,\max} := \max_{T \in \Gamma_h} N_T \,; \tag{5.15}$$
- the Mean Squared Error (MSE) of the functions taken into account and the relative Inter Quartile Range (IQR).

In particular, we look for a simplification method that produces the lowest values for $N_{T,\max}$, MSE and IQR. In fact, low values of MSE and IQR mean that the estimate function $\tilde{f}$ is a good approximation of the exact function f. Moreover, a low value of $N_{T,\max}$ means that the distribution for the data points is more uniform in the simplified mesh.

In Table 5.5 we report the results of this sensitivity analysis and we numerically show that the best configuration is given by

$$\alpha = 0.7 \qquad \beta_1 = 0.15 \quad \text{and} \quad \beta_2 = 0.15\,.$$

Of course a more rigorous approach for the selection of these parameters is advisable and will be investigated in the next future.

α	β_1	β_2	MSE	$N_{T,\max}$
0.9	0.05	0.05	0.2450 (0.0328)	29
0.8	0.1	0.1	0.2412 (0.0267)	22
0.7	0.15	0.15	0.2338 (0.0123)	19
0.65	0.175	0.175	0.2450 (0.0328)	24
0.33	0.33	0.33	0.2361 (0.0146)	28
0	0.5	0.5	0.2349 (0.0111)	78

Table 5.5: Median MSEs, IQRs and $N_{T,\max}$ for different simplified meshes with different values of α, β_1 and β_2.

Chapter 6

Modelling of a Sedimentary Basin

Sedimentary basins are among the best places to find petroleum and natural gas and to store nuclear waste material. In particular, salt basins that are characterized by a low permeability of salt guarantee low water leakages that are the main concern for the safety of nuclear waste storage. For this reason one of the best places for a nuclear waste depository is a salt mine.

The history of the basin has a deep impact on the characteristics of the oil generated. More in the details, the evolution and the temperature experienced by the sediments determine localization, quantity and quality of the oil. For instance, the temperature is a crucial aspect that controls the formation of oil. Moreover, oil usually floats and collects near the cap-Rock, i.e., the sealing layer that triggers the formation of oil-fields. In other terms, we need informations about the past history of the basin to have detailed information about the oil-fields.

Nowadays sedimentary basin studies have been based on the geological interpretation of experienced specialists. Geologists can outline several evolution scenarios of the basin, but we have to choose among them the one which is coherent from a physical viewpoint. Numerical simulations could provide the tool for choosing the right scenario. Moreover, it can even provide quantitative information, e.g. the stress field, that will be difficult to estimate by other means.

Moreover, a numerical simulation does not have the same technical difficulties and the large economic impact that is carried out when we are dealing with analogical experiments. Actually, it is difficult to scale correctly all the physical quantities in a relative small model. Sand-box experiments provide useful information regarding the brittle behaviour of grains, but they cannot represent all the viscous creeping mechanisms which require several million of years to produce a measurable effect. The experiments devoted to investigate the sediment rheol-

ogy are difficult to carry out, too. At the same time, it is also required to reach extreme displacements which cannot be precisely done by means of analogical experiments.

6.1 Issues on Basin Mesh Generation

Data from seismic imaging provide a description of the horizons, i.e., a set of three-dimensional surfaces that represent the deposition of different kinds of sediments, see Figure 6.1 on the left. Moving from these discrete surfaces, we are often demanded to furnish a volume discretization representing the whole sedimentary basin to proceed with a numerical simulation, Figure 6.1 on the right.

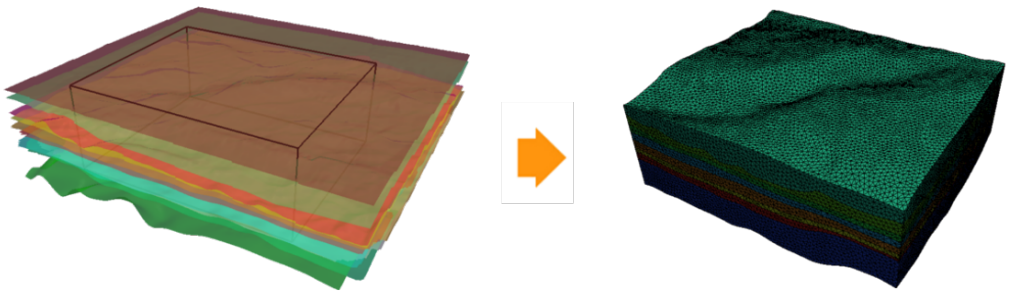


Figure 6.1: Surfaces that represent the horizons on the left and the bounding box, the black line represents the volume of interest; the computational domain on the right.

To get a proper computational domain for a numerical simulation, a lot of operations are required. In this chapter we will focus on the following ones:

a) identification of surface intersection;

b) detection of regions enclosed by this intersection and generation of a conformal mesh which includes such details;

c) mesh quality improvement.

In the geological framework, quadrilateral and hexahedral meshes are basically exploited. In the following sections we tackled the issues a), b) and c) in the case of triangular and tetrahedral meshes. In particular, we propose a new method to reduce the computational effort associated with the detection of triangular surface mesh intersection. Finally, in the last part of the chapter, we extend some recurrent geological operations to triangular surface meshes.and we properly modify them for the triangular and tetrahedral meshes.

6.2 Identification of Surface Intersection

We formalize the detection of an intersection via the simple geometric configuration illustrated in Figure 6.2. Since we are dealing with two triangulated surfaces, we are led to identify the yellow piecewise linear curve Γ as the intersection of the surfaces, see Figure 6.2 right. This curve is composed by the intersection between couples of non-coplanar triangles.

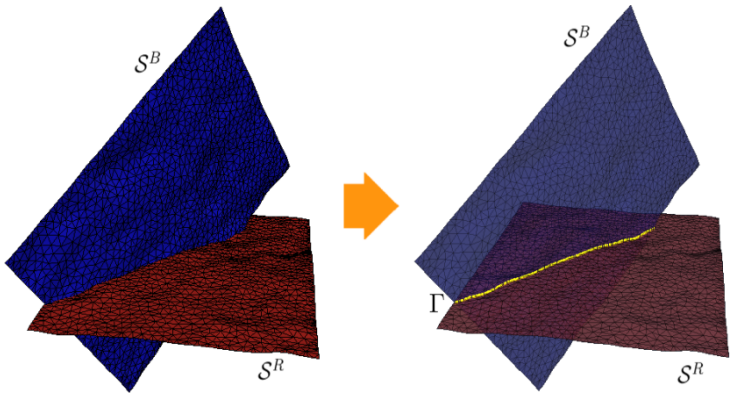


Figure 6.2: Non-coplanar surfaces, left, and corresponding intersection, the yellow line on the right.

To get an efficient procedure, we need a fast algorithm to identify the pairs of triangles intersecting each other. In fact, if we consider the surface in Figure 6.2, the most straightforward approach, which consists in checking the intersection of each triangle of $\mathcal{S}^B$ with each triangle of $\mathcal{S}^R$, is unavoidably ineffective when dealing with large data sets.

A more efficient criterion consists in finding a fast procedure to select, for each triangle $T \in \mathcal{S}^B$, a subset $\mathcal{D}_T \subset \mathcal{S}^R$ of triangles surrounding T and containing

the possible intersecting triangle(s), see Figure 6.3. In this way, we check the intersection with **fewer** elements and not with all the triangles of $\mathcal{S}^R$.

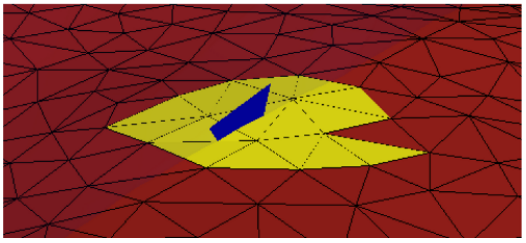


Figure 6.3: Set $\mathcal{D}_T$, the yellow area, associated with the triangle $T \in \mathcal{S}^B$.

The availability of a proper search data structure becomes crucial for a quick detection of $\mathcal{D}_T$. To increase the efficiency and the speed of this procedure, the set $\mathcal{D}_T$ should have these properties:

i) to contain the intersection $T \cap \mathcal{S}^R$;

ii) to be as small as possible;

iii) to be found rapidly.

In Subsections 6.2.1 and 6.2.2 we describe two intersection procedures based on well-established data structures. Then, in Subsection 6.2.3, we propose a new intersection algorithm that essentially merges these two approaches.

6.2.1 Structured data search

We refer to the approach proposed, e.g., in [96]. We associate the data structure with one of the triangulated surfaces, e.g., $\mathcal{S}^R$. For this purpose, we denote by $\mathcal{N}^R$ and $\mathcal{T}^R$ the set of the nodes and of the elements of the triangulated surface $\mathcal{S}^R$, respectively, and we indicate the generic three-dimensional coordinate system by $(0, x, y, z)$.

We build the **bounding box** $\mathcal{B}(\mathcal{S}^R)$ associated with $\mathcal{S}^R$, i.e., the smallest box containing the whole surface $\mathcal{S}^R$. This box is identified by the two points

$$W^R = (x_{sw}, y_{sw}, z_{sw})^t \quad \text{and} \quad NE^R = (x_{ne}, y_{ne}, z_{ne})^t$$

whose coordinates are

$$x_{sw} = \min_{P \in \mathcal{N}^R} x_p, \quad y_{sw} = \min_{P \in \mathcal{N}^R} y_p, \quad z_{sw} = \min_{P \in \mathcal{N}^R} z_p,$$
$$x_{ne} = \max_{P \in \mathcal{N}^R} x_p, \quad y_{ne} = \max_{P \in \mathcal{N}^R} y_p, \quad z_{ne} = \max_{P \in \mathcal{N}^R} z_p,$$

where $P = (x_p, y_p, z_p)^t$ is the generic node of $\mathcal{S}^R$, see Figure 6.4.

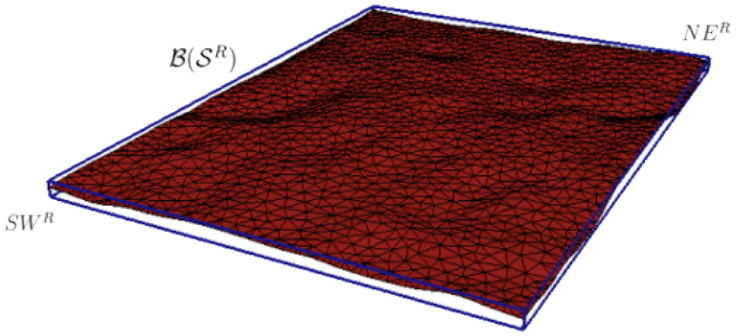


Figure 6.4: Bounding box of the surface $\mathcal{S}^R$.

We successively subdivide $\mathcal{B}(\mathcal{S}^R)$ into fixed-size sub-boxes, of dimensions

$$\begin{aligned} L_x &= \max_{K \in \mathcal{T}^R} dx_K, \\ L_y &= \max_{K \in \mathcal{T}^R} dy_K, \\ L_z &= \max_{K \in \mathcal{T}^R} dz_K, \end{aligned} \tag{6.1}$$

where

$$\begin{aligned} dx_K &= \max_{Q,R \in \mathcal{N}_K} |x_q - x_r|, \\ dy_K &= \max_{Q,R \in \mathcal{N}_K} |y_q - y_r|, \\ dz_K &= \max_{Q,R \in \mathcal{N}_K} |z_q - z_r|, \end{aligned} \tag{6.2}$$

are the dimensions of the bounding box $\mathcal{B}(K)$ associated with the generic triangle $K \in \mathcal{T}^R$, see Figure 6.7 on the left, while $Q = (x_q, y_q, z_q)^t$ and $R = (x_r, y_r, z_r)^t$ are chosen in the set $\mathcal{N}_K$ of the nodes of K.

The three-dimensional space is consequently organized into a **structured cartesian** hexahedral mesh, see Figure 6.5.

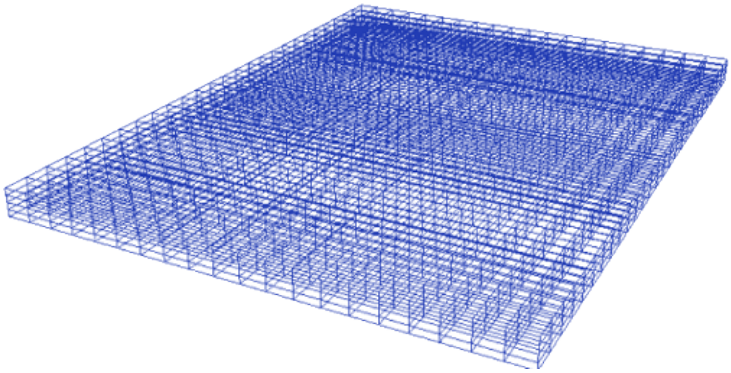


Figure 6.5: Corresponding hexahedral mesh for the surface $\mathcal{S}^R$.

Remark 6.2.1 *This spatial search structure is very flexible. In fact, it can be employed to organize any n-dimensional space after properly redefining the involved geometrical elements.*

Remark 6.2.2 *It is suited to parallelization since the hexahedral mesh can be subdivided into sub-blocks distributed among processors.*

Remark 6.2.3 *It allows us to identify the cell containing a certain point $P \in \mathbb{R}^n$ in a fast way.*

Let us exemplify the property in Remark 6.2.3. We consider a two-dimensional setting, as shown in Figure 6.6. The bounding box now coincides with the rectangle defined by the points $SW^R = (x_{sw}, y_{sw})^t$ and $NE^R = (x_{ne}, y_{ne})^t$, and it is subdivided, for instance, into N_x and N_y cells of length L_x and L_y along the x-axis and y-axis, respectively, in this particular case, we have $N_x = 5$ and $N_y = 4$.

The numbering of the cells follows a lexicographic order. This allows us to immediately find the identificator Id_P of the cell containing the point $P = (x_p, y_p)$, via the formula

$$\mathrm{Id}_P = X_{\mathrm{Id}} + Y_{\mathrm{Id}} N_x,$$

where

$$X_{\mathrm{Id}} = \left\lfloor \frac{x_p - x_{sw}}{L_x} \right\rfloor, \quad Y_{\mathrm{Id}} = \left\lfloor \frac{y_p - y_{sw}}{L_y} \right\rfloor,$$

and where $\lfloor \cdot \rfloor$ denotes the standard floor function.

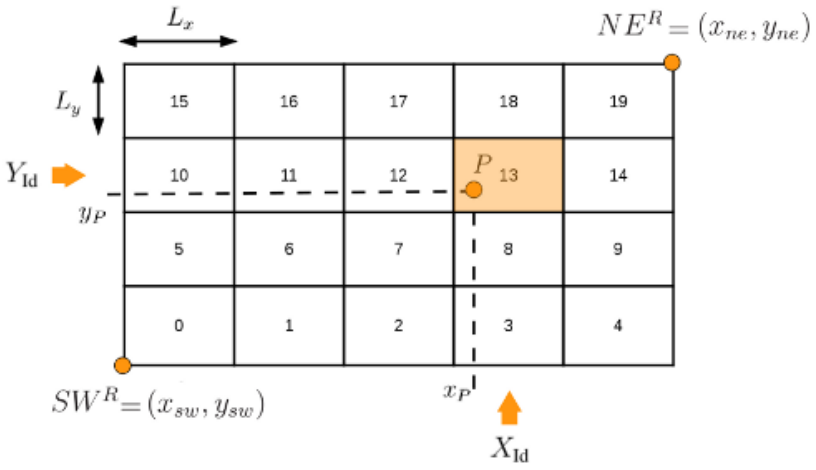


Figure 6.6: Example of direct addressing associated with structured data in a two dimensional setting.

When we are dealing with the intersection between two triangulated surfaces, we consider the space $\mathbb{R}^3$, so we can get the identifier of the hexahedral cell containing P by the formula:

$$\text{Id}_P = X_{\text{Id}} + Y_{\text{Id}} N_x + Z_{\text{Id}} N_x N_y \,, \tag{6.3}$$

where

$$X_{\text{Id}} = \left\lfloor \frac{x_p - x_{sw}}{L_x} \right\rfloor, \quad Y_{\text{Id}} = \left\lfloor \frac{y_p - y_{sw}}{L_y} \right\rfloor, \quad Z_{\text{Id}} = \left\lfloor \frac{z_p - z_{sw}}{L_z} \right\rfloor,$$

$P = (x_p, y_p, z_p)^t$ and L_z is the size along the z-axis of the sub-boxes of $\mathcal{B}(\mathcal{S}^R)$.

We now store all the triangles in $\mathcal{T}^R$ into the hexahedral mesh defined by the spacing defined in Equations (6.2) and (6.3). A priori, each sub-box may contain any number of elements, including none, see Figure 6.7, right for an example, where different elements share the same hexahedral box. For any triangle $T \in \mathcal{T}^R$, we introduce the center $C_T = (x_{C_T}, y_{C_T}, z_{C_T})^t$ of the bounding box $\mathcal{B}(T)$ as representative of T, where

$$x_{C_T} = \frac{x^T_{sw} + x^T_{ne}}{2}, \quad y_{C_T} = \frac{y^T_{sw} + y^T_{ne}}{2}, \quad z_{C_T} = \frac{z^T_{sw} + z^T_{ne}}{2},$$

where $SW^T = (x^T_{sw}, y^T_{sw}, z^T_{sw})^t$ and $NE^T = (x^T_{ne}, y^T_{ne}, z^T_{ne})^t$ are the two points identifying $\mathcal{B}(T)$. Then, the triangle T is stored in the hexahedral cell where its representative C_T falls, according to criterion (6.3). At this point it is rather easy to detect all the triangles close to a certain point P, since they are stored in the hexahedral cell containing P, whose identificator is obtained by (6.3).

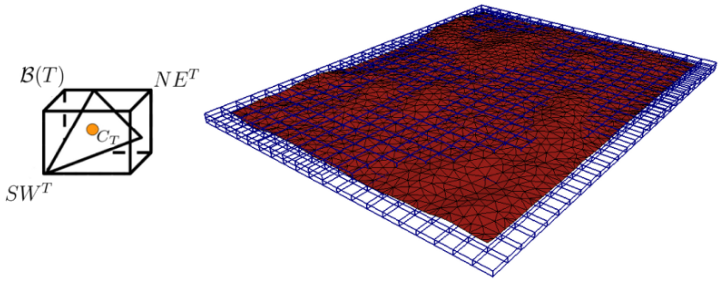


Figure 6.7: Bounding box $\mathcal{B}(T)$ with the representative C_T, left; surface $\mathcal{S}^R$ stored in the corresponding hexahedral mesh, right.

At this level, by exploiting this data structure, it is possible to construct, for each element $T \in \mathcal{S}^B$, the set $\mathcal{D}_T \subset \mathcal{S}^R$ of the triangles in $\mathcal{S}^R$ close to T. To find this set, we proceed with this work-flow:

1. we consider the bounding box $\mathcal{B}(T)$;

2. we build the set $\mathcal{H}_T$ of the hexahedral cells of $\mathcal{S}^R$ intersecting $\mathcal{B}(T)$ moving from the identificators Id_{SW^T} and Id_{NE^T} of the hexahedral cells of $\mathcal{S}^R$ containing SW^T and NE^T;

3. the set $\mathcal{D}_T$ is thus defined by the union of all the triangles contained in $\mathcal{H}_T$.

Remark 6.2.4 *The set $\mathcal{D}_T$ does not satisfy the requirement of minimal possible extension since it may include triangles that do not actually intersect the bounding box $\mathcal{B}(T)$. We refer to Figure 6.8 for an example of non optimal detection of $\mathcal{D}_T$. So useless triangle-triangle intersection tests are performed. The algorithm may be improved by performing a bounding-box intersection test before checking the actual triangle-triangle intersection.*

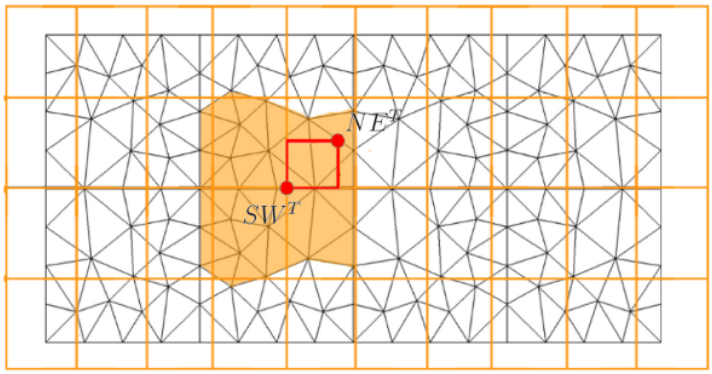


Figure 6.8: Detection of the set $\mathcal{D}_T$, the orange area, via the structured-data approach.

6.2.2 AB search

We present here an alternative data structure for a generic triangular mesh $\mathcal{S}^R$, based on a binary tree. More precisely, we resort to an alternate binary (AB) search tree, [96], see Figure 6.9 for an example.

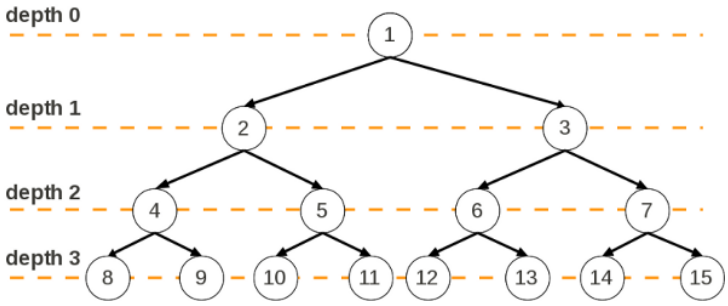


Figure 6.9: Example of AB search tree.

Let us explain how an element $T \in \mathcal{S}^R$ is stored in an AB data structure. At the beginning of the storage procedure, the tree data structure is empty, so that the triangle first considered is stored in the root ①, see Figure 6.9. Afterwards, each element T is stored, starting from the root, in the first available node identified via the algorithm PUTINTREE described in Algorithm 7.

Namely, we consider the vector $\vec{v}_T = (x^T_{sw}, x^T_{ne}, y^T_{sw}, y^T_{ne}, z^T_{sw}, z^T_{ne})^t \in \mathbb{R}^6$ associated with the bounding box $\mathcal{B}(T)$ of the element T, i.e., collecting the coordinates of the points $SW^T = (x^T_{sw}, y^T_{sw}, z^T_{sw})^t$ and $NE^T = (x^T_{ne}, y^T_{ne}, z^T_{ne})^t$, see Figure 6.7 on the left, as well as the vector $\vec{v}_\tau = (x^\tau_{sw}, x^\tau_{ne}, y^\tau_{sw}, y^\tau_{ne}, z^\tau_{sw}, z^\tau_{ne})^t$ associated with the triangle stored at the generic node τ of the tree.

A triangle T finds a location in the tree by properly comparing vectors $\vec{v}_T$ and $\vec{v}_\tau$. In particular, let j_τ denote the depth of the node τ and let $v_{T,i}$ be the i-th component of the vector $\vec{v}_T$, to choose the right coordinate we use the modulus 6, see line 7 in Algorithm 7.

Algorithm 7 Algorithm to put an element in the tree data structure

PUTINTREE ($\vec{v}_T$)

1: $\tau \leftarrow$ root;
2: $\mathrm{j}_\tau = 0$;
3: **if** τ is empty **then**
4: $\quad \tau \leftarrow K$
5: $\quad$ **return**
6: **end if**
7: $\alpha = \mathrm{j}_\tau \pmod 6$;
8: **if** $v_{T,\alpha} < v_{\tau,\alpha}$ **then**
9: $\quad \tau \leftarrow \tau$.left;
10: **else**
11: $\quad \tau \leftarrow \tau$.right;
12: **end if**
13: $\mathrm{j}_\tau = \mathrm{j}_\tau + 1$;

Here τ.left and τ.right, line 8 and 9 in Algorithm 7, denote the child on the left and on the right of τ, respectively, while $v_{T,\alpha}$ and $v_{\tau,\alpha}$ denote the α-th component of vectors $\vec{v}_T$ and $\vec{v}_\tau$, respectively. The comparison test in the algorithm depends on the depth of the tree. The inequalities **alternate** both the extremes of the bounding boxes and the coordinates, x, y and z, this justifies the name of the algorithm as well as it makes this approach suited to organize data in any dimension.

Once an AB data structure is built for the mesh $\mathcal{S}^R$, we proceed with the intersection with the other mesh $\mathcal{S}^B$. For any element $T \in \mathcal{S}^B$, we find the set $\mathcal{D}_T$ of the triangles in $\mathcal{S}^R$ near T. In particular, we build the bounding box $\mathcal{B}(T)$ associated with T, we go through the binary tree of $\mathcal{S}^R$, moving from the

root. The triangle associated with the node τ of the tree is included in $\mathcal{D}_T$ if the corresponding bounding box $\mathcal{B}(\tau)$ intersects $\mathcal{B}(T)$.

This procedure is summarized in Algorithm 8. The algorithm INTINTREE() takes as input the vector $\vec{v}_T$ associated with the bounding box of the element T of $\mathcal{S}^B$, $\mathcal{B}(T)$. It returns a stack Q that contains all the triangles in $\mathcal{S}^R$ that intersect the bounding box of T.

Algorithm 8 Algorithm to find the elements stored in the tree that intersect the bounding box of an element T

INTINTREE ($\vec{v}_T$)

```
1:  S.push(root);
2:  while S is empty do
3:     τ = S.pop;
4:     j_τ = depth(τ);
5:     α = j_τ(mod 6);
6:     if α is even then
7:        k = α + 1;
8:        if v_{T,k} < v_{τ,α} then
9:           S.push(τ.left);
10:       else
11:          Q.push(τ);
12:          S.push(τ.left); S.push(τ.right);
13:       end if
14:    else
15:       k = α - 1;
16:       if v_{T,k} ≥ v_{τ,α} then
17:          S.push(τ.right);
18:       else
19:          Q.push(τ);
20:          S.push(τ.left); S.push(τ.right);
21:       end if
22:    end if
23: end while
24: return Q
```

The stack S is a temporary stack and it is initially empty, while Q will contain the list of the triangles constituting $\mathcal{D}_T$. The region $\mathcal{D}_T$ identified via an AB data structure is, in general, different from the one obtained via a structured data

search. In particular, the AB approach allows us to build a really confined set $\mathcal{D}_T$, where only the triangles of $\mathcal{S}^R$ whose bounding box actually intersects $\mathcal{B}(T)$ are included. This property is not a priori guaranteed by the structured-data based approach, as shown in Figure 6.8.

Moreover, at each step of the procedure, one of the two children of τ is excluded from the search, see line 9 and 17 in Algorithm 8. To maximize the benefits due to this "cutting of branches", we should have a tree as balanced as possible. In fact, if we randomly choose the root and the triangles to be inserted, we risk to build an unbalanced tree. For instance, let us consider the mesh in Figure 6.10. If we choose element 5 as a root and then we iteratively insert elements 6, 9 and 10 following the procedure INTINTREE, we get a completely unbalanced tree, see Figure 6.11 left. On the other hand, if we choose element 9 as a root and then we insert elements 3, 15, 1, 5, 17 and 13, the resulting tree is completely balanced, see Figure 6.11 on the right. So if we are able to identify the right sequence of elements to be inserted in the tree, we **automatically** get a balanced tree.

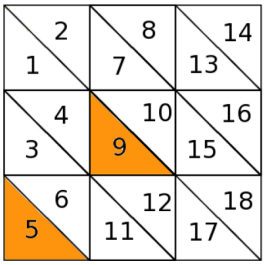


Figure 6.10: A two dimensional planar surface $\mathcal{S}$: two possible choices for the root, leading to a completely unbalanced (triangle 5) and to a balanced (triangle 9) binary tree data structure.

We propose the following approach to decide the insertion order. For a certain surface $\mathcal{S}$, we first define the six vectors SW_x, SW_y, SW_z, NE_x, NE_y, NE_z that collect, for each element $T \in \mathcal{S}$, the coordinates x, y, z of the points SW^K and NE^K defining the bounding box $\mathcal{B}(T)$. We sort the elements in each vector into the ascending order. Then, we build the binary tree data structure following a dichotomy principle. At depth 0, as root of the tree, we select the triangle coinciding with the median of the vector SW_x; the two nodes at depth 1 are represented by the two 4-quantiles of the vector SW_y of order $1/4$ and $3/4$; the four nodes at depth 2 are provided by the four 8-quantiles of the vector SW_z of order $1/8$, $3/8$,

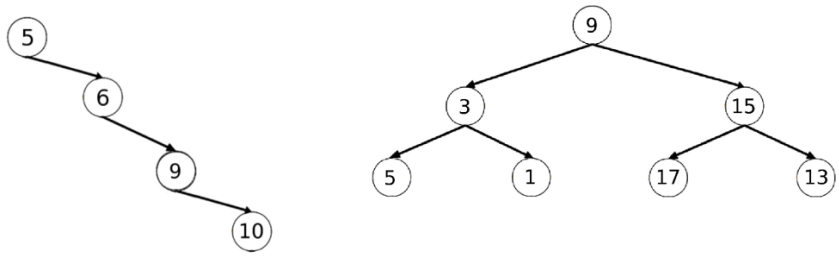


Figure 6.11: Example of unbalanced, on the left, and balanced, on the right, binary search tree associated with the two choices for the root highlighted in Figure 6.10.

$5/8$, $7/8$, and so on. The idea is to take level by level the following quantities:

$$\mathcal{I}_t = \left\{ \frac{2j+1}{t} < 1, j \in \mathbb{N} \right\},$$

with $t = 2^{i+1}$ and i the considered the depth of the tree. Of course, the triangle T is removed from the six vectors as soon as it is stored in the binary tree.

6.2.3 Coupling structured and AB data search

The two algorithms described in Sections 6.2.1 and 6.2.2 have complementary characteristics with respect to the properties i)-iii) itemized at the beginning of this section.

Both the approaches guarantee the first property. The algorithm based on a structured data search detects the set $\mathcal{D}_T$ very rapidly, in practice each search has $O(1)$ complexity, but this set can be rather large, thus violating requirement ii).

On the contrary, the algorithm based on the AB search builds a set $\mathcal{D}_T$ with a rather small number of elements. However, this procedure is, in general, less efficient: the complexity of a single search is, for a balanced tree, of order $O(logN)$, with N the number of elements in the tree.

Our actual goal is to combine these two procedures into a new algorithm able to merge the respective advantages. The basic idea consists of reducing the size of the problem $\mathcal{S}^R \cap \mathcal{S}^B$, by confining the intersection to suitable subsets $\mathcal{S}_s^R$ and $\mathcal{S}_s^B$ of $\mathcal{S}^R$ and $\mathcal{S}^B$, respectively, see Figure 6.12 for a sketch of the procedure.

In more detail, the reduction step, that is the generation of the sub-meshes $\mathcal{S}_s^R$ and $\mathcal{S}_s^B$, is performed via the quick, but rough, algorithm; the intersection phase is assigned to the sharp AB-algorithm. In this way the slower speed of the AB

approach is balanced by the reduced number of mesh elements now involved in the intersection. We can think about a sort of predictor-corrector approach, where the structured data search is the predictor and the AB-algorithm is the corrector.

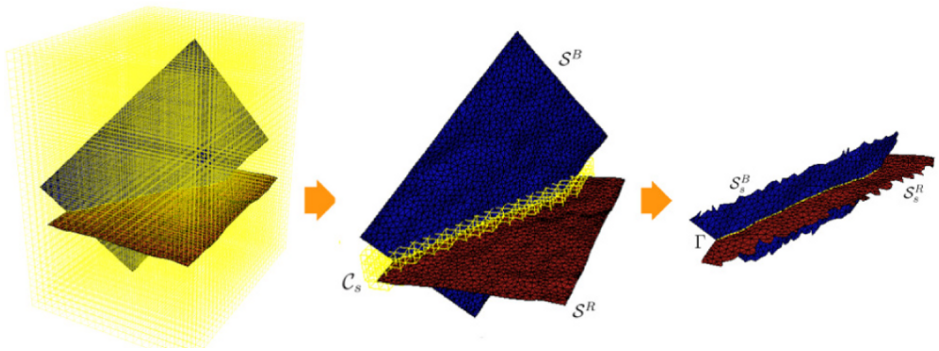


Figure 6.12: Reduction of the intersection $\mathcal{S}^R \cap \mathcal{S}^B$ to the problem $\mathcal{S}_s^R \cap \mathcal{S}_s^B$.

To itemize the main steps of the proposed procedure, we refer to the geometric configuration in Figure 6.12.

1. we associate a **unique** data structure to the union $\mathcal{S}^R \cup \mathcal{S}^B$ of the two intersecting surfaces, following the approach in Section 6.2.1: the hexahedral mesh now contains both the surfaces $\mathcal{S}^R$ and $\mathcal{S}^B$, Figure 6.12 on the left;

2. starting from this data structure, we extract the two sub-meshes $\mathcal{S}_s^B$ and $\mathcal{S}_s^R$. We first build the set $\mathcal{C}_s$ of the hexahedral cells where both the red and the blue triangles are stored together, Figure 6.12 in the middle; then we define $\mathcal{S}_s^B$ and $\mathcal{S}_s^R$ as the union of the triangles $T \in \mathcal{C}_s \cap \mathcal{S}^B$ and $K \in \mathcal{C}_s \cap \mathcal{S}^R$, respectively;

3. we apply the intersection algorithm in Section 6.2.2 to the reduced configuration $\mathcal{S}_s^R \cap \mathcal{S}_s^B$ to detect the intersection line Γ, see in Figure 6.12 on the right. Thus, the binary tree data structure is built for the sub-mesh $\mathcal{S}_s^R$ only.

We assess the performances of this algorithm on the setting in Figure 6.13. The mesh $\mathcal{S}^R$ is characterized by a non uniform sizing, i.e., we distinguish a portion, the one on the left, where the mesh is rather coarse, in contrast to the part on the right where the triangles are rather refined. Formulas (6.2)-(6.3) lead to a hexahedral mesh with huge cells, containing a large number of elements where $\mathcal{S}^R$ is fine. Nevertheless, the reduction step yields the sub-meshes $\mathcal{S}_s^R$ and $\mathcal{S}_s^B$ consisting of 1415 and 2614 triangles in contrast with 4368 and 15519 elements

for $\mathcal{S}^R$ and $\mathcal{S}^B$, respectively. As a consequence, the intersection $\mathcal{S}_s^R \cap \mathcal{S}_s^B$ based on the AB algorithm turns out to be faster, since it considers fewer elements. In Subsection 6.5.1, we numerically show the better performance of this algorithm.

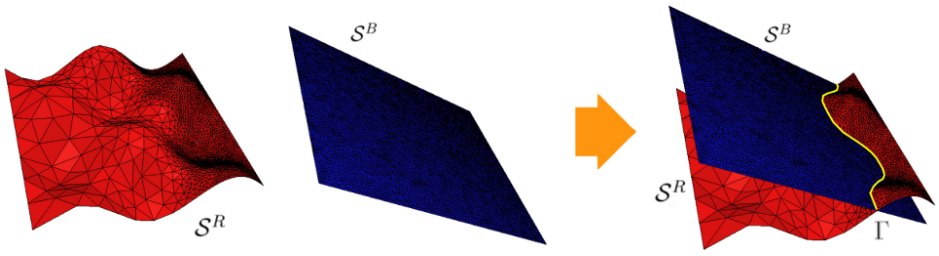


Figure 6.13: Intersecting non-uniform meshes.

6.2.4 Find Intersection Line

Here we recall the theory provided by T. Möller in [82], to find the intersection segment between two non co-planar triangles. We consider two generic triangles T_1 and T_2, whose vertices are $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ and $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$, respectively.

Via the vertices, we can compute the planes π_1 and π_2 identified by these triangles. Consider the triangle T_1, whose normal is given by

$$\mathbf{n}_1 = (\mathbf{v}_2 - \mathbf{v}_1) \wedge (\mathbf{v}_3 - \mathbf{v}_1)\,,$$

where the $\wedge$ denotes the standard vector product. Then, the equation of plane π_1 has the following form

$$\pi_1 \;:\; \mathbf{n}_1 \cdot \mathbf{x} + c_1 = 0\,, \tag{6.4}$$

where $\mathbf{x}$ denotes a generic point of the plane π_1 and

$$c_1 = -\mathbf{n}_1 \cdot \mathbf{v}_1\,.$$

The signed distance of $\mathbf{w}_i$ the vertices of T_2 to π_1 is simply computed by substituting the vertices in the plane Equation (6.4) of the plane, namely

$$d_{\mathbf{w}_i} = \mathbf{n}_1 \cdot \mathbf{w}_i + c_1 \quad i = 1, 2, 3.$$

If $d_{\mathbf{w}_i} \neq 0\ \forall i = 1, 2, 3$ or if all the $d_{\mathbf{w}_i}$ have the same sign, the triangle T_2 lies on one side of the plane π_1, so there is no intersection between T_1 and T_2. If $d_{\mathbf{w}_i} = 0$

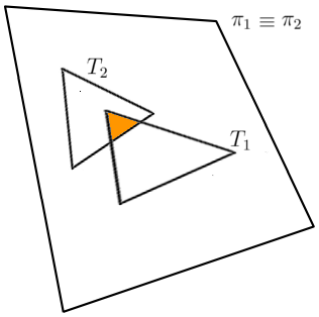


Figure 6.14: Examples of triangles that lie on the same plane: they do **not** identify any intersection line.

$\forall i = 1, 2, 3$, the triangles are co-planar, i.e., $\pi_1 \equiv \pi_2$. In this case there is no intersection line between the triangles, since they only share a common area, see Figure 6.14.

In all the other cases the intersection between the planes π_1 and π_2 is a straight line, L, and, consequently, the intersection between T_1 and T_2 is a subset of L. In particular L could be represented as:

$$\mathbf{x} = \mathbf{o} + t\mathbf{d}\,, \tag{6.5}$$

where $\mathbf{o}$ is a point on the line L, $\mathbf{d} = \mathbf{n}_1 \wedge \mathbf{n}_2$ and $t \in \mathbb{R}$ is the parameter describing the line L.

To find the intersection line, we consider the triangles T_1 and T_2, and then we compute the intersection line L between the planes that these triangles identify. Then, we find the pairs of $t-$parameters that identify the segments on the line L that belongs to the triangles T_1 and T_2, respectively.

The triangles T_1 and T_2 intersect if and only if these two segments overlap. So, the problem of finding the intersection line between T_1 and T_2 is reduced to find the intersection between these one dimensional intervals, [72].

In Figure 6.15 on the left, we show an example of two intersecting triangles, while, in Figure 6.15 on the right, we have two non intersecting triangles.

6.3 Region Detection

At this stage, the triangulated surface and the output of the intersection procedure are still separated entities, i.e., the intersection curve does not necessarily follow

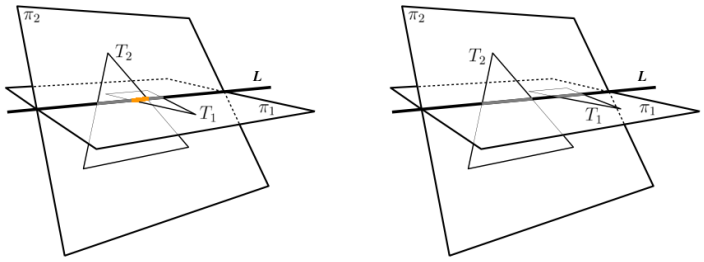


Figure 6.15: On the left we highlight the intersection line between T_1 and T_2, in orange, on the right there is no intersection between the two triangles.

the edges of the triangulation, but it may cross the mesh elements. Furthermore, the intersection curve identifies distinct regions on the mesh at hand, for instance in Figure 6.16, right we recognize nine distinct areas.

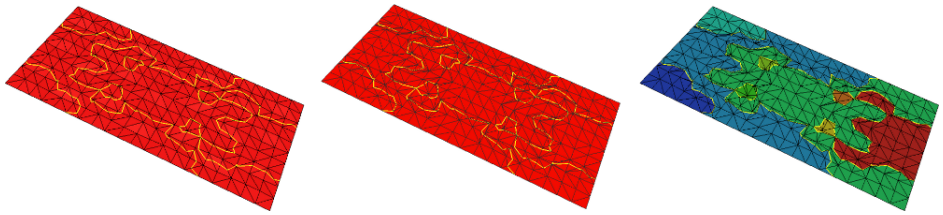


Figure 6.16: Example of composite intersection $\mathcal{I}$, the yellow line on the left; inclusion of $\mathcal{I}$ into the mesh, in the middle, and region detection, on the right.

Our actual goal is to detect such regions **automatically**. In particular, let $\mathcal{I}$ denote the intersection curve on a surface $\mathcal{S}$, $\partial\mathcal{S}$ being the boundary of $\mathcal{S}$. We aim at finding a partition $\mathcal{P} = \{\omega_1, \omega_2, \dots \omega_n\}$ of $\mathcal{S}$ such that:

- $\bigcup_{i=1}^{n} \omega_i = \mathcal{S}$;
- $\mathring{\omega}_i \cap \mathring{\omega}_j = \emptyset\,, \quad \forall i \neq j, i, j = 1, \dots, n$;
- $\bigcup_{i=1}^{n} \partial\omega_i = (\mathcal{I} \cup \partial\mathcal{S})$.

The sub-domains ω_i are assumed to be closed set and $\mathring{\omega}_i$ stands for the internal part of ω_i. The proposed approach consists of two distinct phases:

a) we include the intersection curve $\mathcal{I}$ into the surface mesh via a suitable re-meshing procedure. Hence, the information of these two distinct geometric entities is properly linked;

b) we subdivide the triangulated surface into regions ω_i in order to define a partition $\mathcal{P}$ of $\mathcal{S}$ matching the properties above.

In the two next sections we deal with these two phases, separately.

6.3.1 Inclusion of the intersection curve

This step is quite complex since we cannot make any a-priori assumption on the shape of the intersection curve $\mathcal{I}$. We only know that the segments constituting the intersection line lie in some triangle of the surface.

The proposed method aims at properly re-meshing each triangle crossed by $\mathcal{I}$ with the constraint of including the intersection segments in the new mesh. For this purpose, as first step, we have to find the elements crossed by $\mathcal{I}$. Successively, we have to properly re-mesh each crossed element, to guarantee the global conformity of the new mesh. We could detect the crossed triangles via the data structure search algorithms of Subsection 6.2.1 or 6.2.2.

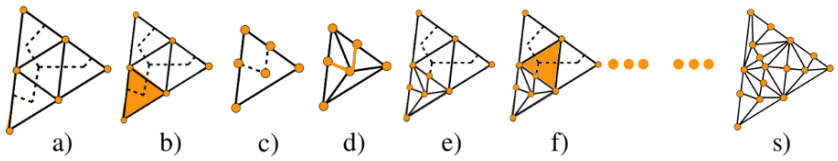


Figure 6.17: Example of inclusion of an intersection curve.

To illustrate the procedure which creates a new triangulation conforming with the intersections, let us consider the four elements in Figure 6.17, a). We aim at adding the dashed segments to the existing mesh to get a new mesh that includes them. As sketched in Figure 6.17, we process separately each of the four elements. We re-mesh each triangular element including the edges inside it, Figure 6.17, d). The conformity is inherited by the global mesh, Figure 6.17, s).

In Figure 6.16, middle we show the final result of the inclusion of the intersection curve procedure applied to the configuration on the left.

Remark 6.3.1 *The quality of the generated elements is not necessarily good. Really thin triangles might be generated after the re-meshing as shown in Figure 6.18. So, a process to recover equilateral triangles can be used to improve the quality of the mesh. We discuss this issue in Subsection 6.5.2.*

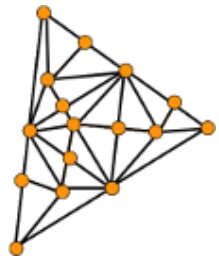


Figure 6.18: A zoom of the step s) of Figure 6.17.

6.3.2 Subdivision of a Mesh into Regions

The inclusion of the intersection curve $\mathcal{I}$ into the mesh $\mathcal{S}$ makes the detection of the different regions a straightforward task.

We apply a sort of **diffusive** procedure. We assign a source to a certain triangle of the mesh. Then, we exploit the diffusion of this source in the adjacent elements, driven by the triangle connectivity and by these simple rules:

1) each triangle T spreads the source into the adjacent triangles, i.e., the triangles that shares an edge with T;

2) the diffusion cannot cross the intersection segments.

At the end of the procedure, all the triangles of the mesh have to be assigned to a certain region ω_i. The proposed method identifies one region for time, lines 8-17 in Algorithm 9. So it has to be restarted until it has partitioned the whole surface, line 3 in Algorithm 9.

We formalize the subdivision into region procedure via the algorithm REGDET(). It takes as input the intersection line $\mathcal{I}$ and it provides as output a partition of the surface mesh.

Relation $(\partial T_i \cap \partial T \not\in \mathcal{I})$, line 13 in Algorithm 9, essentially checks if the triangle T_i is on the correct part of $\mathcal{I}$.

Algorithm 9 Algorithm to identify different regions

REGDET ($\mathcal{I}$)

Data: $\mathcal{T}$ be the set of the triangles mesh, $\mathcal{D}$ be the subset of $\mathcal{T}$ formed by the triangles already processed, S a temporary stack and i an integer that counts the regions.

1: $i = 0$;
2: $\mathcal{D} = \emptyset$;
3: **repeat**
4: $i = i + 1$;
5: $\omega_i = \emptyset$;
6: randomly take a triangle T such that $T \in \mathcal{T}$ && $T \notin \mathcal{D}$;
7: S.push(T);
8: **repeat**
9: T = S.pop();
10: insert T in ω_i;
11: Q_T is the star of T;
12: **for all** $T_i \in Q$ **do**
13: **if** $T_i \notin \mathcal{D}$ && $\partial T_i \cap \partial T \notin \mathcal{I}$ **then**
14: S_1.push(T_i);
15: **end if**
16: **end for**
17: **until** S is empty
18: **until** $\#\mathcal{T} \neq \#\mathcal{D}$

Notice that the stack S follows the diffusive process. Consider the region ω_i, at the beginning S contains a single randomly selected triangle $T \in \omega_i$. Then, via the for cycle, the triangles adjacent to T are gradually inserted in S and, successively, in ω_i.

In Figure 6.19, we exemplify such a procedure on the simple mesh reported in Figure 6.17 s). This configuration of the intersection line $\mathcal{I}$ requires that three different diffusive processes are developed.

Remark 6.3.2 *The procedures described in Sections 6.2 and 6.3 can be extended, in a straightforward way, to more general frameworks, e.g., to quadrilateral meshes. Moreover, these approaches can be generalized to a higher and to a lower dimension, for instance to detect volumes identified by a set of surfaces in a tetrahedral mesh, see Figure 6.20, or to detect the set of piecewise linear curves identified by*

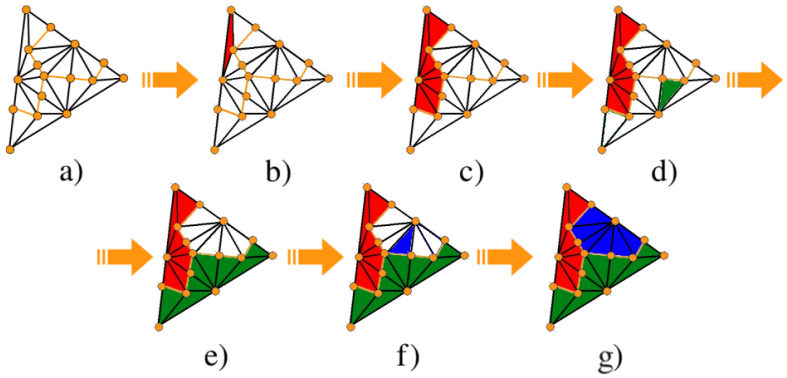


Figure 6.19: Example of region subdivision, the yellow curve marks $\mathcal{I}$.

a set of points, see Figure 6.21.

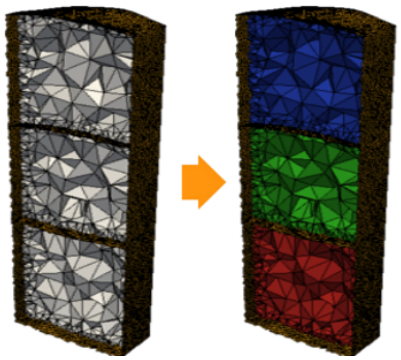


Figure 6.20: Example of volume detection moving from a set of surfaces.

6.4 Mesh Quality Improvement

The inclusion of the intersection curve, see Subsection 6.3.1, often yields really thin and stretched triangles. As we have already show in Chapter 1, the shape of the triangles influences the numerical simulations. In particular, we recall that the accuracy of the solution may improve if the elements are oriented according to a

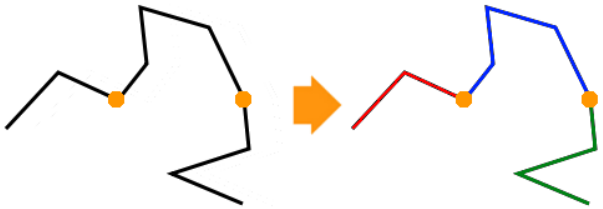


Figure 6.21: Example of line detection moving from a set of points. On the left the initial mesh where we highlight the set of point, the orange bullet; on the right the three part of the piecewise linear curve that these points identify.

precise criterion, for instance the solution of the partial differential equations we are interested in, see Section 1.1.1.

When we include the intersection curve, the mesh presents stretched and skinny triangles along the piecewise line we have inserted. But these triangles do **not** have a specific orientation and shape, so a priori they could lead to a high numerical error, and it could be necessary a mesh optimization strategy to get equilateral triangles.

To quantify the shape of a generic triangle T, we consider the so-called **aspect ratio**, introduced in Definition 1.1.2:

$$Q(T) := \frac{2r}{R},$$

where R, r are the radii of the circumscribed and of the inscribed circles, respectively. If T is an equilateral triangle, $Q(T) = 1$, viceversa if $Q(T) \ll 1$, T is a very stretched element. We resort the classical mesh optimization procedure, see Section 1.2, to improve the quality of the mesh, i.e., to make the aspect ratio of all the triangles of the mesh as close as possible to the optimal value.

In the following paragraphs we describe how the standard mesh operation described in Section 1.2 are applied, in order to get a good quality mesh.

Remark 6.4.1 *Even in this framework all the topological constraints described in Section 1.2 related to these mesh modification procedures have to be applied.*

Node Smoothing

We have implemented the trapezium drawing approach proposed in [104] to improve the aspect ratio of the mesh triangles without changing the connectivity of

the mesh node.

This kind of smoothing technique is particularly suited when we do not have any information about the real geometry, but we have only the triangulated surface. In fact, it does not require any information about the surface, but it finds the new location of a point $\mathbf{v}$ via the coordinates of the nodes connected to $\mathbf{v}$.

Edge Flipping

We consider two adjacent triangles, T_1 and T_2, i.e., two triangles that share one edge $\mathbf{ab}$ and the quadrilateral $T_1 \cup T_2$ formed by them. We choose the best diagonal for this quadrilateral, in order to maximize the minimum aspect ratio of these triangles.

In this framework, the presence of different sub-domains gives a further constraint on this operation. In fact, if we decide to flip an edge that lies on the common line between two different regions, we lose the boundary of the domains, see Figure 6.22. To preserve these lines, we add this condition on the flipping of a generic edge $\mathbf{ab}$:

(iv) the edge $\mathbf{ab}$ does not belong to the boundary between two different subdomains.

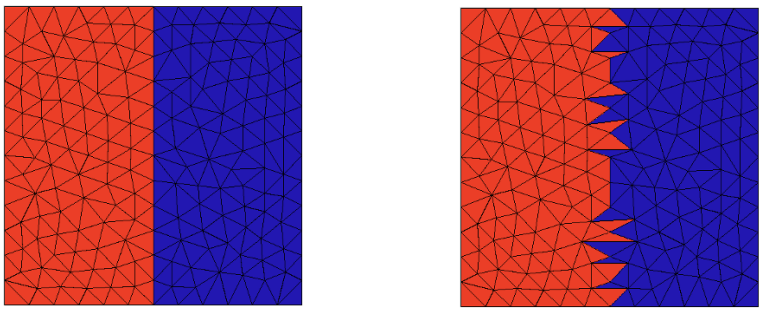


Figure 6.22: Flipping of the edges that belongs to the boundary between two subdomains.

Edge Contraction

Since the low quality triangles are characterized by too small angles, if we contract the small edges, i.e., the edges opposite to small angles, we improve the quality of the whole mesh.

Once we decide to contract an edge $\mathbf{ab}$, we have to decide the location of the new point $\mathbf{c}$ where the edge is contracted to. As we said in Section 1.2, there are different alternatives for the location of this point: one of the end-points or in the middle point of the segment.

When we consider a mesh subdivided in different domains like the ones shown in Figure 6.23 on the left, the location of the point $\mathbf{c}$ has to be carefully chosen. In fact, if we always contract the edge onto the middle point of a segment, we could have a wrong representation of the intersection line, see Figure 6.23 on the right.

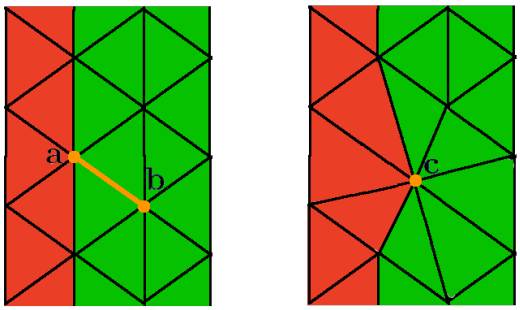


Figure 6.23: Contraction of the edge $\mathbf{ab}$ into the middle point $\mathbf{c}$; this creates a wrong approximation of the intersection line.

To overcome this issue, when we contract an edge $\mathbf{ab}$, we decide the new point $\mathbf{c}$ following these criteria:

(i) if $\mathbf{ab}$ is an interface edge or if both its end-points are inside the same sub-domain, we contract $\mathbf{ab}$ into the middle point;

(ii) if $\mathbf{a}$ is on the interface and $\mathbf{b}$ is inside one of the sub-domains, we contract the edge $\mathbf{ab}$ into $\mathbf{a}$;

(iii) if $\mathbf{b}$ is on the interface and $\mathbf{a}$ is inside one of the sub-domains, we contract the edge $\mathbf{ab}$ into $\mathbf{b}$;

(iv) if $\mathbf{a}$ and $\mathbf{b}$ are on the interface, but $\mathbf{ab}$ is not a boundary edge we do not contract the edge.

Condition (iv) avoids the possibility to get a bad approximation of a sub-domain. In fact, if we decide to contract this kind of edges, we get a wrong approximation of the subdomains independently of the choice for $\mathbf{c}$, see Figure 6.24.

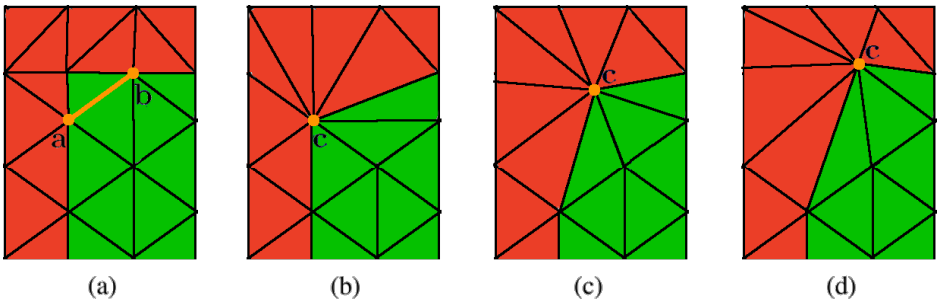


Figure 6.24: Contraction of the edge $\mathbf{ab}$ that does not satisfy condition (iv), if we contract the edge into one of its endpoints, (b) and (d), or into the middle point, (c), we do not get a good approximation of the sub-domain.

Edge Splitting

We apply this operation on too long edges in order to get a uniform edge length. Since we do not have any information about the real geometry represented by the mesh, we always halve the edge via its middle point.

Mesh Improvement Scheme

To improve the quality of the mesh, we follow the scheme in Algorithm 10. This algorithm receives the surface mesh, Γ_h and the number of iteration, N_{max}, as inputs and it applies the following sequence of operations.

As anticipated, the edge contraction is used to increase the minimal angle in the mesh Γ_h, but we use this operation combined with the edge splitting to make the mesh as uniform as possible. After these operations the quality of the triangles is further improved by the swapping and the smoothing operations, lines 11 - 19.

Algorithm 10 The mesh improvement scheme

IMPROVE(Γ_h, N_{max})

Data: i is a counter.

$i = 0$;
for $i < N_{max}$ **do**
  compute all the edge length of the mesh in a set L;
  L_{min} and L_{max} be the minimum and maximum of L;
  for all the triangles of Γ_h **do**
    contract the edges of length $< 0.3L_{\text{min}}$;
  end for
  for all the triangles of Γ_h **do**
    split the edges of length $> 0.7L_{\text{max}}$;
  end for
  for all the triangles of Γ_h **do**
    swap all the edges to improve the quality;
  end for
  for all the vertices of Γ_h **do**
    find the new location of the node with the smoothing;
  end for
  for all the triangles of Γ_h **do**
    swap all the edges to improve the quality;
  end for
    $i = i + 1$;
end for

6.5 Numerical Tests

In this section we give a more quantitative analysis on the proposed algorithms. Firstly, in Subsection 6.5.1, we analyse in more details the procedures related to the surface intersection and the region detection. Then, in Subsection 6.5.2, we show the reliability of the mesh improvement algorithm described in Algorithm 10. These tests are made by a laptop with a 2.26 GHz processor.

6.5.1 Surface Intersection and Region Detection Test

We consider two pairs of intersecting surfaces with the aim of detecting the corresponding intersection curves $\mathcal{S}_i^R \cap \mathcal{S}_i^B$, for $i = 1, 2$, as well as the regions bounded

by these, see Figure 6.25.

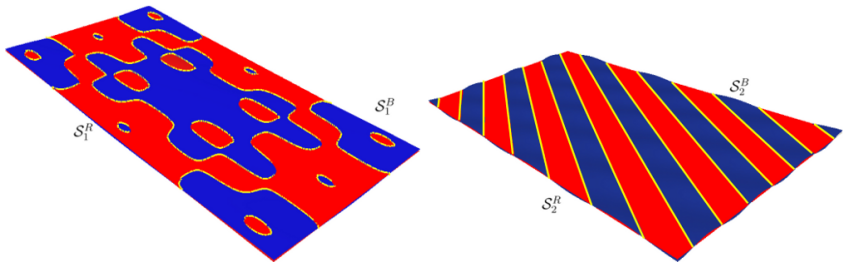


Figure 6.25: Intersecting surfaces: $\mathcal{S}_1^R \cap \mathcal{S}_1^B$, left, $\mathcal{S}_2^R \cap \mathcal{S}_2^B$, right.

To approximate the four surfaces, we resort to different families of meshes. We compare in terms of CPU time the performances of the surface intersection procedures addressed in Section 6.2. Finally, we apply the region detection strategy proposed in Section 6.3.

Tables 6.1 and 6.2 gather the results of such a comparison for uniform meshes of about 6800, 68000 and 680000 triangles, respectively. In particular, for the different surface triangulations, we collect the CPU time, in seconds, required by the intersection algorithm based on a structured data search (second column), the intersection procedure exploiting an unbalanced (third column) and a balanced (fourth column) AB search tree, the mixed structured-AB data search approach (fifth column) and, finally, the region detection phase (sixth column).

$\mathcal{T}$	STRUCT.	AB-UNBAL.	AB-BAL.	MIXED	REG. DETECT.
$\simeq 6800$	2	3	2	2	1
$\simeq 68000$	8	121	45	13	2
$\simeq 680000$	30	35148	10114	67	19

Table 6.1: $\mathcal{S}_1^R \cap \mathcal{S}_1^B$: CPU time for the different surface intersection algorithms and for the region detection on uniform meshes.

The values in Table 6.1 and 6.2 confirm, first of all, the importance of creating a balanced binary tree: by comparing the values in the third and in the fourth column, we recognize that the CPU time approximately triplicates in the case of an unbalanced binary tree, it becomes almost 30 times greater for the second configuration approximated via the finest mesh.

$\mathcal{T}$	STRUCT.	AB-UNBAL.	AB-BAL.	MIXED	REG. DETECT.
$\simeq 6800$	2	5	3	2	1
$\simeq 68000$	20	142	49	50	3
$\simeq 680000$	62	22753	791	172	23

Table 6.2: $\mathcal{S}_2^R \cap \mathcal{S}_2^B$: CPU time for the different surface intersection algorithms and for the region detection on uniform meshes.

Then, we remark that the approach proposed in Subsection 6.2.3 improves the performances of the intersection algorithm based on an AB tree, even though the tree is balanced.

The gain becomes particularly evident for increasingly finer meshes and in the case of the first geometric configuration where the surface intersection is more localized. The selected sub-meshes $\mathcal{S}_{1,s}^B$ and $\mathcal{S}_{1,s}^R$ are small enough to speed up the AB tree search procedure. On the contrary, a more widespread surface intersection, as in the case $\mathcal{S}_2^R \cap \mathcal{S}_2^B$, does not necessarily lead to small sub-meshes, $\mathcal{S}_{2,s}^B$ and $\mathcal{S}_{2,s}^R$, with a consequent less significant reduction of the corresponding CPU times.

Moreover, both the Tables 6.1 and 6.2 suggest us that the best intersection algorithm is the one based on a structured data search for both the geometric configurations and for this kind of meshes. Finally, the region detection is a really cheap operation in terms of computational costs for both the configurations and for each of the meshes to be selected.

In Tables 6.3 and 6.4 we show the peak of memory characterizing each process. We observe that, for a fixed number of elements, the values associated with the different search algorithms are roughly comparable. Thus, for the two considered configurations, the memory usage does not represent a relevant criterion to select the search algorithm.

$\mathcal{T}$	STRUCT.	AB-UNBAL.	AB-BAL.	MIXED	REG. DETECT.
$\simeq 6800$	14.0MB	15.0MB	15.0MB	15.0MB	24.4MB
$\simeq 68000$	42.9MB	34.0MB	34.0MB	46.5MB	60.7MB
$\simeq 680000$	408.6MB	254.0MB	254.0MB	437.1MB	444.5MB

Table 6.3: $\mathcal{S}_1^R \cap \mathcal{S}_1^B$: peak of memory during the different surface intersection algorithms and for the region detection on uniform meshes.

$\mathcal{T}$	STRUCT.	AB-UNBAL.	AB-BAL.	MIXED	REG. DETECT.
$\simeq 6800$	14.7MB	15.8MB	15.8MB	15.8MB	25.4MB
$\simeq 68000$	27.6MB	33.9MB	33.9MB	33.6MB	60.2MB
$\simeq 680000$	232.2MB	254.0MB	254.0MB	275.1MB	456.2MB

Table 6.4: $\mathcal{S}_2^R \cap \mathcal{S}_2^B$: peak of memory during the different surface intersection algorithms and for the region detection on uniform meshes.

The conclusion drawn above about the better performances of the structured data search algorithm is no longer the same when considering non-uniform meshes, as already anticipated in Subsection 6.2.3 on a simpler configuration. The columns in Table 6.5 provide the CPU time demanded by the structured data search algorithm, by the balanced AB tree approach and by the procedure proposed in Subsection 6.2.3, respectively when dealing with non-uniform meshes. The surfaces $\mathcal{S}_i^B$ and $\mathcal{S}_i^R$, with $i = 1, 2$, are approximated via meshes consisting of about 20000 and 65000 elements, respectively. Figure 6.26 shows the corresponding surfaces of intersection where the non-uniform structure of the meshes is evident.

	STRUCT.	AB-BAL.	MIXED
$\mathcal{S}_1^R \cap \mathcal{S}_1^B$	593	27	14
$\mathcal{S}_2^R \cap \mathcal{S}_2^B$	624	30	18

Table 6.5: CPU time for the different surface intersection algorithms on non-uniform meshes.

	STRUCT.	AB-BAL.	MIXED	REG. DETECT.
$\mathcal{S}_1^R \cap \mathcal{S}_1^B$	15.6MB	18.4MB	18.4MB	27.6MB
$\mathcal{S}_2^R \cap \mathcal{S}_2^B$	15.9MB	18.1MB	18.8MB	28.8MB

Table 6.6: Peak of memory during the different surface intersection algorithms and for the region detection on non-uniform meshes.

In the presence of non-uniform meshes, the mixed structured-AB data search approach turns out to be the most effective: a significant saving in terms of CPU time is guaranteed with respect to the balanced AB tree procedure and, as expected, it becomes even more remarkable with respect to the structured data search

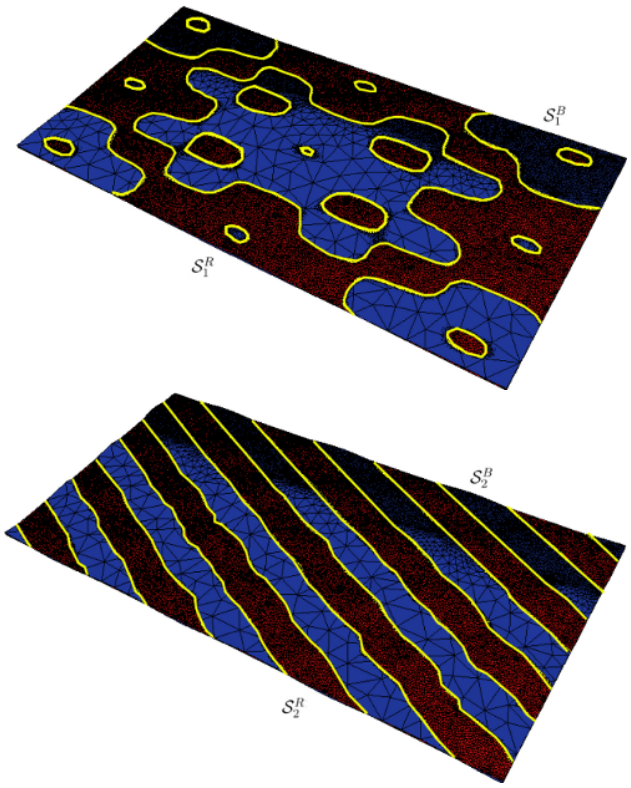


Figure 6.26: Intersecting surfaces and corresponding meshes: $\mathcal{S}_1^R \cap \mathcal{S}_1^B$, top, $\mathcal{S}_2^R \cap \mathcal{S}_2^B$, bottom.

algorithm. As shown in Table 6.6, also in the presence of non-uniform meshes, the different data search algorithms exhibit analogous values for the corresponding peak of memory.

We conclude this subsection by comparing the proposed mixed structured-AB search approach with one search algorithm available in the CGAL library [2]. This library provides a fast intersection detection algorithm based on the Axis Aligned Bounding Box (AABB) Tree method, [96]. Nevertheless, this procedure identifies only the triangles which intersect each other and not the entire intersection line. To obtain a fair comparison, we have accordingly modified our code by adding the option to identify only the intersecting triangles.

The Table 6.7, 6.8 and 6.9 collect the results of this comparison, in terms of CPU time. We remark that the CPU time of the proposed algorithm shows a

$\mathcal{T}$	STRUCT.	AB-UNBAL.	AB-BAL.	MIXED	CGAL
$\simeq 6800$	1	9	3	2	1
$\simeq 68000$	4	130	40	5	5
$\simeq 680000$	10	15321	510	69	65

Table 6.7: $\mathcal{S}_1^R \cap \mathcal{S}_1^B$: CPU time for the different surface intersection algorithms on uniform meshes.

$\mathcal{T}$	STRUCT.	AB-UNBAL.	AB-BAL.	MIXED	CGAL
$\simeq 6800$	1	4	2	1	1
$\simeq 68000$	2	112	39	14	4
$\simeq 680000$	18	17412	542	121	58

Table 6.8: $\mathcal{S}_2^R \cap \mathcal{S}_2^B$: CPU time for the different surface intersection algorithms on uniform meshes.

	STRUCT.	AB-BAL.	MIXED	CGAL
$\mathcal{S}_1^R \cap \mathcal{S}_1^B$	17	3	1	1
$\mathcal{S}_2^R \cap \mathcal{S}_2^B$	16	4	2	1

Table 6.9: CPU time during the different surface intersection algorithms on non-uniform meshes.

scalability with respect to the number of elements which is very similar to the one characterizing the CGAL algorithm. We highlight that, while the CGAL library is strongly optimized, no special optimization has been performed on the code that implements the mixed structured-AB search algorithm.

6.5.2 Mesh Quality Test

Example 1

In this example we consider the surface in Figure 6.27 on the left. In Figure 6.27 right, we show the output of the mesh quality improvement procedure in Algorithm 10. The number of the mesh elements is significantly increased; the original mesh has 1136 elements while the regularized mesh consists of 2726 triangles. But, now all the triangles are close to an equilateral element.

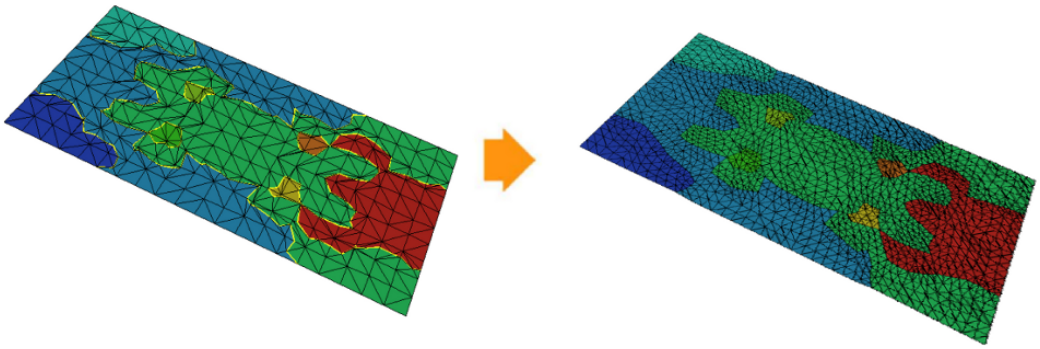


Figure 6.27: Instance of mesh quality improvement: original mesh, left, and improved mesh, right.

This effect becomes more evident from the histograms in Figure 6.28, that represent the distribution of the quality index $Q(T)$ on the mesh elements before and after the mesh regularization. In fact, the values of the quality index of the final mesh, Figure 6.28 on the right, are more concentrated around the optimal value than the ones of the initial mesh, Figure 6.28 on the left.

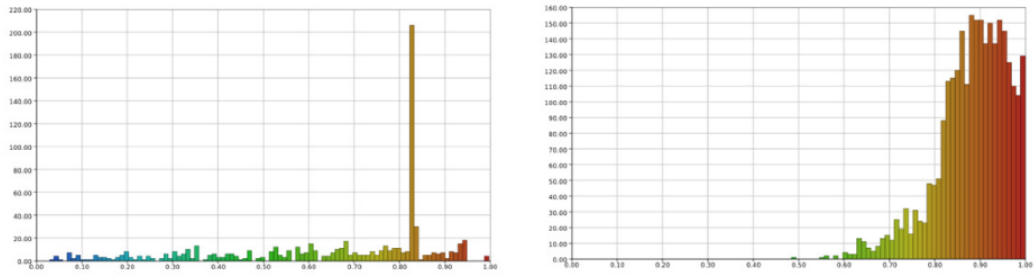


Figure 6.28: Distribution of $Q(T)$ before, left, and after, right, the mesh quality improvement.

Example 2

Now we consider the surface of the Stanford bunny, in Figure 6.29 on the left. In this case we are dealing with a surface divided into five subdomains. From the details in Figure 6.30 on the left, we appreciate the presence of distorted triangles,

especially close to the interfaces.

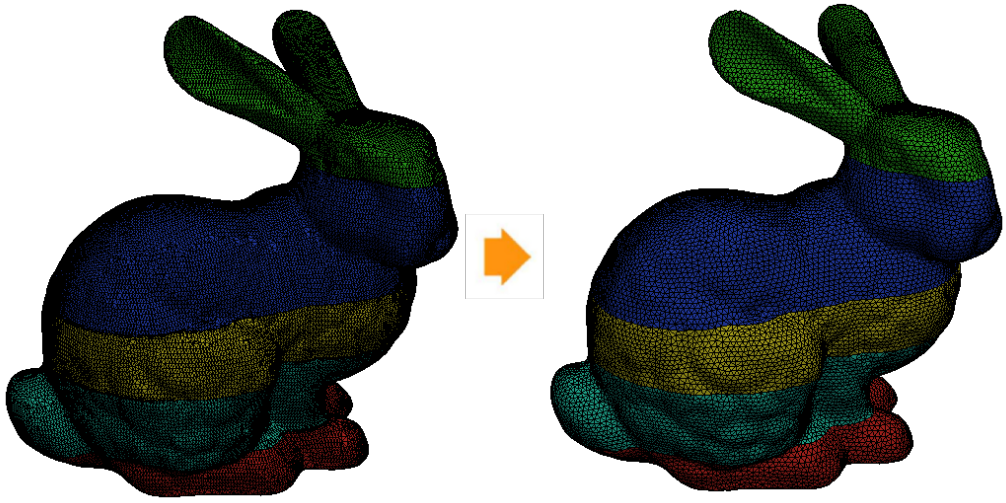


Figure 6.29: Instance of mesh quality improvement: original mesh, left, and improved mesh, right.

From Figure 6.29 and 6.30 on the right, we notice that the triangles have a shape closer to the equilateral one and that the intersection lines are preserved in the final mesh.

In Figure 6.31, we give a more quantitative analysis on this mesh improvement scheme. Even in this case the quality indexes of the triangles move to the optimal value, i.e. 1, when we consider the improved mesh.

6.6 Application to the Geometrical Modelling of a Geological Basin

This section covers some other fundamental aspects that may be relevant when dealing with the geometrical modelling of geological basins. Namely, the possibility of dealing with missing data, the treatment of the so-called hard and soft horizons and the construction of a volume of interest.

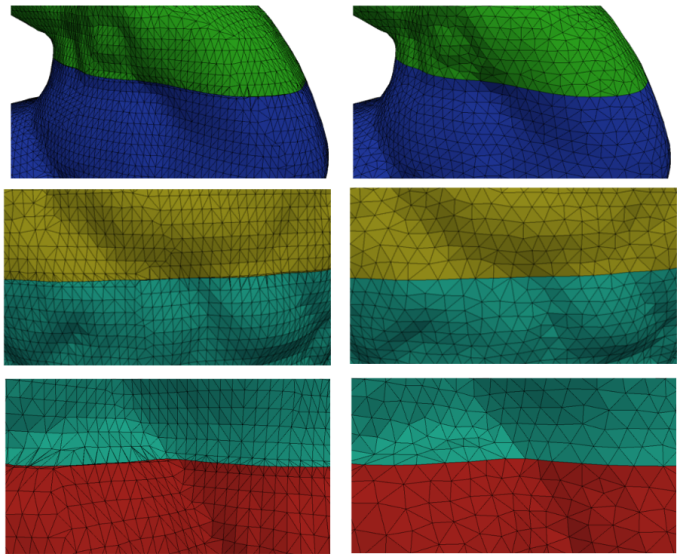


Figure 6.30: Some details of the mesh in Figure 6.29: original mesh, left, and improved mesh, right.

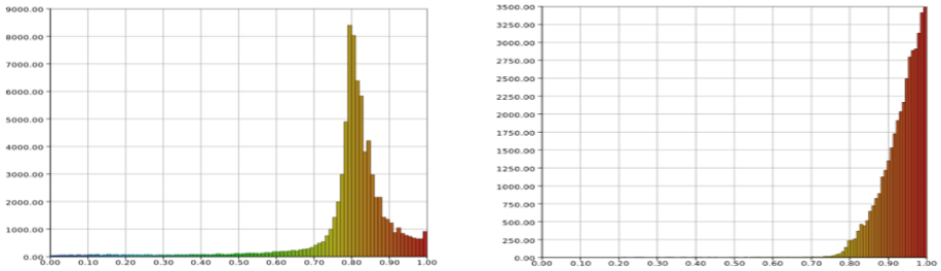


Figure 6.31: Distribution of $q(T)$ before, left, and after, right, the mesh quality improvement.

6.6.1 Lack of data

The seismic data used for the simulations are often incomplete or non reliable due to either a lack of existing coverage or inadequate and old measurements. Thus, moving from the available information, it is sometimes necessary to reconstruct the horizons in the geological basin of interest before generating a corresponding tetrahedral mesh.

In particular, we investigate two different techniques to deal with a possible lack of data:

- kriging;
- radial basis function.

Kriging

As first approach, we resort to a well-known geo-statistical technique, i.e., the **kriging** [20]. It is a regression method to recover the value of a certain field at unobserved locations starting from observations of the same field in nearby sites. In this case, the field of interest is the location of a horizon characterized by a lack of information, for instance, a hole. To apply this technique, we need to assume that the surface is described via the graph of a function f, i.e., $z = f(x, y)$, we refer to Figure 6.32 for a possible example.

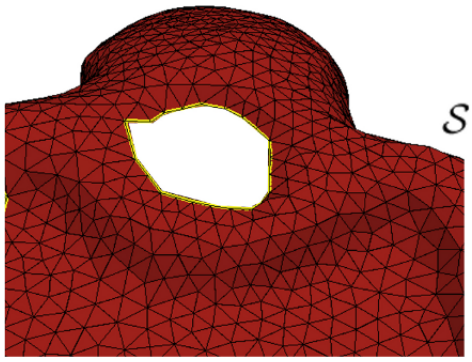


Figure 6.32: Example of a hole suitable for the Kriging recovery approach.

In more detail, moving from the z-coordinates associated with a set $\mathcal{N}_P$ of points on the horizon, we recover the z-coordinate, z_P, of a point P located inside a hole as

$$z_P = \sum_{Q \in \mathcal{N}_P} \lambda_Q \, z_Q \, , \tag{6.6}$$

with λ_Q a suitable weight associated with the point $Q \in \mathcal{N}_P$.

The type of kriging determines the choice for the weights. We resort to a standard ordinary kriging where the weights essentially depend on the variogram associated with the starting data [20]. Moreover, the points in $\mathcal{N}_P$ are not necessarily

spread on the whole horizon, but they might be located only in a neighbourhood of the point P. The computation of the weights λ_Q is not always an easy task. In fact, it resorts to several searching processes, that lead to the solution of several linear systems. Details on kriging may be found in [20].

Radial Basis Function

As an alternative approach we employ an implicit representation of the horizons based on radial basis functions, [63]. An horizon $\mathcal{S}$ could be identified as the zero-level iso-surface of a suitable function $f : \mathbb{R}^3 \to \mathbb{R}$, i.e.,

$$\mathcal{S} = \{P \in \mathbb{R}^3 : \ f(P) = 0\}$$

with

$$f(P) = \sum_{Q \in \mathcal{N}} c_Q \, \phi(r_P), \tag{6.7}$$

where $r_P = ||P - Q||$ is the standard Euclidean distance between P and Q, ϕ the so called **radial basis function** and c_Q are proper coefficients.

There are different choices for the radial basis function ϕ, see Table 6.10. In this framework we use $\phi(r) = r^3$.

function	parameters	usage
$\phi(r) = \sqrt{c^2 + r^2}$	$c > 0$	topography
$\phi(r) = r^\beta$	$\beta = 1 \ \beta = 3$	data in $\mathbb{R}^3$
$\phi(r) = r^2 \log r$		data in $\mathbb{R}^3$

Table 6.10: Most common radial basis functions

The coefficients c_Q in Equation (6.7) are determined by imposing interpolation constraints. Since this condition usually leads to solve a full hill-conditioned system, we use an iterative solver with a proper pre-conditioner [59].

Thanks to the numerical validation, we believe that the approach based on an implicit representation of the non-complete horizon is more suited to deal with any kind of surfaces. This method allows us to treat in a more straightforward way a large variety of surfaces, even the ones that cannot be expressed via the graph of a function.

Independently of the technique adopted to recover the missing data, we properly close the hole via an advancing front mesh generator. The initial front will

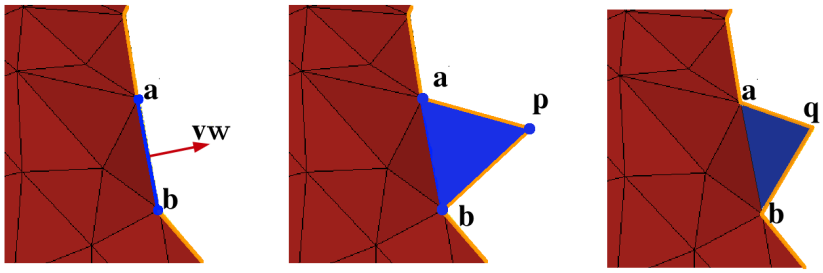


Figure 6.33: The first step of the advancing front technique. We highlight the front, orange line, the edge **ab**, in blue, where the advancing front method starts and the direction **vw**, where we look for the correct position of **p**, left; we get the position of **p** such that the triangle **abp** is equilateral, middle, then we move **p** on the closest point of the surface, **q** and we generate a triangle **abq**, right.

coincide with the boundary of the hole. Then, we iteratively add new triangles until we reconstruct the whole surface.

Given an edge e of the front, we consider the normal direction **vw**, i.e., the direction perpendicular to the triangle and to the edge e that points inside the hole, see Figure 6.33 on the left. Then, we find the location of a point **p** such that the triangle **abp** is equilateral, Figure 6.33 in the middle.

Then we move this point **p** on the closest point of the surface, **q**, reconstructed via one of the missing data recovering techniques, see Figure 6.33 on the right. We proceed iteratively for each front edge until we cover the entire hole, see Figure 6.34.

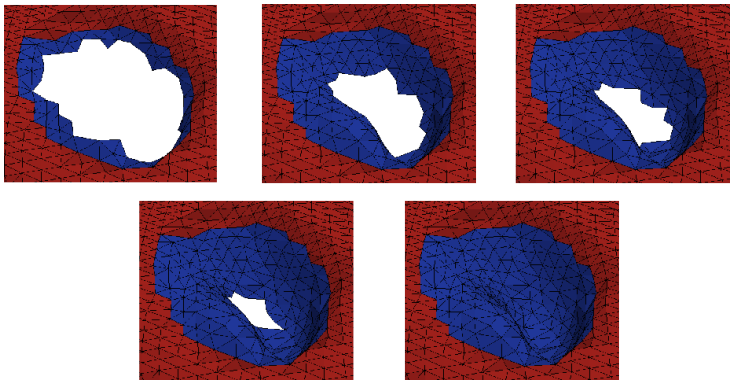


Figure 6.34: A sequence of steps for an advancing front mesh generator.

6.6.2 Hard and soft rocks

Different kinds of rock do exist in nature. As a possible classification, we may distinguish them in hard and soft rocks. The hardness of a rock essentially depends on the nature of the grains constituting the rock as well as on what kind of natural glue holds them together. Rocks baked in the deep underground are usually very hard, marble for instance, while mudstones and shales are examples of soft rocks. The different nature of the geological layers overlapping in the basin of interest has to be properly taken into account when generating the corresponding geometry. For instance, if a layer of marble stands above a layer of clay, we expect that the layer below is compressed by the layer above. Of course, direct measurements, such as core drillings, can be helpful in recovering these scenarios.

To deal with this possible interplay among horizons, we have set up an ad hoc procedure. Let us focus on the geological configuration in Figure 6.35, left: a hard horizon, $\mathcal{S}_h$, is compressing a soft horizon, $\mathcal{S}_s$. In the geometry of interest, a new surface, $\mathcal{S}_r$, replaces both the horizons $\mathcal{S}_h$ and $\mathcal{S}_s$. In more detail, we assume that $\mathcal{S}_r$ coincides with $\mathcal{S}_s$ in the regions where $\mathcal{S}_h$ lays over $\mathcal{S}_s$; viceversa, $\mathcal{S}_r$ coincides with $\mathcal{S}_h$ where $\mathcal{S}_h$ lays under $\mathcal{S}_s$, see Figure 6.35, right.

From an operative point of view, all the operations described in this chapter play a key role in the "hard and soft" procedure. In particular, we compute the intersection curve $\mathcal{S}_h \cap \mathcal{S}_s$ and we detect on both $\mathcal{S}_h$ and $\mathcal{S}_s$ the regions bounded by this curve.

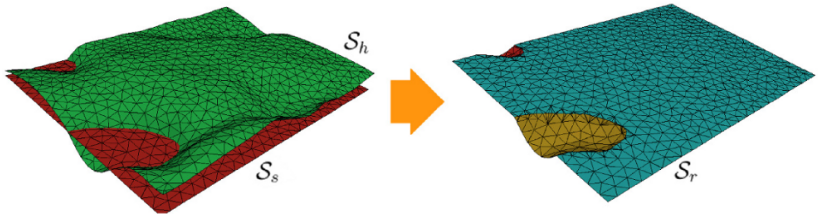


Figure 6.35: Starting geological configuration, on the left, a hard horizon, the green surface, intersects a soft horizon the red one; the horizon $\mathcal{S}_r$ replaces $\mathcal{S}_h$ and $\mathcal{S}_s$, on the right.

6.6.3 Selection of Sub-volumes

Very often we are interested in recovering geological information related to a certain sub-volume of interest rather than to the whole sedimentary basin, see Fig-

ure 6.36. This volume becomes the representative of the basin at hand. In particular, we aim at generating a tetrahedral mesh of this geological volume which takes into account the presence of any horizons and faults. Before proceeding with the volume mesh generation, suitable preprocessing procedures are sometimes demanded on the involved horizons and faults.

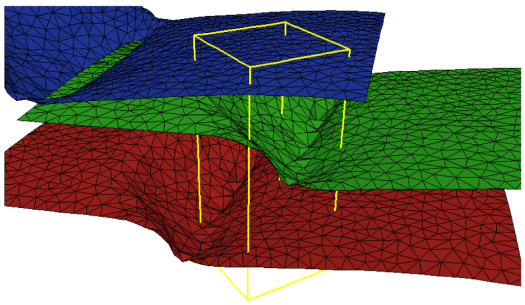


Figure 6.36: Example of volume of interest in a geological configuration.

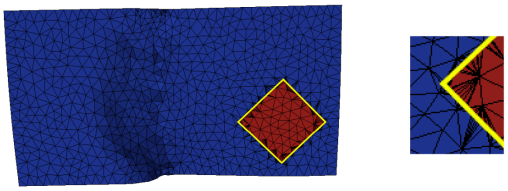


Figure 6.37: In red a portion of the blue horizon in Figure 6.36 cut by the volume of interest, on the left; zoom in on the corner leftmost, on the right.

Purpose of this section is to show how the geometric operations introduced Sections 6.2, 6.3 and 6.4 can be useful in this preprocessing phase. First of all, we are able, via the surface intersection and the region detection procedures, to identify, for each horizon, the corresponding portion cut by the volume of interest, see for an example Figure 6.37. Starting from these cutouts on the different horizons, we can build the boundary of the geological volume.

Nevertheless, as highlighted in Section 6.4, the involved operations among surfaces often corrupt the quality of the mesh elements, this is evident in Figure 6.37 right, as well as in Figure 6.38 right, where a lot of stretched triangles appear. Thus, before dealing with the tetrahedral mesh generation, it is crucial to

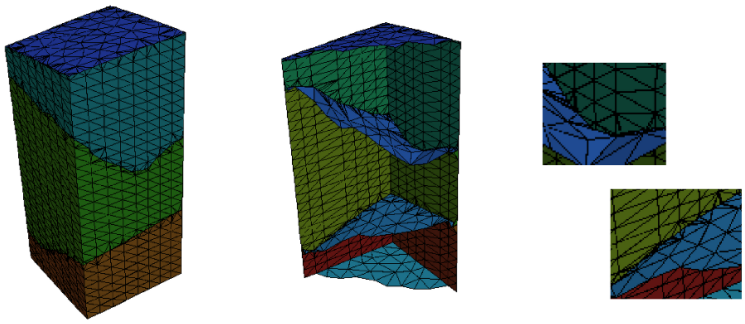


Figure 6.38: Example of a geological volume of interest on the left; a corresponding vertical cutoff in the middle; two zooms in on poor quality triangles, on the right.

improve the mesh quality of horizons and faults. We pursue this task essentially by exploiting the four geometric operations in Section 6.4. Finally, the three dimensional mesh is built using a generalized Delaunay procedure implemented in the "TetGen" library, [108].

This whole procedure allows us to obtain a detailed representation of the geological volume of interest via a tetrahedral mesh of good quality, which is constrained to the horizons and faults inside the volume. Figure 6.39 shows an instance of the outcome of the procedure, when applied to a rather complex geological configuration.

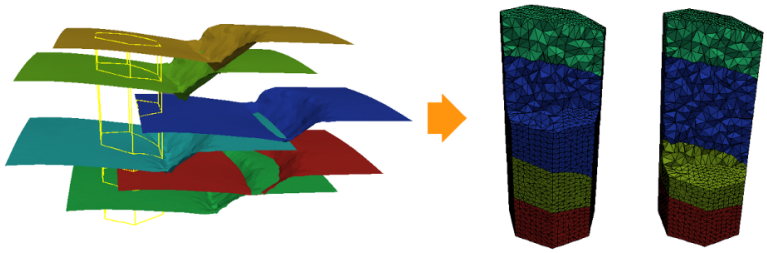


Figure 6.39: Example of tetrahedral mesh generation for the geological volume of interest on the left; two vertical cutoffs of the resulting volumetric mesh on the right.

Conclusions and Future Works

In this chapter we collect the main conclusions arising from the new results and techniques developed in this thesis. More precisely, we focus on the last four chapters, since the first two are devoted to introduce the basic concepts related to mesh generation and adaptation and to the discretization of Partial Differential Equations defined on Surfaces via a finite element approach.

Anisotropic Error Estimators for PDEs Defined on Surfaces

In Chapter 3 we extend the theory provided in [40] to Partial Differential Equations defined on surfaces. We derive a novel anisotropic interpolation error estimator for a function defined a surface. The results provided by this new approach have lead us to employ the anisotropic interpolation estimate into a-posteriori analysis context, for approximating the standard Laplace-Beltrami problem and a generic convection-diffusion problem defined on a surface. Finally, in the last part of this chapter, we propose a generalization of the well-known Zienkiewicz-Zhu error estimator to functions defined on surfaces and to an anisotropic setting.

The preliminary and good quality numerical results suggest a possible generalization of this theory to more complex differential problems or different discretization schemes. A framework of undoubted interest is represented by unsteady problems, where the surface itself evolves.

Higher Dimensional Re-Meshing

In Chapter 4, we have focused on a curvature-adapted anisotropic surface remeshing method, based on a high-dimensional embedding. This new method directly optimizes a triangular mesh in the embedded via the classical mesh modification operations in such a way that the triangles are as uniform as possible in the this embedded space. It has several advantages, for instances, sharp features are

always correctly captured; it is robust in handling strong anisotropy, and it is easy to implement. The numerical results show that this novel method correctly works for a large variety of surfaces characterized by an arbitrarily complex geometry.

There are many still issues. A very important theoretical question refers to the map Φ involved in Equation (4.1): How well may the map Φ approximate the geodesic distances for 3d surfaces? Do upper or lower bounds on distance variations exist for this map? A theoretical study of these issues could lead to more efficient methods, and meshes with fewer elements.

The new edge-flip algorithm used in this chapter seems very useful in improving both the geometry approximation and the mesh quality. However, its actual termination is not yet rigorously proved. We found that the selection of the threshold angle for checking inverted faces is very crucial, see Definition 4.2.1, but a more rigorous choice of the threshold could produce highly stretched triangles in the adapted mesh.

Moreover, in practice, many surfaces are provided as a polygonal mesh, i.e., the original geometry is not available. A good recovery and estimation of the surface normals are necessary in such a case to achieve good results.

Since this is a preliminary study of this new method and we have essentially focused on the robustness of the algorithm in order to mesh very complex geometries, the running time of this implementation is far from being optimal and there are many possibilities to improve it.

Surface Mesh Simplification

In the first part of Chapter 5, we have introduced a surface mesh simplification algorithm that produces a simplified surface mesh that maintains a high fidelity in approximating polygonal models. This algorithm uses iterative edge contraction to get a surface mesh with a reduced number of elements. Moreover, in Section 5.2.2 we have proposed an alternative procedure to increase the speed of this process, when we are dealing with a mesh with a huge number of elements. Via this new method, we get better results in terms of computational time, but the shape, size and dimension of triangles are not always optimal to fit the surface as well as possible. So, a sensitivity analysis on the parameter t used in Algorithm 6 is one of the issue to be investigated in the future.

Then, in the second part of the chapter, we introduce a new mesh simplification process. The novelty of this new method is that the simplification is driven by both geometrical and data information, so that the final simplified mesh leads to qualitatively good statistical estimates. In particular, the proposed simplification

method generates a suitable mesh that approximates the original geometry and properly associates the original data with the new geometry allowing good inferential properties in terms of data displacement and distribution. This represents a crucial property when dealing with data related to real-life applications, such as the medical setting we have investigated.

We have performed a preliminary study on the weights of the cost functional involved in this geometric-statistic approach. Among future goals, we aim at providing a more rigorous approach to select the weights, for instance, by applying some proper optimization procedure strictly related to the application and/or the geometry at hand.

Furthermore, certain computational improvements of the simplification procedure are planned, such as, the employment of a greedy strategy during the edge contraction step.

Modelling a Sedimentary Basin

In Chapter 6 we illustrate various mesh generation techniques applied to the geometric reconstruction of complex geological structures. Besides the specific target application, they are of rather general use.

We have implemented and compared different data structures, also in combination, to conclude that the best data structure strictly depends on the kind of involved meshes. If the surface meshes are structured or uniform, the structured data search turns out to be the most effective algorithm. Vice-versa, the CPU times demanded by this straightforward approach can be large on non-uniform meshes. The mixed structured-AB data search approach proposed in Section 6.2.3 shows performances less sensitive with respect to the type of the mesh. On structured and uniform meshes, the CPU times are comparable with the ones guaranteed by the structured data algorithm; on the contrary, we have shown that the computational gain led by the mixed procedure can become extremely relevant when dealing with non-uniform meshes. To take advantage of this remark, it could be useful to have a computable measure of the "uniformity" of a mesh in order to automatically choose the best algorithm, or to properly alternate the two procedures.

The use of this new data structure allowed us to analyse very complex geological situations at an affordable cost. In particular, we have combined algorithms specialized to identify the intersection of horizons and faults with a simple, but effective, algorithm to automatically detect the different regions forming the geological basin, completed with suitable mesh enhancing algorithms. This has allowed us to obtain good and conforming surface meshes for the different portions

of the external and internal boundary of the basin of interest. The resulting surface are ready to be the input to a three-dimensional mesh generator for the production of meshes suitable for numerical simulations.

Moreover, the "mixed" technique suits well with parallel implementation, where each box is associated with a different processor or thread. Porting to GPU architectures is also one of the foreseen further development of this work.

Appendix A

Spatial Spline Regression Models

In this appendix we describe the theory provided by B. Ettinger et al. in [35]. More precisely, we deal with the regression analysis of data observed over non-planar bi-dimensional domains. This type of data structure occurs in different applications, for instance the cortical thickness of the brain and the shear stress generated by the blood flow over the wall of an internal carotid artery.

The method proposed by B. Ettinger et al. in [35] adopts a functional data analysis approach, and it uses a regression method that efficiently handles these data. It consists of two phases:

1. the mapping of the original surface domain to a flat domain;

2. the employment spatial regression methods suited to deal with data on planar domains, [97].

More precisely, they use a conformal map to flatten the original surface domain. The main advantage of using of a conformal map, with respect to any other map, is that it preserves the angles of the original surface domain in the planar domain. Then, they use the penalized least square estimation technique proposed in [92, 97] is used.

A.1 Flattering Map

In [35], the authors deal with data associated with points over a surface embedded in a three-dimensional space. For instance, the hemodynamic data are referred to points (x_1, x_2, x_3) on the artery wall that is a bi-dimensional non-planar domain.

In [97] some first analyses of these data were performed, by flattening a simplified version of the carotid domain. In particular, a new coordinate system is defined as (s, r, θ), where s is the curvilinear abscissa along the artery centerline, r the artery radius, and θ the angle of the surface point with respect to the artery centerline. The domain is then reduced to the plane $(s, \theta, \bar{r})$, where $\bar{r}$ is the average carotid radius on the carotid tract considered. This planar rectangular domain is essentially obtained by cutting the artery wall along the axial direction given by s and then opening and flattening the artery wall. In particular, this planar domain is equivalent to a simplified three-dimensional artery geometry, where the radius is kept fixed to a constant value and the curvature of the artery is not taken into account. The map just described will be referred to in the following as the *angular map*.

Notice that, by flattening the domain with the angular map and then applying a spatial regression method for planar domains, any information related to the vessel radius and curvature is lost; even though, these two geometrical quantities greatly influence the hemodynamics in the artery and statistically discriminate aneurysm presence and location, [99]. Moreover, to have a bijective angular map, it is necessary to exclude the aneurysmal sac, otherwise, different points on the carotid wall would be mapped to the same point on the plane.

A.2 Spatial Spline Regression Models

In this section we recall the spatial regression methods defined in [92], then we explain the approach proposed in [35].

A.2.1 Spatial Spline Regression Model for planar domains

In this section, we present the Spatial Spline Regression models for planar domains introduced in [92] and then generalized in [97].

Let $\{\mathbf{u}_i = (u_i, v_i);\ i = 1, \dots, n\}$ be a set of n fixed data locations on a bounded regular domain $\Omega \subset \mathbb{R}^2$. Let z_i be the real-valued variable of interest observed at point $\mathbf{u}_i$. Assume the model

$$z_i = f(\mathbf{u}_i) + \epsilon_i \quad \forall i = 1, \dots, n\,, \tag{A.1}$$

where ϵ_i are independent observational errors with null mean and constant variance, and f is a twice continuously differentiable real-valued function to be estimated. According to the Spatial Spline Regression model, the estimate of f is

found by minimizing the following functional

$$\sum_{i=1}^{n}(z_i - f(\mathbf{u}_i))^2 + \underbrace{\lambda \int_{\Omega}(\Delta f)^2 d\Omega}_{(I)}\,, \tag{A.2}$$

i.e., a sum of squared errors regularized via the L^2-norm of the Laplacian of f

$$\Delta f = \frac{\partial^2 f}{\partial u^2} + \frac{\partial^2 f}{\partial v^2}\,.$$

In Equation (A.2), the Laplacian of f measures the local curvature of f. Via the penalty, i.e., the term (I) in Equation (A.2), we are essentially controlling the roughness of the solution. Moreover, since the Laplacian operator is invariant with respect to Euclidean transformations of the domain, the smoothness of the estimate does not depend on the arbitrarily chosen coordinate system.

The estimation problem (A.2) cannot be solved analytically. As a consequence, an approximate solution is found by resorting to a finite element approach. Thanks to the intrinsic construction of the finite element space, in [97] the authors reduce the estimation problem (A.2) to a linear system.

A.3 Spatial Spline Regression Model for non-planar domains

Now, we move to the theory provided by B. Ettinger et al. in [35]. Consider a surface Γ embedded in the three-dimensional space and n fixed data locations $\{\mathbf{x}_i = (x_{1i}, x_{2i}, x_{3i}), \forall i = 1, \ldots, n\}$ that lie over Γ. For each location $\mathbf{x}_i$, a real-valued random variable of interest, z_i, is observed. As in the planar case, we assume the model

$$z_i = f(\mathbf{x}_i) + \epsilon_i \quad \forall i = 1, \ldots, n\,, \tag{A.3}$$

where ϵ_i are independent observational errors with mean 0 and constant variance, and f is a twice continuously differentiable real-valued function defined on the surface domain Γ. As above, the aim pursued before is to estimate this function.

In this case the manifold is just the support of the data, in the sense that the data are referred to locations lying on the manifold. We do not have any interest in analysing the properties of the manifold itself, but rather we use its geometrical properties when dealing with data occurring over it.

Generalizing (A.2), in [35] the authors propose to estimate f in (A.3) by minimizing the following penalized sum of squared error functional

$$J_\lambda(f(\mathbf{x})) = \sum_{i=1}^{n} (z_i - f(\mathbf{x}_i))^2 + \lambda \int_\Gamma (\Delta_\Gamma f(\mathbf{x}))^2 \, d\Gamma \,, \tag{A.4}$$

where Δ_Γ is the Laplace-Beltrami operator for functions defined over the surface Γ, see Chapter 2. Here we recall that the Laplace-Beltrami operator is the generalization of the common Laplacian and it is used to operate on functions defined on surfaces in Euclidean space.

In [35], B. Ettinger et al. show that it is possible to solve the estimation problem (A.4) by exploiting existing techniques for planar domains. In particular, the authors propose to reduce (A.4) to a problem over a planar domain. To achieve this goal, the idea is to flatten Γ by means of a conformal map. Specifically, for the surface domain Γ, they define a map

$$\begin{aligned} X : \Omega &\to \Gamma \,, \\ \mathbf{u} = (u, v) &\mapsto \mathbf{x} = (x_1, x_2, x_3) \,. \end{aligned} \tag{A.5}$$

where Ω is an open, convex and bounded set in $\mathbb{R}^2$. Denote by $X_u(\mathbf{u})$ and $X_v(\mathbf{u})$ the column vectors of first order partial derivatives of X with respect to u and v, respectively.

For the map X to be conformal, it is required that

$$\|X_u(\mathbf{u})\| = \|X_v(\mathbf{u})\| \,, \quad \langle X_u(\mathbf{u}), X_v(\mathbf{u}) \rangle = 0 \,, \quad \forall \mathbf{u} \in \Omega \,,$$

where $\langle \cdot, \cdot \rangle$ denotes the standard Euclidean scalar product and $\| \cdot \|$ is the corresponding norm.

Let us also define the space-dependent metric tensor as the following symmetric positive definite matrix

$$G(\mathbf{u}) := \begin{pmatrix} \|X_u(\mathbf{u})\|^2 & \langle X_u(\mathbf{u}), X_v(\mathbf{u}) \rangle \\ \langle X_v(\mathbf{u}), X_u(\mathbf{u}) \rangle & \|X_v(\mathbf{u})\|^2 \end{pmatrix} = \begin{pmatrix} g_{11}(\mathbf{u}) & g_{12}(\mathbf{u}) \\ g_{21}(\mathbf{u}) & g_{22}(\mathbf{u}) \end{pmatrix}.$$

where

$$\mathcal{W}(\mathbf{u}) := \sqrt{\det(G(\mathbf{u}))} \,,$$

and $G^{-1}(\mathbf{u}) = \{g^{ij}(\mathbf{u})\}_{i,j=1,2}$ is the inverse of the matrix $G(\mathbf{u})$.

Using this notation, for a function $f \circ X \in \mathcal{C}^2(\Omega)$, the Laplace-Beltrami operator associated with the surface Γ can be expressed as

$$\Delta_\Gamma f(\mathbf{x}) = \frac{1}{\mathcal{W}(\mathbf{u})} \sum_{i,j=1}^{2} \partial_i (g^{ij}(\mathbf{u}) \mathcal{W}(\mathbf{u}) \partial_j f(X(\mathbf{u}))$$

where $\mathbf{u} = X^{-1}(\mathbf{x})$. In [35], B. Ettinger et al. show that (A.4) can be equivalently expressed as the following problem over the planar domain Ω:

$$\begin{aligned} J_\lambda(f(X(\mathbf{u}))) = &\sum_{i=1}^{n} (z_i - f(X(\mathbf{u}_i)))^2 + \\ &+\lambda \int_\Omega \left[\frac{1}{\mathcal{W}(\mathbf{u})} \sum_{i,j=1}^{2} \partial_i (g^{ij}(\mathbf{u}) \mathcal{W}(\mathbf{u}) \partial_j f(X(\mathbf{u})) \right]^2 \mathcal{W}(\mathbf{u}) d\Omega \,, \end{aligned} \tag{A.6}$$

where $X(\mathbf{u}_i) = \mathbf{x}_i$. Moreover, for conformal coordinates, the functional J_λ reduces to

$$J_\lambda(f(X(\mathbf{u}))) = \sum_{i=1}^{n} \left(z_i - f(X(\mathbf{u}_i))\right)^2 + \lambda \int_\Omega \left[\frac{1}{\sqrt{\mathcal{W}(\mathbf{u})}} \Delta f(X(\mathbf{u})) \right]^2 d\Omega \,, \tag{A.7}$$

where Δf is the standard Laplacian over the planar domain Ω. Therefore, this problem turns out to be a modification of the estimation problem presented in Section A.2.1.

From a computational viewpoint, the conformal map in Equation (A.5) may be approximated via finite elements. The planar finite elements mentioned in Section A.2.1, can be adapted to a three-dimensional triangular mesh. In [57] a technique based on finite elements is specifically developed for flattening tubular surfaces. In [35], B. Ettinger et al. resort to a similar approach. This approach to estimate the conformal map uses a three-dimensional triangular mesh that approximates the original surface domain Γ. The three-dimensional mesh is flattened into a planar triangular mesh that discretizes Ω via the finite element approximation to the conformal map. One benefit of using a conformal map is that it preserves angles and thus shapes, i.e., compare Figure A.1 and A.2.

After the conformal flattening, we are ready to apply the estimation method of Section A.2.1 with the variant provided in Equation (A.7), to accommodate for the

domain deformation implied by the flattening phase. Note also that the estimates along the two "cut" sides have to coincide; this is in fact an artificial cut. To prevent a seam, we have to take care of maintaining the periodicity of the estimate along the "cut" edges, see [35, 97]. Similarly to SSR over planar domains, the estimator of f is linear with respect to the observed data values, so that classical inferential tools may be derived. In fact, many of the properties of SSR over planar domains hold for SSR over non-planar domains as provided in [35].

A.4 Simulations Studies

In this section, we provide the results on a first simulation study. We illustrate the performance of the smoothing technique over non-planar domains proposed by B. Ettinger et al.. In particular, we compare the results obtained via their SSR model for non-planar domains with those yielded by the SSR model for planar domains combined with a simple angular flattening.

Notice that the methods differ in two ways. The first is the flattening map. For the SSR model over non-planar domains, since it is generated by a conformal map, the triangulation preserves the shapes of the triangles in the original mesh. On the other hand, the triangulation generated by the angular map does not preserve the shape of the triangles in the original mesh.

The second difference is the penalty. The SSR models over non-planar domains use information from the conformal flattening map to adjust for the domain deformation implied by the map, hence considering the full three-dimensional domain.

Figure A.1, A.2 and A.3 illustrate the flattening of a test surface domain. Figure A.1 shows the original non-planar domain approximated by a three-dimensional triangular mesh, while Figure A.2 displays the conformally equivalent planar triangulated domain. Then, Figure A.3 shows the planar domain obtained with the angular map described in Section A.1.

The sides of the planar triangulations are labelled to have a correspondence with respect to the surface in Figure A.1. In particular, the sides of the planar triangulation labelled with "bottom" and "top" correspond to the bottom and to the top open boundaries of the original three-dimensional domain. The two sides indicated by "cut" correspond to a cut along the three-dimensional domain, connecting the two open boundaries of the surface, that is introduced when calculating the flattening map, see [57].

For these simulations, three domains are considered, see Figure A.4 and they

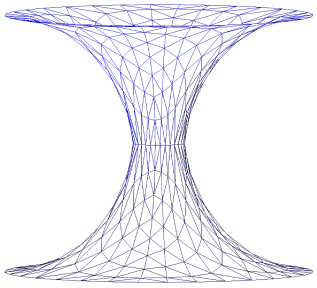


Figure A.1: Three-dimensional triangular mesh approximating a non-planar test domain.

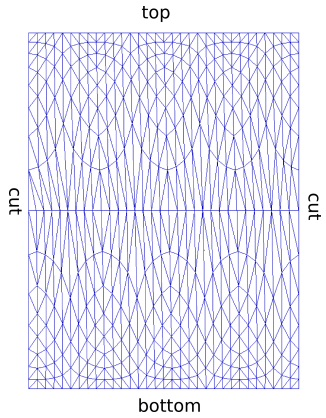


Figure A.2: The planar triangulation obtained by conformally flattening the domain in Figure A.1.

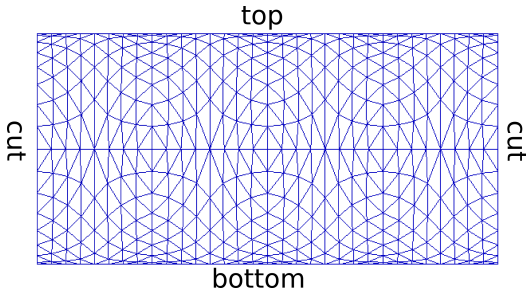


Figure A.3: The planar triangulated domain obtained by the angular flattening of the domain in Figure A.1.

are approximated by three-dimensional triangular meshes. Each geometry is topologically equivalent to a cylinder. Over each of these non-planar domains, we consider 50 test functions, having the form

$$f(x_1, x_2, x_3) = a_1 \sin(2\pi x_1) + a_2 \sin(2\pi x_2) + a_3 \sin(2\pi x_3) + 1$$

with coefficients a_i, for $i = 1, 2, 3$, randomly generated from independent normal distributions with mean 1 and standard deviation 1. The data locations $\mathbf{x}_i$ coincide with the nodes of the three-dimensional meshes.

Figure A.4: Three test surface domains. On each surface, the colour map indicates one of the selected test functions $f(x_1, x_2, x_3) = a_1 \sin(2\pi x_1) + a_2 \sin(2\pi x_2) + a_3 \sin(2\pi x_3) + 1$, with coefficients a_1, a_2 and a_3 randomly generated from independent normal distributions with mean 1 and standard deviation 1.

The noisy observations z_i in correspondence with the locations $\mathbf{x}_i$, for $i = 1, \ldots, n$, are obtained by adding independent normally distributed errors, with mean 0 and a standard deviation 0.5, to the test function, in accordance with model (A.3). An example of a test function and the corresponding level of noise is illustrated on each geometry in Figure A.4 and A.5, respectively.

For each simulation replicate, optimal values of the smoothing parameter λ in Equation (A.2) and (A.4) are selected by generalized cross validation for both the models on planar and non-planar domains, as described in [97] and [35], respectively.

Table A.1 shows the median and inter-quantile ranges of the Mean Square Errors (MSE) of f estimators over the 50 simulations. The table also reports the results of pairwise Wilcoxon tests verifying if the distribution of MSE for the estimates provided by SSR over non-planar domains (SSRNP) is stochastically lower than the distribution of the MSE for the estimates provided by SSR method over planar domains (SSRP), [111]. The p-values of these tests show that the MSE of SSR over non-planar domains estimates are significantly lower than the ones of SSR over planar domains, uniformly over the three surface domains considered.

Figure A.5: On each test surface, at each of the data location $\mathbf{x}_i$, coinciding with the nodes of the three-dimensional meshes approximating the surface domains, independent normally distributed errors with mean 0 and a standard deviation 0.5 are added to the test function; the colour maps are obtained by linear interpolation of the resulting noisy observations.

MSE	Geometry 1	Geometry 2	Geometry 3
angular map + SSRP	0.027 (0.018)	0.127 (0.130)	0.111 (0.153)
SSRNP domains	0.025 (0.017)	0.104 (0.095)	0.068 (0.055)
SSRNP vs. SSRP	$0.016e+00$	$5.300e-10$	$3.700e-09$

Table A.1: Median (inter-quantile ranges) of MSE of f estimators over the 50 simulations; p-values of pairwise Wilcoxon tests verifying if the distribution of MSE for the estimates provided by SSR over non-planar domains is stochastically lower than the distribution of the MSE for the estimates provided by SSR method over planar domains.

Figure A.6: The estimates provided by SSR over non-planar domains, with values of λ selected by generalized cross-validation.

Bibliography

[1] The stanford 3d scanning repository, University of Stanford, Standford U.S.A. https://graphics.stanford.edu/data/3Dscanrep/. (Cited in page 193).

[2] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org. (Cited in page 252).

[3] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Trans. Graph.*, 22(3):485–493, 2003. (Cited in page 155).

[4] I. Babuška and W. C. Rheinboldt. A-posteriori error estimates for the finite element method. *Int. J. Numer. Meth. Eng.*, 12(10):1597–1615, 1978. (Cited in page 22).

[5] I. Babuvška and W. C. Rheinboldt. Error estimates for adaptive finite element computations. *SIAM J. Numer. Anal.*, 15(4):736–754, 1978. (Cited in page 123).

[6] R. E. Bank and A. Weiser. Some a posteriori error estimators for elliptic partial differential equations. *Math. Comput.*, 44(170):283–301, 1985. (Cited in page 22).

[7] R. Beck, R. Hiptmair, R. H. Hoppe, and B. Wohlmuth. Residual based a posteriori error estimators for eddy current computation. *ESAIM: Math. Model. Numer. Anal.*, 34(01):159–182, 2000. (Cited in page 22).

[8] C. Bennis, J.-M. Vézien, and G. Iglésias. Piecewise surface flattening for non-distorted texture mapping. In *ACM SIGGRAPH Comput. Graph.*, volume 25, pages 237–246. ACM, 1991. (Cited in page 29).

[9] M. Bertalmıo, L.-T. Cheng, S. Osher, and G. Sapiro. Variational problems and partial differential equations on implicit surfaces. *J. Comput. Phys.*, 174(2):759–780, 2001. (Cited in page 27).

[10] F. J. Bossen and P. S. Heckbert. A pliant method for anisotropic mesh generation. In *Proceedings of the 5th International Meshing Roundtable*, pages 63–74, Albuquerque, NM, 1996. Sandia National Laboratories. (Cited in page 163).

[11] G. Bredon. *Topology and Geometry*, volume 139 of *Graduate Texts in Mathematics*. Springer, Berlin, 1993. (Cited in page 28).

[12] M. D. Buhmann. *Radial Basis Functions: Theory and Implementations*, volume 12. Cambridge University Press, 2003. (Cited in page 29).

[13] G. Canas and S. Gortler. Shape operator metric for surface normal approximation. In B. Clark, editor, *Proceedings of the 18th International Meshing Roundtable*, pages 447–461. Springer Berlin Heidelberg. (Cited in page 155).

[14] G. D. Cañas and S. J. Gortler. Surface remeshing in arbitrary codimensions. *Vis. Comput.*, 22(9-11):885–895, 2006. (Cited in pages 2, 151, 153).

[15] M. Castro-Diaz, F. Hecht, B. Mohammadi, and O. Pironneau. Anisotropic unstructured mesh adaption for flow simulations. *Int. J. Numer. Meth. in Flu.*, 25(4):475–491, 1997. (Cited in page 20).

[16] S. W. Cheng, T. K. Dey, and J. A. Levine. A practical Delaunay meshing algorithm for a large class of domains*. In *Proceedings of the 16th International Meshing Roundtable*, pages 477–494. Springer, 2008. (Cited in page 155).

[17] M. Chung, S. Robbins, and A. Evans. Unified statistical approach to cortical thickness analysis. In G. Christensen and M. Sonka, editors, *Information Processing in Medical Imaging*, volume 3565 of *Lecture Notes in Computer Science*, pages 627–638. Springer Berlin Heidelberg. (Cited in pages 204, 217).

[18] M. K. Chung, S. M. Robbins, F. K. M. Dalton, C. R. J. Davidson, A. L. Alex, and A. C. E. F. Cortical thickness analysis in autism with heat kernel

smoothing. *NeuroImage*, 25:1256–1265, 2005. (Cited in pages 204, 213, 217).

[19] P. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland, Amsterdam, 1978. (Cited in page 80).

[20] N. Cressie. *Statistics for spatial data*. Wiley series in probability and mathematical statistics: Applied probability and statistics. J. Wiley, 1993. (Cited in pages 257, 258).

[21] A. M. Dale, B. Fischl, and M. I. Sereno. Cortical surface-based analysis: I. segmentation and surface reconstruction. *NeuroImage*, 9(2):179 – 194, 1999. (Cited in page 204).

[22] H. L. de Cougny and M. S. Shephard. Surface meshing using vertex insertion. In *Proceedings of the 5th International Meshing Roundtable*, pages 243–256. Citeseer, 1996. (Cited in pages 2, 12).

[23] K. Deckelnick, G. Dziuk, C. M. Elliott, and C.-J. Heine. An h-narrow band finite-element method for elliptic equations on implicit surfaces. *IMA J. Numer. Anal.*, 30(2):351–376, 2010. (Cited in page 27).

[24] A. Demlow. Higher-order finite element methods and pointwise error estimates for elliptic problems on surfaces. *SIAM J. Numer. Anal*, 47(2):805–827, 2009. (Cited in pages 48, 57).

[25] A. Demlow and G. Dziuk. An adaptive finite element method for the Laplace-Beltrami operator on implicitly defined surfaces. *SIAM J. Numer. Anal.*, 45(1):421–442, 2007. (Cited in pages 33, 44, 57, 58, 62).

[26] A. Demlow and M. A. Olshanskii. An adaptive surface finite element method based on volume meshes. *SIAM J. Numer. Anal.*, 50(3):1624–1647, 2012. (Cited in page 27).

[27] C. Dobrzynski and P. Frey. Anisotropic Delaunay mesh adaptation for unsteady simulations. In *Proceedings of the 17th International Meshing Roundtable*, pages 177–194. Springer, 2008. (Cited in pages 2, 12).

[28] Q. Du and L. Ju. Finite volume methods on spheres and spherical centroidal voronoi meshes. *SIAM J. Numer. Anal.*, 43(4):1673–1692, 2005. (Cited in pages 52, 53).

[29] W. Du Toit. *Radial Basis Function Interpolation*. PhD thesis, Stellenbosch: Stellenbosch University, 2008. (Cited in page 29).

[30] R. Durán, M. A. Muschietti, and R. Rodríguez. On the asymptotic exactness of error estimators for linear triangular finite elements. *Numer. Math.*, 59(1):107–127, 1991. (Cited in page 141).

[31] G. Dziuk. Finite elements for the Beltrami operator on arbitrary surfaces. In S. Hildebrandt and R. Leis, editors, *Partial Differential Equations and Calculus of Variations*, volume 1357 of *Lecture Notes in Mathematics*, pages 142–155. Springer Berlin Heidelberg, 1988. (Cited in pages 2, 27, 28, 30, 32, 37, 43, 44, 45, 47, 49, 85, 89, 95, 116, 137).

[32] G. Dziuk and C. Elliott. l^2-estimates for the evolving surface finite element method. *Math. Comput.*, 82(281):1–24, 2013. (Cited in page 27).

[33] G. Dziuk and C. M. Elliott. Finite elements on evolving surfaces. *IMA J. Numer. Anal.*, 27(2):262–292, 2007. (Cited in page 27).

[34] H. Edelsbrunner, M. J. Ablowitz, S. H. Davis, E. J. Hinch, A. Iserles, J. Ockendon, and P. J. Olver. *Geometry and Topology for Mesh Generation (Cambridge Monographs on Applied and Computational Mathematics)*. Cambridge University Press, New York, NY, USA, 2006. (Cited in pages 12, 16, 18, 190).

[35] B. Ettinger, S. Perotto, and L. M. Sangalli. Spatial regression models over two-dimensional manifold. Technical report, MOX Report No. 54/2012, 2012. (Cited in pages 3, 204, 205, 209, 213, 267, 268, 269, 270, 271, 272, 274).

[36] P. Farrell, S. Micheletti, and S. Perotto. An anisotropic Zienkiewicz–Zhu-type error estimator for 3d applications. *Int. J. Numer. Meth. Engng.*, 85(6):671–692, 2011. (Cited in pages 11, 137, 141).

[37] D. A. Field. Laplacian smoothing and Delaunay triangulations. *Com. Appl. Numer. Math.*, 4(6):709–712, 1988. (Cited in page 162).

[38] L. Formaggia, S. Micheletti, and S. Perotto. Anisotropic mesh adaptation in computational fluid dynamics: Application to the advection-diffusion-reaction and the Stokes problems. *Appl. Numer. Math.*, 51(4):511–533, Dec. 2004. (Cited in pages 11, 12, 25).

[39] L. Formaggia, F. Nobile, A. Quarteroni, and A. Veneziani. Multiscale modelling of the circulatory system: a preliminary analysis. *Comput. Vis. Sci.*, 2(2-3):75–83, 1999. (Cited in page 185).

[40] L. Formaggia and S. Perotto. New anisotropic a priori error estimates. *Numer. Math.*, 89(4):641–667, 2001. (Cited in pages 11, 55, 56, 85, 97, 103, 263).

[41] L. Formaggia and S. Perotto. Anisotropic error estimates for elliptic problems. *Numer. Math.*, 94(1):67–92, 2003. (Cited in pages 11, 25).

[42] L. Formaggia, A. Quarteroni, and A. Veneziani. *Cardiovascular Mathematics*. MS&A: Modeling, Simulation and Applications. Springer, 2009. (Cited in page 185).

[43] P. Frey and H. Borouchaki. Surface meshing using a geometric error estimate. *Int. J. Numer. Methods Engng.*, 58(2):227–245, 2003. (Cited in page 12).

[44] P. J. Frey. Generation and adaptation of computational surface meshes from discrete anatomical data. *Int. J. Numer. Meth. Eng.*, 60(6):1049–1074, 2004. (Cited in page 23).

[45] P. J. Frey and F. Alauzet. Anisotropic mesh adaptation for CFD computations. *Comput. Meth. Appl. Mech. Engrg.*, 194(48):5068–5082, 2005. (Cited in page 25).

[46] P. J. Frey and H. Borouchaki. Geometric surface mesh optimization. *Comput. Vis. Sci.*, 1(3):113–121, 1998. (Cited in page 155).

[47] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 209–216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. (Cited in pages 12, 186).

[48] M. Garland and P. S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *Proceedings of the Conference on Visualization '98*, VIS '98, pages 263–269, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press. (Cited in page 185).

[49] C. Gauss, J. Morehead, and A. Hiltebeitel. *General investigations of curved surfaces of 1827 and 1825*. General Investigations of Curved Surfaces of 1827 and 1825. The Princeton university library, 1902. (Cited in page 153).

[50] C. Geuzaine and J. F. Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *Int. J. Numer. Meth. Engng.*, 79:1309–1331, 2009. http://www.geuz.org/gmsh/. (Cited in page 179).

[51] D. A. Gilbarg and N. S. Trudinger. *Elliptic Partial Differential Equations of Second Order*, volume 224. Springer, 2001. (Cited in pages 33, 34).

[52] G. H. Golub and C. F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996. (Cited in page 56).

[53] S. Gross and A. Reusken. *Numerical methods for two-phase incompressible flows*, volume 40. Springer, 2011. (Cited in page 48).

[54] A. Guéziec. *Surface Simplification Inside a Tolerance Volume*. IBM TJ Watson Research Center, 1996. (Cited in page 186).

[55] W. Habashi, M. Fortin, J. Dompierre, M.-G. Vallet, and Y. Bourgault. Anisotropic mesh adaptation: a step towards a mesh-independent and user-independent cfd. In *Barriers and Challenges in Computational Fluid Dynamics*, pages 99–117. Springer, 1998. (Cited in page 20).

[56] D. J. Hagler, A. P. Saygin, and M. I. Sereno. Smoothing and cluster thresholding for cortical surface-based group analysis of fMRI data. *NeuroImage*, 33(4), 2006. (Cited in page 205).

[57] S. Haker, S. Angenent, A. Tannenbaum, and R. Kikinis. Nondistorting flattening maps and the 3-d visualization of colon ct images. *IEEE Trans. Med. Imag.* (Cited in pages 271, 272).

[58] E. Hartmann. On the curvature of curves and surfaces defined by normalforms. *Comput. Ai. Geom. Design*, 16(5):355–376, 1999. (Cited in pages 32, 71).

[59] M. Heroux, R. Bartlett, V. H. R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, and A. Williams. An overview of Trilinos. Technical

Report SAND2003-2927, Sandia National Laboratories, 2003. (Cited in page 258).

[60] H. Hoppe. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108. ACM, 1996. (Cited in pages 2, 12).

[61] H. Hoppe. Progressive meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 99–108, New York, NY, USA, 1996. ACM. (Cited in page 185).

[62] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 19–26. ACM, 1993. (Cited in pages 2, 12, 186).

[63] A. Iske. *Multiresolution Methods in Scattered Data Modelling*, volume 37. Berlin, Springer, 2004. (Cited in page 258).

[64] W. H. Kim, D. Pachauri, C. Hatt, M. K. Chung, S. Johnson, and V. Singh. Wavelet based multi-scale shape features on arbitrary surfaces for cortical thickness discrimination. In P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1250–1258. 2012. (Cited in page 204).

[65] R. Kimmel, R. Malladi, and N. Sochen. Images as embedded maps and minimal surfaces: movies, color, texture, and volumetric medical images. *Int. J. Comput. Vis.*, 39(2):111–129, 2000. (Cited in page 153).

[66] D. Kovacs, A. Myles, and D. Zorin. Anisotropic quadrangulation. *Comput. Ai. Geom. Design*, 28(8):449–462, 2011. (Cited in page 153).

[67] M. Křížek and P. Neittaanmäki. Superconvergence phenomenon in the finite element method arising from averaging gradients. *Numer. Math.*, 45(1):105–116, 1984. (Cited in page 141).

[68] M. Křížek and P. Neittaanmäki. On a global superconvergence of the gradient of linear triangular elements. *Journal of computational and applied mathematics*, 18(2):221–233, 1987. (Cited in page 141).

[69] G. Kunert. An a posteriori residual error estimator for the finite element method on anisotropic tetrahedral meshes. *Numer. Math.*, 86(3):471–490, 2000. (Cited in page 22).

[70] P. Ladeveze and D. Leguillon. Error estimate procedure in the finite element method and applications. *SIAM J. Numer. Anal.*, 20(3):485–509, 1983. (Cited in page 22).

[71] Y. K. Lai, Q. Y. Zhou, S. M. Hu, J. Wallner, D. Pottmann, et al. Robust feature classification and editing. *IEEE Trans. Vis. Comput. Graph.*, 13(1):34–45, 2007. (Cited in pages 2, 151, 153, 154, 178).

[72] D. H. Laidlaw, W. B. Trumbore, and J. F. Hughes. Constructive solid geometry for polyhedral objects. In *Computer Graphics (Proceedings of SIGGRAPH 86)*, volume 20:4, pages 161–170, aug 1986. (Cited in page 238).

[73] A. Lakhany, I. Marek, and J. Whiteman. Superconvergence results on mildly structured triangulations. *Comput. Meth. Appl. Mech. Engrg.*, 189(1):1–75, 2000. (Cited in page 141).

[74] C. L. Lawson. Software for C^1 surface interpolation. *Mathematical Software III, Academic Press*, pages 164–191, 1977. (Cited in page 14).

[75] B. Lévy and N. Bonneel. Variational anisotropic surface meshing with voronoi parallel linear enumeration. In *Proceedings of the 21st International Meshing Roundtable*, pages 349–366. Springer, 2013. (Cited in pages 2, 151, 152, 153, 154, 155, 180, 181, 182, 183, 184).

[76] X. Li, M. S. Shephard, and M. W. Beall. 3D anisotropic mesh adaptation by mesh modification. *Comput. Meth. Appl. Mech. Engng.*, 194(48):4915–4950, 2005. (Cited in page 25).

[77] D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygonal environments. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 199–208, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. (Cited in page 188).

[78] S. Łukaszyk. A new concept of probability metric and its applications in approximation of scattered data sets. *Comput. Mech.*, 33(4):299–304, 2004. (Cited in page 29).

[79] G. Maisano, S. Micheletti, S. Perotto, and C. Bottasso. On some new recovery-based a posteriori error estimators. *Comput. Meth. Appl. Mech. Engrg.*, 195(37):4794–4815, 2006. (Cited in pages 22, 141).

[80] E. Marchandise, J.-F. Remacle, and C. Geuzaine. Optimal parametrizations for surface remeshing. *Engng. Comput.*, pages 1–20, 2012. (Cited in page 29).

[81] E. Miglio, S. Perotto, and F. Saleri. Model coupling techniques for free-surface flow problems: Part i. *Nonlin. Anal. Theor. Meth. Appl.*, 63(57):e1885 – e1896, 2005. (Cited in page 185).

[82] T. Möller. A fast triangle-triangle intersection test. *J. Graph. Tools*, 2(2):25–30, 1997. (Cited in page 237).

[83] R. Nicosia and A. Codova. *Geometria*. Sei, 1984. (Cited in page 8).

[84] J. T. Oden and S. Prudhomme. Goal-oriented error estimation and adaptivity for the finite element method. *Comput. Math. Appl.*, 41(5):735–756, 2001. (Cited in page 23).

[85] M. A. Olshanskii and A. Reusken. A finite element method for surface pdes: matrix properties. *Numer. Math.*, 114(3):491–520, 2010. (Cited in page 27).

[86] M. A. Olshanskii, A. Reusken, and J. Grande. A finite element method for elliptic equations on surfaces. *SIAM J. Numer. Anal.*, 47(5):3339–3358, 2009. (Cited in page 27).

[87] S. J. Owen, D. R. White, and T. J. Tautges. Facet-based surfaces for 3d mesh generation. In *In Proceedings 11th International Meshing Roundtable*, pages 297–311, 2002. (Cited in page 155).

[88] J. Peraire, M. Vahdati, K. Morgan, and O. Zienkiewicz. Adaptive remeshing for compressible flow computations. *J. Comput. Phys.*, 72(2):449 – 466, 198. (Cited in pages 9, 11).

[89] M. Picasso. Numerical study of the effectivity index for an anisotropic error indicator based on zienkiewicz–zhu error estimator. *Comm. Numer. Meth. Engng.*, 19(1):13–23, 2003. (Cited in pages 20, 137).

[90] H. Pottmann, T. Steiner, M. Hofer, C. Haider, and A. Hanbury. *The isophotic metric and its application to feature sensitive morphology on surfaces*. Springer, 2004. (Cited in page 153).

[91] S. Prudhomme and J. T. Oden. On goal-oriented error estimation for elliptic problems: application to the control of pointwise errors. *Comput. Meth. Appl. Mech. Engng.*, 176(1):313–331, 1999. (Cited in page 23).

[92] T. Ramsay. Spline smoothing over difficult regions. *J. R. Stat. Soc. Ser. B Stat. Methodol.* (Cited in pages 267, 268).

[93] J. F. Remacle, C. Geuzaine, G. Compre, and E. Marchandise. High-quality surface remeshing using harmonic maps. *Int. J. Numer. Meth. Eng.*, 2010. (Cited in page 29).

[94] R. Ronfard and J. Rossignac. Full-range approximation of triangulated polyhedra. *Comput. Graph. Forum*, 15(3):67–76, 1996. (Cited in pages 186, 190).

[95] J. Rossignac and P. Borrel. Multi-resolution 3d approximations for rendering complex scenes. In B. Falcidieno and T. Kunii, editors, *Modeling in Computer Graphics*, IFIP Series on Computer Graphics, pages 455–465. Springer Berlin Heidelberg, 1993. (Cited in page 187).

[96] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, 2006. (Cited in pages 226, 231, 252).

[97] L. Sangalli, J. Ramsay, and T. Ramsay. Spatial spline regression models. Technical report, Tech. rep. N. 08/2012, MOX, Dipartimento di Matematica "F.Brioschi", Politecnico di Milano, 2012. (Cited in pages 267, 268, 269, 272, 274).

[98] L. M. Sangalli, J. O. Ramsay, and T. O. Ramsay. Spatial spline regression models. *J. Ro. Stat. Soc. B Stat. Meth.*, 75(4):681–703, 2013. (Cited in page 204).

[99] L. M. Sangalli, P. Secchi, S. Vantini, and A. Veneziani. A case study in exploratory functional data analysis: geometrical features of the internal carotid artery. *J. Amer. Statist. Assoc.* (Cited in page 268).

[100] R. Schneiders. Mesh generation & grid generation on the web. http://www.robertschneiders.de/meshgeneration//meshgeneration.html. (Cited in page 18).

[101] J. Schöberl. Netgen an advancing front 2d/3d-mesh generator based on abstract rules. *Comput. Vis. Sci.*, 1(1):41–52, 1997. (Cited in page 19).

[102] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. *SIGGRAPH Comput. Graph.*, 26(2):65–70, July 1992. (Cited in page 186).

[103] V. Selmin and L. Formaggia. Simulation of hypersonic flows on unstructured grids. *Int. J. Numer. Meth. Eng.*, 34(2):569–606, 1992. (Cited in pages 9, 11, 20).

[104] I. B. Semenova, V. V. Savchenko, and I. Hagiwara. Two techniques to improve mesh quality and preserve surface characteristics. In *Proceedings, 13th International Meshing Roundtable*, pages 277–288, 2004. (Cited in page 244).

[105] J. A. Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press, 1999. (Cited in page 30).

[106] J. R. Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Appl. Comput. Geom. Geom. Engng.*, pages 203–222. Springer, 1996. (Cited in page 18).

[107] J. R. Shewchuk. What is a good linear element? interpolation, conditioning, and quality measures. *Proceedings of the 11th International Meshing Roundtable*, pages 115–126, 2002. (Cited in pages 6, 7).

[108] H. Si. A quality tetrahedral mesh generator and three-dimensional delaunay triangulator. *Weierstrass Institute for Applied Analysis and Stochastic, Berlin, Germany*, 2006. (Cited in pages 18, 262).

[109] M. Soucy and D. Laurendeau. Multiresolution surface modeling based on hierarchical triangulation. *Comput. Vis. Image Under.*, 63(1):1–14, 1996. (Cited in page 186).

[110] H. Wei, L. Chen, and Y. Huang. Superconvergence and gradient recovery of linear finite elements for the Laplace-Beltrami operator on general surfaces. *SIAM J. Numer. Anal.*, 48(5):1920–1943, 2010. (Cited in pages 2, 52, 53, 101, 124).

[111] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, pages 80–83, 1945. (Cited in pages 216, 274).

[112] J. Xu and Z. Zhang. Analysis of recovery type a posteriori error estimators for mildly structured grids. *Math. Comput.*, 73(247):1139–1152, 2004. (Cited in pages 51, 141).

[113] O. C. Zienkiewicz and J. Z. Zhu. A simple error estimator and adaptive procedure for practical engineerng analysis. *Int. J. Numer. Meth. Eng.*, 24(2):337–357, 1987. (Cited in pages 20, 50, 136).

[114] O. C. Zienkiewicz and J. Z. Zhu. The superconvergent patch recovery and a posteriori error estimates. part 1: The recovery technique. *Int. J. Numer. Meth. Eng.*, 33(7):1331–1364, 1992. (Cited in pages 2, 20, 50, 55, 101, 136, 141).

[115] O. C. Zienkiewicz and J. Z. Zhu. The superconvergent patch recovery and a posteriori error estimates. part 2: Error estimates and adaptivity. *Int. J. Numer. Meth. Eng.*, 33(7):1365–1382, 1992. (Cited in pages 2, 20, 50, 55, 101, 136).

Acknowledgements

Now the most difficult part of my Ph.D. thesis begins. During these five years at the Politecnico, I met a lot of people that helped me. First of all I want to thank Prof.ssa Simona Perotto and Prof. Luca Formaggia that introduced me to the very interesting world of mesh generation and adaptation. Another important thank goes to Dr. Hang Si that let me know part of the secrets of edge-flipping and the ENI company that gives me the financial support to get the Ph.D. .

After these special thanks, I do not want to make a cold list of the other people that played an important role in these years at the Politecnico, therefore they will read these few words waiting for their names. I would prefer to describe some of the most funny and interesting "adventures" that we lived together, in order to give new life to these memories. Moreover, I prefer to switch to the Italian language to make this challenging work easier.

In questi anni ho conosciuto persone davvero in gamba, alcune delle quali in grado di raggiungermi inaspettatamente sia in Italia che all'estero e capaci di ritrovare automobili disperse in quel di Milano.

Grazie alla loro preparazione, disponibilità e gentilezza sono state in grado di arricchire la mia conoscenza della matematica, del primo soccorso e della gestione di emergenze antincendio.

Alcune di loro mi hanno permesso di partecipare nuove entusiasmanti esperienze che, fino a pochi anni fa, non avrei mai pensato di poter vivere, come l'insegnamento a una classe universitaria, la preparazione di un esame, la partecipazione e allestimento di conferenze in giro per il mondo.

Di questi anni al Politecnico mi mancheranno moltissime cose. Mi mancheranno gli ovetti per premiare i piccoli successi di ogni giorno, il cioccolato per sollevare il morale dopo o durante una difficile giornata, le feste a tema per occasioni importanti come compleanni, lauree, dottorati o traguardi ben più importanti e anche le cospirazioni fatte prima per l'organizzare questi eventi.

Mi mancherà l'eterna sfida delle 11 in cui si cerca di convincere con qualsiasi

mezzo a disposizione gli amici di “Berlino Est” a partecipare alla pausa caffè e mi rammarico di essere riuscito solo una volta ad ottenere l’amplein grazie a una presentazione alla “Love Actually”!

Sono felice di essere stato nominato a mia insaputa “imperatore Franco I” del “regno di Tender” durante un difficile Natale 2010. Pensandoci bene, questa incoronazione non è stato un evento isolato, ma è stata la goccia che ha fatto traboccare il vaso. Partendo dall’idea di un impero tutto nostro di Tender, negli anni seguenti sono scaturite moltissime fantasie: la nomina di ministri, il conio di una propria moneta, la ricerca di una bandiera e la pianificazione di campagne espansionistiche contro l’edificio Nave e MOX-Off.

Aver dovuto lasciare questo ambiente dopo così tanti anni è stato davvero difficile e confesso che, durante l’ultimo viaggio di ritorno, una lacrimuccia è scappata. Ma, sebbene ci siano stati tanti addii e arrivederci, ho potuto apprezzare che il piacevole ambiente del Tender che abbiamo costruito continua ad esistere anche con i nuovi arrivati e sono particolarmente contento che anche altre persone ne potranno godere.

Oltre a ringraziare i miei “più amici che colleghi” ci terrei anche a ringraziare le segretarie e il reparto tecnico del politecnico che, come i mediani in una squadra di calcio, “fanno uno sporco lavoro, ma qualcuno lo deve pur fare” e che non sono premiati per la loro infinita pazienza nei confronti di studenti con la testa fra le nuvole come me per quanto riguarda scadenze, organizzazione di viaggi, consegna tesi e prenotazione di aule.

A questo punto un doveroso ringraziamento va anche ai miei parenti. Prima di tutto vorrei ringraziare mia zia Orietta che ha dovuto sopportare una non-stop di due giorni di lettura di assurde formule matematiche per la correzione dell’inglese e le prometto che cercherò di migliorare.

Infine vorrei ringraziare tantissimo i miei genitori. Anche per loro ho deciso di scrivere i ringraziamenti della tesi in inglese, cosicché li possano leggere e non mi chiedano di fare un’imbarazzante “lettura ufficiale”. Li ringrazio perché, rispetto a alcune persone che abbiamo incontrato durante il mio corso di studi dalle medie in avanti hanno sempre creduto nelle mie capacità e mi hanno spronato facendomi scoprire risorse che non pensavo nemmeno di avere. Infatti spesso ci capita di dire: “Ci avresti mai pensato che saresti arrivato fino a qui?”. Inoltre, loro più di chiunque altro in questi anni hanno dovuto sopportare i miei alti e bassi dovuti al lavoro che andava un giorno male e il giorno dopo bene, e soprattutto hanno avuto un’infinita pazienza nei giorni della scrittura/correzione della tesi.