Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 124 (2015) 265 – 277

Procedia Engineering

www.elsevier.com/locate/procedia

24th International Meshing Roundtable (IMR24)

Anisotropic Finite Element Mesh Adaptation via Higher Dimensional Embedding

Franco Dassi[a], Hang Si[a], Simona Perotto[b], Timo Streckenbach[a]

[a]*Weierstrass Institute, Mohrenstr. 39, 10117 Berlin, Germany*
[b]*Politecnico di Milano, Piazza Leonardo da Vinci 32, I-20133 Milano, Italy*

Abstract

In this paper we provide a novel anisotropic mesh adaptation technique for adaptive finite element analysis. It is based on the concept of higher dimensional embedding, which was exploited in [1–4] to obtain an anisotropic curvature adapted mesh that fits a complex surface in $\mathbb{R}^3$. In the context of adaptive finite element simulation, the solution (which is an unknown function $f : \Omega \subset \mathbb{R}^d \to \mathbb{R}$) is sought by iteratively modifying a finite element mesh according to a mesh sizing field described via a (discrete) metric tensor field that is typically obtained through an error estimator. We proposed to use a higher dimensional embedding, $\Phi_f(\mathbf{x}) := (x_1, \ldots, x_d,\ s\,f(x_1, \ldots, x_d),\ s\,\nabla f(x_1, \ldots, x_d))^t$, instead of the mesh sizing field for the mesh adaption. This embedding contains both informations of the function f itself and its gradient. An isotropic mesh in this embedded space will correspond to an anisotropic mesh in the actual space, where the mesh elements are stretched and aligned according to the features of the function f. To better capture the anisotropy and gradation of the mesh, it is necessary to balance the contribution of the components in this embedding. We have properly adjusted $\Phi_f(\mathbf{x})$ for adaptive finite element analysis. To better understand and validate the proposed mesh adaptation strategy, we first provide a series of experimental tests for piecewise linear interpolation of known functions. We then applied this approach in an adaptive finite element solution of partial differential equations. Both tests are performed on two-dimensional domains in which adaptive triangular meshes are generated. We compared these results with the ones obtained by the software BAMG – a metric-based adaptive mesh generator. The errors measured in the L_2 norm are comparable. Moreover, our meshes captured the anisotropy more accurately than the meshes of BAMG.

Keywords: anisotropic meshes; mesh optimization; partial differential equations.

1. Introduction

Anisotropic meshes are partitions of a given domain with elements elongated along prescribed directions. They have been shown to be particularly well suited for the interpolation of functions and for numerical modeling characterized by strong directional properties, such as semiconductor device modeling, electrochemical modeling, porous

E-mail address: Franco Dassi, Franco.Dassi@wias-berlin.de, Simona Perotto, simona.perotto@polimi.it, Hang Si, Hang.Si@wias-berlin.de, Timo Streckenbach, Timo.Streckenbach@wias-berlin.de

doi:10.1016/j.proeng.2015.10.138

media flow, fluid dynamics, etc., which exhibit boundary or internal layers of various kinds due to singular perturbation arising, e.g., from gate boundary conditions in semiconductors, reacting surfaces in electrochemical problems, or moving reaction fronts. Compared with the resolution of these layers by isotropically adapted meshes, anisotropic meshes can greatly reduce the involved numbers of degrees of freedom, i.e., of the dimensionality of the discrete system, and therefore of the computational effort. The quality of the mesh is essential for the accuracy of the solution. In particular, one can expect superconvergence effects for properly aligned anisotropic meshes.

Anisotropy means the way distance and angles are distorted. It is well understood that the anisotropy can be described through a field $\mathcal{M}$ of metric tensors associated with a space domain $\Omega \subseteq \mathbb{R}^d$, where each metric tensor $M(\mathbf{x}) \in \mathcal{M}, \mathbf{x} \in \Omega$ is a $d \times d$ symmetric positive definite matrix. Given an open curve $C \subset \Omega$, the length of C with respect to $\mathcal{M}$ is defined as: $l_{\mathcal{M}}(C) = \int_{t=0}^{1} \sqrt{\mathbf{v}(t)^t M(c(t))\mathbf{v}(t)}\mathrm{d}t$, where $c(t) : \mathbb{R} \to \mathbb{R}^d, t \in (0,1)$ denotes a parameterization of C and $\mathbf{v}(t) = \partial c(t)/\partial t$ is the tangent vector. Then, the geodesic distance $d_{\mathcal{M}}(\mathbf{x}, \mathbf{y})$ between two points $\mathbf{x}, \mathbf{y} \in \Omega$ is defined as the length of the (possibly non-unique) shortest curve C that connects $\mathbf{x}$ and $\mathbf{y}$: $d_{\mathcal{M}}(\mathbf{x}, \mathbf{y}) = \min(l_{\mathcal{M}}(C))$.

In the majority of works concerning anisotropic mesh generation, a discrete metric tensor field $\mathcal{M}$ (e.g., defined on the vertices) is used to describe the anisotropic feature of the domain. Then, a uniform mesh (with equal edge length) with respect to the metric tensor field $\mathcal{M}$ is sought. This will produce an anisotropic mesh of that domain. One of the most commonly used approaches is mesh adaption, i.e., given an initial mesh $\mathcal{T}_0$ of the domain Ω, one iteratively updates $\mathcal{T}_i, i = 0, 1, 2, ...$ through local mesh operations, like edge/face swapping, vertex smoothing, vertex insertion/deletion, to obtain the desired anisotropic mesh. This has been shown to be very effective in generating anisotropic meshes, see e.g. [10–13,26,27].

An alternative way to describe anisotropy is to use a higher dimensional embedding $\Phi : \mathbb{R}^d \to \mathbb{R}^n$, where $d < n$ [1]. The map Φ embeds the space Ω into a higher dimensional space $\Phi(\Omega)$ such that the anisotropy in Ω corresponds to an isotropy in $\Phi(\Omega)$, hence an isotropic mesh in $\Phi(\Omega)$ will correspond to an anisotropic mesh in Ω. One can use this embedding to define distances and angles in the higher dimensional space $\Phi(\Omega)$. One example of such embedding on a smooth surface $\Omega \subset \mathbb{R}^3$ is to use the normal field of the surface, i.e., $\Phi : \mathbb{R}^3 \to \mathbb{R}^6$, $\Phi(\mathbf{x}) = (x, y, z, sn_x, sn_y, sn_z)^T$ [1,2], where (n_x, n_y, n_z) denotes the unit normal to Ω at $\mathbf{x}$, and $s \in [0, +\infty)$ is a constant which tunes the amount of anisotropy. This embedding essentially approximates the geodesic lengths in Ω by the Euclidean lengths in $\mathbb{R}^6$. An isotropic mesh of $\Phi(\Omega)$ in $\mathbb{R}^6$, when transformed back into $\mathbb{R}^3$, identifies a curvature-adapted anisotropic surface mesh of Ω. This embedding has been successfully used to generated curvature-adapted anisotropic surface meshes [2,3].

In this paper we extend this idea to generate anisotropic meshes for adaptive finite element simulation. The goal is to develop a novel approach for mesh adaptation framework for this application. In a classical adaptive finite element, we start from an initial, usually uniform, mesh and then, to get a proper adapted mesh, we apply this sequence of operations:

$$\texttt{SOLVE}\to\texttt{ESTIMATE}\to\texttt{METRIC}\to\texttt{ADAPT}.$$

The procedure `SOLVE` solves the PDE to get a discrete solution u_h, in `ESTIMATE` the numerical error is estimated based on the actual mesh and u_h. Then, at the step `METRIC` a metric field that suitably employs the informations provided by the error estimate is constructed. Finally, the `ADAPT` procedure is called to re-create the mesh according to the given metric field. This is an iterative procedure and there are different ways to stop this loop. One possibility is to break when a prescribed error bound is obtained, or a maximum number of iterations is reached, or when we get a saturation of the mesh, i.e., the modification done at the step `ADAPT` are “few”, see e.g. [5,6].

Contrary to the classical mesh adaptation procedure, the proposed adaptation strategy in this paper does not involve both the estimation of an error and the construction of a metric field. In each iteration of the mesh adaptation, we use the following steps:

$$\texttt{SOLVE}\to\texttt{RECOVER GRADIENT}\to\texttt{ADAPT},$$

and the process stops when a desired maximum number of iterations is reached. There are different free or commercial software to get the solution of a PDE. In this framework, we use the p∂elib library to have a finite element solution of the problem at hand [24]. At step `RECOVER GRADIENT` we apply a recovery gradient scheme to get the gradient components of the embedding map.

The remind of this paper is organized as following. In Section 2, we describe the higher dimensional embedding proposed in [2] and how we have modified it to achieve a planar triangular anisotropic mesh. In Section 3, we describe the principle behind our gradient recovery strategy used in our adaptive finite element process, which is an important stage for obtaining the gradients that used by the proposed higher dimensional embedding. In Section 4, we explain in detail the mesh adaption procedure. In Section 5, we present experimental results as well as adaptive finite element tests for some published academic examples. The comparison with BAMG - a metric-based adaptive mesh generator, [22,23], is also reported. Finally, conclusions and future works are given in Section 6

2. Higher Dimensional Embedding

It has been shown in [1,2] that the anisotropy is obtained by increasing the dimensions: *an isotropic mesh in a higher dimensional space will correspond to an anisotropic mesh in the lower dimensional space.*, see an example in Figure 1.

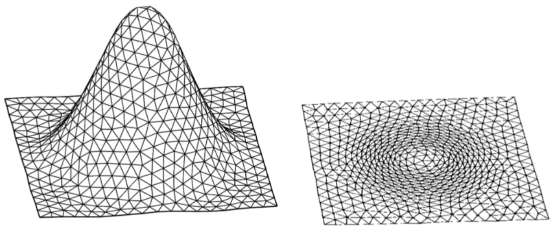

Fig. 1. An isotropic mesh in $\mathbb{R}^3$ (left) and the corresponding anisotropic mesh in $\mathbb{R}^2$ (right). This is a very representative picture of the idea behind the higher dimension embedding proposed in [2].

In [2], given a surface $\Gamma \subset \mathbb{R}^3$, the authors embed it into $\mathbb{R}^6$ by the map, $\Phi : \Gamma \to \mathbb{R}^6$:

$$\Phi(\mathbf{x}) = (x,\ y,\ z,\ s\,n_x,\ s\,n_y,\ s\,n_z)^t\,,$$

where $(n_x, n_y, n_z)^t$ denotes the unit normal to Γ at $\mathbf{x}(x,\ y,\ z)$ and $s \in [0, +\infty)$ is a user-specified constant.

This embedding Φ is an instrumental to get approximation of the geodesic edge lengths in Γ. In fact, where the surface is flat, the lengths of edges remain the same in $\Phi(\Gamma)$. On the contrary, where Γ presents a very high variation of curvature, the lengths of edges in $\Phi(\Gamma)$ become much larger than theirs euclidean lengths measured in $\mathbb{R}^3$. Consequently, since the distances in $\mathbb{R}^6$ are affected by the normals, if we build an isotropic mesh of $\Phi(\Gamma)$ in the embedding space, we will get a curvature-adapted anisotropic mesh of Γ in $\mathbb{R}^3$.

In this paper we are interested in a different task: we desire an anisotropic adapted mesh, where the elements are aligned according to the trend of a known smooth function f or the solution u_h of a partial differential equation (PDE). To better understand the proposed approach, we consider only the case of a smooth function f over a two-dimensional space, then, at the end of this section, we will extend this idea to u_h.

Consider a flat domain Ω with a Lipschitz smooth boundary and a smooth function $f : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$. To proceed with this adaptation procedure, we define the embedding map $\Phi_f : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^5$ as:

$$\Phi_f(\mathbf{x}) := (x,\ y,\ s\,f(x,\ y), s\,g_x(x,\ y),\ s\,g_y(x,\ y))^t\,, \tag{1}$$

here $s \in [0, +\infty)$ is a user-specified parameter, $f(x,\ y)$, $g_x(x,\ y)$ and $g_y(x,\ y)$ are values at the point $(x,\ y)$ of the function f and its gradient components, respectively.

We exploit the standard scalar product in $\mathbb{R}^5$ to have an approximation of the lengths and the angles in this embedded space. Consider three points $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \Omega$, we define the length of the segment $\mathbf{ab}$ in the embedded space as

$$l_{\mathbf{ab}} := \sqrt{\left(\Phi_f(a) - \Phi_f(b), \Phi_f(a) - \Phi_f(b)\right)}\,, \tag{2}$$

where $(\cdot,\cdot)$ is the standard scalar product in $\mathbb{R}^5$. Then, the angle $\widehat{\mathbf{abc}}$ is defined via the cosine:

$$\cos\left(\widehat{\mathbf{abc}}\right) := \frac{\left(\Phi_f(a) - \Phi_f(b), \Phi_f(c) - \Phi_f(b)\right)}{\sqrt{\left(\Phi_f(a) - \Phi_f(b), \Phi_f(a) - \Phi_f(b)\right)}\sqrt{\left(\Phi_f(c) - \Phi_f(b), \Phi_f(c) - \Phi_f(b)\right)}}\,. \tag{3}$$

The computation of these two quantities takes into account some information related to the behavior of the function f. First of all, the term $f(x, y)$ in Equation 1 takes into account the jumps of the function f: if an edge of the mesh crosses a jump of the interpolating function, its length will be longer than the standard euclidean length in $\mathbb{R}^2$. Moreover, the last two terms in Equation 1, g_x and g_y, take into account the trend of the function. Indeed, an edge whose end-points have different gradient will be longer in this embedded space with respect to standard euclidean length in $\mathbb{R}^2$.

The computation of these two quantities is all we need to make an initial mesh as uniform as possible in the embedded space $\mathbb{R}^5$, i.e., to get a final triangular mesh where all the edges have the same target embedded length and the embedded angles are as close as possible to 60°.

Embedding Map for a weak solution of a PDE. When we are dealing with a piece-wise polynomial function, u_h, that comes from the resolution of a PDE, we can exploit a similar embedding map, $\Phi_{u_h} : \mathbb{R}^2 \to \mathbb{R}^5$ defined as

$$\Phi_{u_h}(\mathbf{x}) := (x,\ y,\ s\,u_h(x,\ y),\ s\,g_x(x,\ y),\ s\,g_y(x,\ y))^t\,, \tag{4}$$

where s is a user-specified parameter as before and

$$g_x(x,\ y) := [\nabla u_h\,(x,\ y)]_x\,, \qquad g_y(x,\ y) := [\nabla u_h\,(x,\ y)]_y\,,$$

here $[\nabla u_h\,(x,\ y)]_x$ and $[\nabla u_h\,(x,\ y)]_y$ are the x and y components of the gradient of the discrete solution u_h, respectively. Unfortunately, when we are dealing with the approximation of a piecewise linear solution of a PDE, the gradient of u_h is generally discontinuous across the edges of a mesh element. Indeed, we can get a discontinuous approximation of the gradient that may invalidate the higher dimensional embedding and, consequently, the whole adaptation procedure. To avoid this difficulty, we exploit a gradient recovery procedure. It is described in Section 3.

Modifications of the Embedding Map. From Equation 1 and 4 it is clear that we take into account different quantities, the coordinates of the point in $\mathbb{R}^2$, the function value, and the gradient of the function. These three quantities may have very different ranges. It is necessary to make a suitable scaling factor to make them comparable. More precisely, we select a scaling factor to make each component of the vector in Equation 1 and 4 between [0, 1].

Unfortunately, since the variation on the gradient of a function can be arbitrary large when we are dealing with boundary or internal layers, the normalization on the gradient components may drastically reduce the small variation on the gradient, so that the adaptation procedure can not capture them. To increase the sampling of the mesh in these zones, we modify these two embeddings. For simplicity, we show how we changed the embedding defined in Equation 1, the same variation has be done for the one in 4:

$$\tilde{\Phi}_f(\mathbf{x}) := (\tilde{x},\ \tilde{y},\ s\,\tilde{f}(x,\ y), s\,v_x(x,\ y),\ s\,v_y(x,\ y))^t\,, \tag{5}$$

where $\tilde{x}$, $\tilde{y}$, $\tilde{f}(x,\ y)$ are properly normalized between [0, 1] and

$$v_x(x,\ y) := \operatorname{sign}\left(g_x(x,\ y)\right)\,\sqrt{|\tilde{g}_x(x,\ y)|} \qquad \text{and} \qquad v_y(x,\ y) := \operatorname{sign}\left(g_y(x,\ y)\right)\,\sqrt{\left|\tilde{g}_y(x,\ y)\right|}\,, \tag{6}$$

where $\tilde{g}_x(x,\ y)$ and $\tilde{g}_y(x,\ y)$ are the gradient components normalized and $\operatorname{sign}(\cdot)$ denotes the standard "signum" function. The square root increases the magnitude of the gradient. So that it can effect the computation of the length of the edges and the size of the angles in the embedded space even where the interpolating function presents small variations. In this preliminary study we use the square root, since it was the easiest way to achieve this goal, but other choices can be investigated. In Subsection 5.3, we numerically verify how the error is effected by this adjustment.

3. Gradient Recovery

The gradient of a piecewise linear solution of a PDE can be discontinuous across the edges of the mesh. However, it is possible to proceed with a so called **gradient recovery** procedure that smooths the gradient of the piecewise linear solution u_h [15–17]. This is a common post processing procedure adopted when we might be more interested in computing the gradient rather the function itself. For instance, when we are dealing with an elasticity problem, we can be more interested in computing the stresses and the strains rather than the displacements of the elastic body [25].

In this paragraph we will give a brief description of this gradient recovery techniques, in particular we will focus on the one we used in the embedding map Equation 4. For a more detailed description of them, we refer the reader to [15–17].

There are different ways to compute a smooth gradient moving from a piecewise linear function u_h. We can divide them in two main categories: local averaging and global averaging schemes. The former computes the recovered gradient at a point $\mathbf{x}$ of the domain starting from a neighborhood of $\mathbf{x}$. The latter defines the new smooth gradient at $\mathbf{x}$ via the information provided by the whole domain. In the proposed embedding map we will consider the local averaging schemes, since they provides good results and they are much easier to implement.

Let us consider a planar two dimensional triangular mesh Ω_h, a piecewise linear solution computed on this domain, u_h, and a node of the mesh $\mathbf{x} \in \Omega_h$. Moving from the gradient defined on the triangles that share the node $\mathbf{x}$, we can compute the so-called **simple averaging** to get a smooth value of the gradient at the node $\mathbf{x}$:

$$(\mathbf{G}_h \nabla u_h)(\mathbf{x}) := \frac{1}{m} \sum_{j=1}^{m} \nabla u_h|_{T_j}(\mathbf{x}), \tag{7}$$

where m is the number of triangles that share the node $\mathbf{x}$ and $\nabla u_h|_{T_j}$ is the gradient of the piecewise linear function u_h on the triangle T_j. Under particular hypothesis on the mesh elements, a super-convergence result holds for the simple averaging schemes [19], so this quantity will offer a better approximation of the gradient than the one provided by the finite element solution itself.

4. Mesh Adaptation Procedure

The idea of the proposed re-meshing strategy is to apply the standard mesh modification operations in $\mathbb{R}^2$, but evaluate all the lengths and the angles in the embedded space, see Equation 2 and 3. More precisely, we start from an initial mesh and then we apply the classical mesh modification procedure, such as edge flipping, edge contraction, edge splitting and node smoothing, to make the mesh as uniform as possible in the embedded space. This standard mesh modification operations are widely discussed in the literature. The following paragraphs briefly explain how we apply in this novel mesh adaptation strategy.

Edge Flipping. This operation is the most efficient and effective way to modify a mesh. Consider two triangles **abc** and **bad** that share the edge **ab**, an edge flip will replace the edge **ab** with the edge **cd**, consequently, the triangles **abc** and **bad** will be replaced by **adc** and **bcd**, see Figure 2 left.

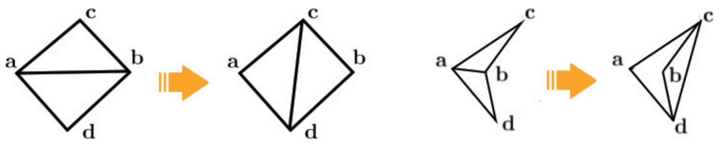

Fig. 2. Flipping of the edge **ab** left, example of an un-flippable edge **ab** due to condition (b), right.

In a triangular planar mesh, it is not always possible to make this operation, the edge **ab** has to satisfy precise criteria to avoid the creation of undesired configuration:

(a) **ab** is not a boundary edge;
(b) neither the angle **dac** nor **dbc** has to be greater than 180°, see Figure 2, right.

Once an edge **ab** satisfies both conditions (a) and (b), we decide to flip it if and only if

$$\theta_{\mathbf{a}} + \theta_{\mathbf{b}} < \theta_{\mathbf{c}} + \theta_{\mathbf{d}}, \tag{8}$$

where θ_* are the embedded angles at the vertex $*$, see Equation 3. We can interpret the condition defined in Equation 8 as a "Delaunay Criteria" in the embedded space. Moreover, to make this operation more effective, we have developed an edge-flip algorithm inspired to the well-known Lawson's flip for the construction of a two-dimensional Delaunay triangulation [20].

Edge Contraction. This is a really important operation when we are dealing with a mesh adaptation procedure since it reduces the number of triangles in the mesh. In the proposed adaptation procedure we will use it to remove all the edges in the mesh that have an embedding length lower than $0.5L$, where L is the target embedded edge length. One of the possible way to remove an edge is via a sequence of 2-by-2 flip and a final 3-to-1 flip. After the edge is removed, we always call the `FlipEdges`() routine to locally improve the mesh.

Edge Splitting. This is the reverse operation of an edge contraction and it is a common way to refine the mesh and, consequently, to increase the resolution of the mesh in the zones of interests. In this framework we will split at its middle point all the edges that have an embedding length greater than $1.5L$. Even in this case, the `FlipEdges`() routine is called in the neighborhood of the new inserted point to locally improve the mesh.

Node Smoothing. Contrary to the previous mesh modification procedure this one does not change the topology of the mesh, but it moves the point to a new location. One of the possible way to compute this new position is

$$\mathbf{x}' = \mathbf{x} + \alpha \sum_{\mathbf{x}_i \in \omega_{\mathbf{x}}} w(d(\mathbf{x}, \mathbf{x}_i))\mathbf{u}_i , \tag{9}$$

here α is a constant, w is a function $w : \mathbb{R} \to \mathbb{R}^+$, $\omega_{\mathbf{x}}$ is the set of vertices that are connected to $\mathbf{x}$, $\mathbf{u}_i$ are the unit vectors that identifies the direction from $\mathbf{x}$ to $\mathbf{x}_i$ and d is the distance between $\mathbf{x}$ and $\mathbf{x}_i$. The choice of α and the function w in Equation 9 determines the smoothing method. In this framework, we adopted the smoothing proposed in [3]: the basic idea is to use the distance d evaluated in the embedded space and the function proposed by F. J. Bossen and P. S. Heckbert in [21] as w.

4.1. The Mesh Adaptation Algorithm

The adaptation procedure has the following inputs: an input function F that can be a smooth known function f or a piecewise linear function u_h, an initial planar triangular mesh of the domain Ω, Ω_h; a user-specified s factor, this input is related to the embedding, the bigger it is the more the triangles will be stretched, a user-specified edge length L, this is the target length in the embedded space, the smaller this length is, the finer will be the resulting mesh, and a `maxIter` number of iterations.

The method applies the sequence of standard mesh operations to get a finial mesh where all the triangles are as close as possible to the equilateral one in the embedded space, i.e., all the sides have length L in the embedded space and all their angles evaluated in the embedded space are as close as possible to 60°. We underline that we use the embedded length of the edges and size of the angles *only* to drive the adaptation procedure. More precisely, we still work in $\mathbb{R}^2$, but we the angles and lengths are evaluated in $\mathbb{R}^6$. This sequence of operation is shown in Algorithm 1.

In the first part of this iterative procedure, we reduce as much as possible the number of vertices in the actual mesh to reduce the computational effort, line 2. Then we split all the edges that have an embedded length $> 1.5L$. At this level, we get a mesh where the edges have embedded length close to the target length L. Finally, we apply edge flipping and node smoothing to improve the measure of the embedded angles. The operation done at line 10 depends on the function we are interpolating: if we are dealing with the interpolation of a known function f, at this level we only recompute the coordinates of the points in the embedded space, while, if we are dealing with the solution of a PDE, we compute the solution on the new adapted mesh and then update the coordinates in the embedded space.

5. Results

In this section we perform and report a series of numerical results to test and validate the proposed mesh adaptation strategy. In Subsection 5.1 we investigate the influence of the s factor on the higher dimensional embedding. In

Algorithm 1 The adaptation procedure for a piecewise linear solution $u_h w$

```
IMPROVE($F$, $\Omega_h$, $s$, $L$)
Data: $F$ the function we are interpolating, $\Omega_h$ the initial mesh, $s$ the user-specified constant for the embedding, $L$ the target embedded length.
 1: for i=1 to maxIter do
 2:     repeat
 3:         contract all the edges such that $l_{\mathbf{ab}} < 0.5L$;
 4:         FlipEdges() on all the edges;
 5:     until an edge is contracted
 6:     split all the edges such that $l_{\mathbf{ab}} > 1.5L$;
 7:     FlipEdges() on all the edges;
 8:     smooth points;
 9:     FlipEdges() on all the edges;
10:     update the embedding map
11: end for
```

Subsection 5.2, we analyze the convergence rate of the error varying the embedded edge length. In Subsection 5.3, we numerically verify the significant role of the square root of the gradient components in the embedding Φ_f in Equation 5. In Subsection 5.4, we apply the the proposed anisotropic mesh adaptation procedure for piecewise linear approximation of some known functions. Finally, in Subsection 5.5, we apply the same procedure on a piecewise linear function provided by the resolution of a PDE. In the last two subsections, we also compared and reported our results with another freely available anisotropic mesh generation software, BAMG [22].

To evaluate the discretization error associated with the meshes, we consider the following quantity

$$e_{tot} := \int_{\Omega_h} |f_h - f|^2 \, dx \,, \tag{10}$$

where Ω_h is piecewise triangular adapted mesh, f_h is the piecewise linear approximation of the function f we are interpolating. Moreover, to have a measure of the stretch of the triangles in the adapted mesh, we compute the quantity

$$\sigma_{max} := \max_{T \in \Omega_h} \sigma_T \,, \tag{11}$$

where σ_T is the so-called stretching factor [5]. If σ_T is close to 1 the shape of the triangle T will be close to the equilateral one, on the contrary, high values of σ_T will correspond to high stretched elements. In all of our tests, we fixed the maximum mesh adapation iteration number `maxIter` := 5.

5.1. $s-$Factor Test

In this test we consider the function $f_1 : [-1,\ 1] \times [-1,\ 1] \to \mathbb{R}$,

$$f_1(x,\ y) := \tanh\left(2(x - y) - 1\right) \,. \tag{12}$$

This function presents an internal boundary layer around the line $2x-2y-1 = 0$, so we will expect that the triangles in the adapted mesh will be stretched along this direction. We fix the target length $L = 0.1$ and we consider the following values of the parameter $s = \{0.5, 1, 5, 10\}$.

The adapted mesh are shown in Figure 3, and the mesh statistics are reported in Table 1. We can see that when s increases, the mesh elements are more stretched (anisotropic) along the internal boundary layer. Moreover, the sampling will be localized where the gradient presents variations.

5.2. Embedding Length Test

In this test case we consider the function $f_2 : [-1,\ 1] \times [-1,\ 1] \to \mathbb{R}$,

$$f_2(x,\ y) := \tanh(60x) - \tanh(60x - 60y - 30) \,. \tag{13}$$

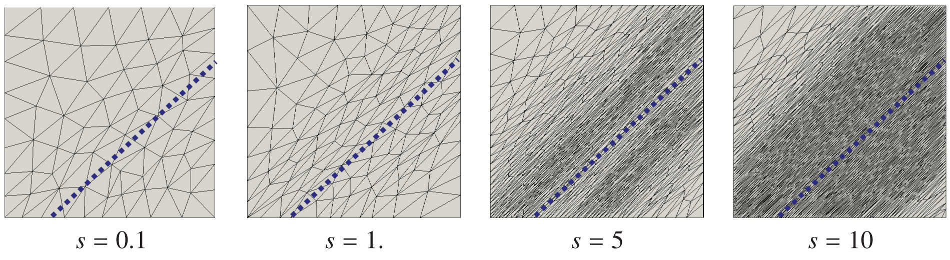

Fig. 3. Adapted mesh with $s = 0.1$, $s = 1.$, $s = 5$ and $s = 10$, respectively. The highlighted dashed line corresponds to $2x - 2y - 1 = 0$.

s	Ele.	e_{tot}	σ_{max}
0.5	121	1.780e-02	3.637e+00
1	407	1.669e-03	2.2116+01
5	1145	2.854e-04	5.3173+01
10	1822	8.739e-05	1.578e+02

Table 1. Statistics of the resulting meshes with different values of s.

We fix the parameter $s = 1.$ and apply the adaptation procedure described in Subsection 4.1, with these different embedded target lengths, $L = \{0.1, 0.5, 0.025, 0.0125\}$. In Figure 4 left, we collect the results obtained and we show the trend of the error, e_{tot}, with respect to these different embedded lengths, Figure 4 right. As it was expected the error decreases by decreasing the target embedded length.

Ele.	L	e_{tot}
622	0.1	2.579e-02
2293	0.05	3.835e-03
8288	0.025	1.033e-03
30032	0.0125	4.1787-04

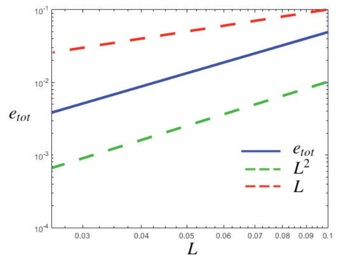

Fig. 4. The numerical data obtained with this example, left, and the trend of the error e_{tot} with respect to the embedded edge lengths, right.

5.3. Numerical Test on the new Embedding

In this example we numerically verify the important role of the square root in Equation 6. We consider two different embedding maps: the one defined in Equation 5 and

$$\tilde{\Psi}_f(\mathbf{x}) := (\tilde{x},\ \tilde{y},\ s\,\tilde{f}(x,\ y),\ s\,w_x(x,\ y),\ s\,w_y(x,\ y))^t\,, \tag{14}$$

where $\tilde{x}$, $\tilde{y}$, $\tilde{f}(x,\ y)$ are properly normalized between $[0,\ 1]$ and

$$w_x(x,\ y) := \text{sign}\,(g_x(x,\ y))\,|\tilde{g}_x(x,\ y)| \qquad \text{and} \qquad w_y(x,\ y) := \text{sign}\left(g_y(x,\ y)\right)\left|\tilde{g}_y(x,\ y)\right|\,, \tag{15}$$

where $\tilde{g}_x(x,\ y)$ and $\tilde{g}_y(x,\ y)$ are the gradient components normalized and sign$\,(\cdot)$ denotes the standard "signum" function. To make this comparison, we use the function f_2, we fix the target embedding length and the factor to $L = 0.05$ and $s = 1.$, respectively. In Figure 5 left, we provide the quantity

$$e_{loc} := \max_{T \in A} \int_T |f_h - f_2|^2\,dx\,,$$

where A is the set of triangles of the adapted mesh shown in Figure 5 right, f_h is the piecewise linear approximation of the function f_2. From Figure 5 right, we highlight that the mesh provided by the embedding $\tilde{\Phi}_f$ is more refined around the layer while, the one given by $\tilde{\Psi}_f$ is more refined inside the layer, but coarser around it. Indeed, where the interpolating function presents "small variations", the embedding $\tilde{\Phi}_f$ offers a better approximation with respect to the one provided by $\tilde{\Psi}_f$. This different behavior is numerically verified via the data in Figure 5, where we can see that the error e_{loc} is lower when we use the embedding $\tilde{\Phi}_f$.

Embedding	e_{loc}
$\tilde{\Psi}_f$	3.157e-03
$\tilde{\Phi}_f$	1.900e-04

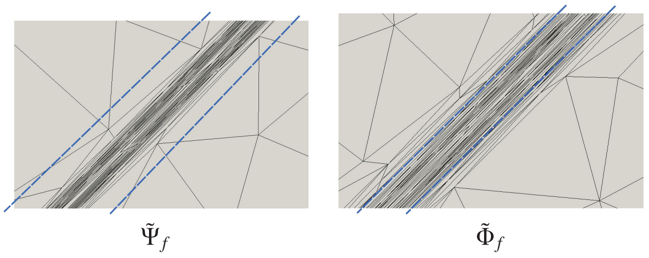

Fig. 5. On the left the quantity e_{loc} computed using the embedding $\tilde{\Psi}_f$, Equation 14, and $\tilde{\Phi}_f$, Equation 5. On the right a detail of the adapted meshes, here we highlight with a dashed strait lines the refined region of the other method.

5.4. Comparison with BAMG

In this subsection we compare the proposed mesh adaptation strategy with the re-meshing procedure of the anisotropic mesh generator BAMG [22]. For this comparison we consider the adaptation with the "`-AbsErr`" flag and we report the values of the "`-err`" flag used [23]. We use the functions f_1 and f_2 of the previous examples and the functions $f_3 : [-1, 1] \times [-1, 1] \to \mathbb{R}$,

$$f_3(x, y) := \sin\left(5\,(x - 0.2)^3\,(y^2 - y + 1)\right),$$

and $f_4 : [0, 1] \times [0, 1] \to \mathbb{R}$,

$$f_4(x, y) := 4\left(1 - e^{-100x} - \left(1 - e^{-100}\right)x\right)y\,(1 - y)\,,$$

We collect all the data in Table 2. We notice that these two mesh adaptation procedure are comparable in terms of number of elements and error e_{tot}, so they offer a similar approximation of the interpolating function at hand. However, the higher dimensional approach make the triangles more stretched than BAMG. In fact, the values of σ_{max} for this adaptation procedure are always greater than the ones provided by BAMG in all the examples.

	BAMG				higher embedding			
function	`-err`	Ele.	e_{tot}	σ_{max}	L	Ele.	e_{tot}	σ_{max}
f_1	1.500e-04	3128	3.644e-04	5.953e+00	0.026	3302	3.720e-04	1.019e+01
f_2	1.000e-04	8106	2.063e-03	1.787e+01	0.025	8288	1.033e-03	4.763e+02
f_3	1.500e-04	9707	3.969e-03	1.836e+01	0.03	9611	3.772e-03	2.2743+05
f_4	1.900e-03	6915	9.255e-04	1.114e+01	0.02	6943	2.436e-04	2.301e+02

Table 2. Comparison between the mesh adapted with BAMG and the proposed re-meshing method with $s = 1$.

5.5. Adaptive finite element applications

5.5.1. A priori test case

Before dealing with the more complex a-posteriori cases, we consider an a-priori case. This test is the same considered in [5,28]. We consider the following PDE: find u such that

$$\begin{cases} -\mu\Delta u = f & \text{in } \Omega\,, \\ u = 0 & \text{in } \partial\Omega\,, \end{cases} \tag{16}$$

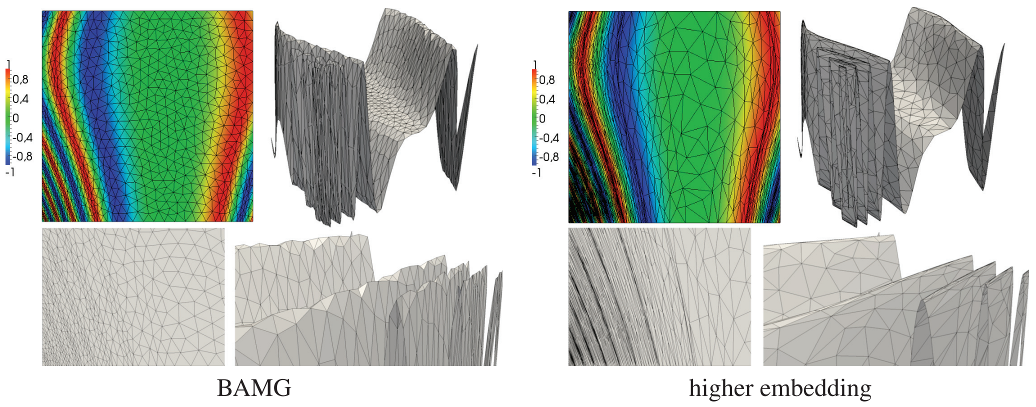

Fig. 6. Example f_3, on the left the adapted mesh with BAMG with a detail, `-err`=0.15, Ele. 1837, $e_{tot} = 2.682e-02$, $\sigma_{max} = 9.319e+00$. On the right the mesh adapted with the embedding procedure with the same detail, $L = 0.07$, Ele. 1764, $e_{tot} = 2.134e-02$, $\sigma_{max} = 2.860e+03$. We show the 3d representation of the function f_3 on the whole domain and in the porposed detail.

here $\mu = 1.$, $\Omega = [0,\,1] \times [0,\,1]$ and

$$f(x,\,y) := 4\alpha^2\,(y - y^2)\,e^{-\alpha x} + 8(1 - e^{\alpha x} - (1 - e^{-\alpha})\,x))\,,$$

where we chosen $\alpha = 100$. The analytical solution of Equation 16 is f_4 and it exhibits an exponential layer along the $x = 0$ boundary with an initial steepness of α. Since we have the exact solution of this PDE, we can still use Equation 10 to evaluate the error. In Table 3, we collect the numerical results. Even in this example the embedding adaptation procedure offers a result comparable to the one provided by BAMG, but the triangles in the latter approach are more stretched than the ones obtained by the former.

	BAMG	higher embedding
Ele.	6866	6159
e_{tot}	9.331e-04	6.098e-03
σ_{max}	9.062e+00	4.500e+03

Table 3. Comparison between the mesh adapted with BAMG and the proposed re-meshing method with `-err` 0.0019 and $L = 0.018$ for the two mesh adaptation process, respectively.

.

5.5.2. *A posteriori test cases*

We apply this new anisotropic mesh adaptation procedure when we are dealing with the solution of a partial differential equation. Since we do not have the exact solution of these PDEs, we consider the solution obtained with a very fine mesh as a reference solution. More precisely we compute:

$$e_{tot} := \int_{\Omega_h} |u_h - u_{ref}|^2\,dx\,, \tag{17}$$

where Ω_h is the triangular mesh of the reference solution u_{ref}, u_h is the piecewise linear solution of the PDE defined on the adapted mesh.

The "double ramp" example. We consider the scalar advection-diffusion problem with homogeneous Dirichlet boundary conditions, [8]: find u

$$\begin{cases} -\mu\Delta u + \overrightarrow{\beta} \cdot \nabla u = 1 & \text{in } \Omega\,, \\ u = 0 & \text{in } \partial\Omega\,, \end{cases} \tag{18}$$

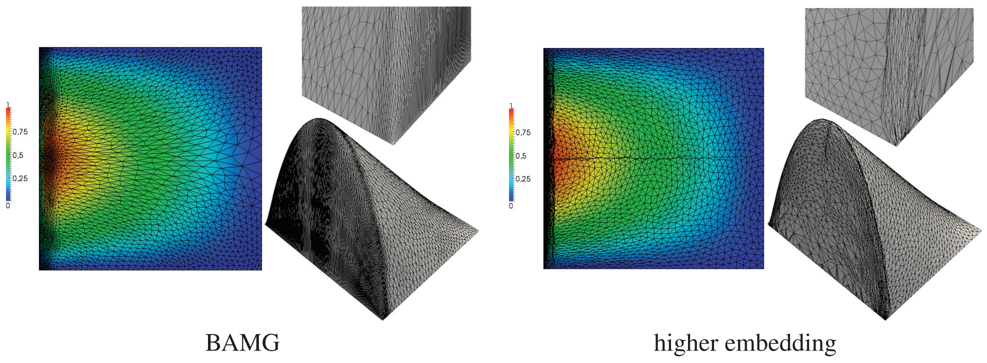

Fig. 7. The Example of this Subsubsection 5.5.1, on the left the adapted mesh with BAMG. On the right the mesh adapted with the embedding procedure. We show the 3d representation of the function on the whole domain and in a detail.

here $\mu = 0.001$ and $\overrightarrow{\beta} = (1, 0)^t$. The domain Ω is an L-shaped region contained in a square of edge length equal to 4.

In Figure 8 we show the resulting adapted meshes. The triangles are perfectly aligned according to the trend of u_h, Figure 8 right and Figure 9. Then, in Table 4, we collect the result obtained with this new adaptation procedure and BAMG. As in the test cases of Subsection 5.4, these two methods are comparable in terms of error, but in the higher dimensional embedding we get more stretched elements, see Figure 9.

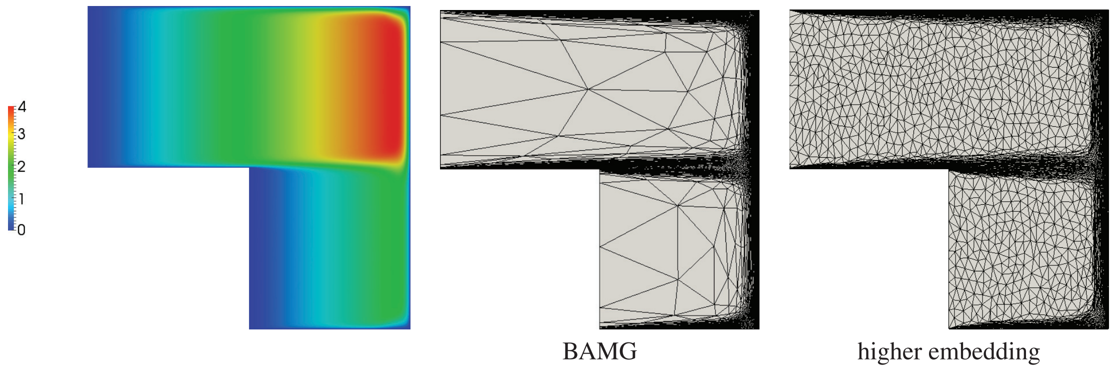

Fig. 8. The reference solution, left, the adapted mesh with BAMG, middle, the one obtained with the higher dimensional embedding, right.

	BAMG	higher embedding
Ele.	7206	7145
e_{tot}	6.167e-03	9.122e-03
σ_{max}	1.435e+02	7.965e+03

Table 4. Comparison between the mesh adapted with BAMG and the proposed re-meshing method with `-err` 0.0033 and $L = 0.0278$ for the two mesh adaptation process, respectively.

The channel test case. We consider the scalar advection-diffusion problem, [8]: find u

$$\begin{cases} -\mu\Delta u + \overrightarrow{\beta}\cdot\nabla u = 0 & \text{in } \Omega\,, \\ u = 1 & \text{in } \partial\Omega_1\,, \\ u = 0 & \text{in } \partial\Omega_2\,, \\ \mu\frac{\partial u}{\partial n} = 0 & \text{in } \partial\Omega_3\,, \end{cases} \tag{19}$$

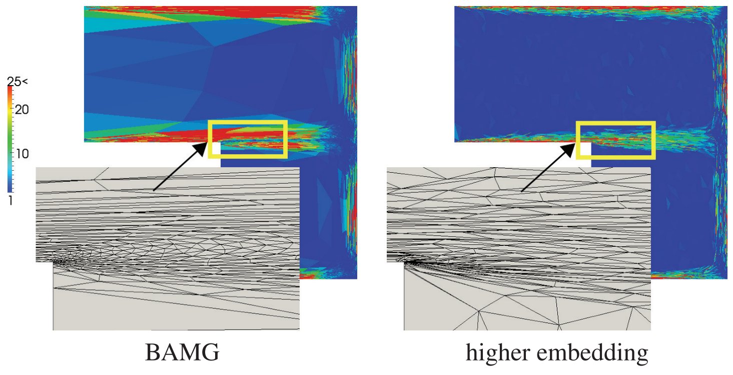

Fig. 9. The stretching factor on the adapted mesh with BAMG, left, the one obtained with the higher dimensional embedding, right.

here $\mu = 0.05$, $\overrightarrow{\beta} = (x, -y)^t$ and $\partial u/\partial n$ is the normal derivative of u along the boundary of Ω. The domain Ω is the same as the previous example: the boundary $\partial\Omega_1$ corresponds to the edge $\{x = 0\}$ of the L-shaped domain, then edges on $\{x = 4\}$ and $\{y = 0\}$ are $\partial\Omega_3$, the other ones are $\partial\Omega_2$.

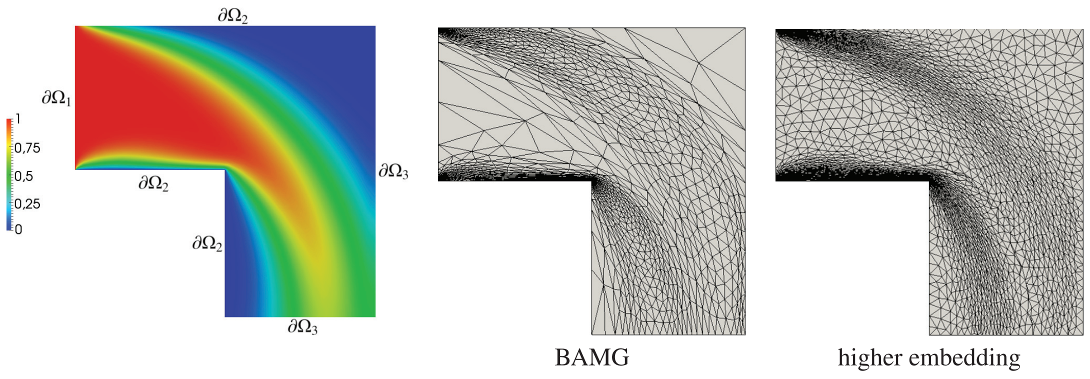

Fig. 10. The reference solution, left, the adapted mesh with BAMG, middle, the one obtained with the higher dimensional embedding, right.

	BAMG	higher embedding
Ele.	4438	4337
e_{tot}	4.143e-03	6.650e-03
σ_{max}	6.880e+01	3.456e+02

Table 5. Comparison between the mesh adapted with BAMG and the proposed re-meshing method with `-err` 0.0165 and $L = 0.028$ for the two mesh adaptation process, respectively.

The triangles are stretched and aligned to the layers of the solution u_h, see Figure 10 right. Moreover, the error is comparable to the one of BAMG, while the triangles are more stretched in the higher embedding adapted mesh.

6. Conclusions and Future Work

In this paper we presented a novel method to get an anisotropic mesh, where the mesh elements are aligned according to the trend of the interpolating function f, or the piecewise linear finite element solution of a PDE, u_h. This is an extension to function interpolation of the higher dimensional embedding method proposed in [1,2].

Even if the results obtained are at least comparable to the ones provided by BAMG, it would be necessary a further analysis on this new approach. More precisely, we may find a way to reduce the sampling in the region where the actual function is flat. Moreover, even if we empirically verify that few iterations are enough to get a good anisotropic adapted mesh, it could be better to find a more rigorous criteria to stop the adaptation procedure.

However, the results in the two dimensional case allow a possible application to the three dimensional case, i.e., when we are dealing with a function defined in a volume. Another interesting application can be the interpolation of function defined on a surface in the three dimensional space and a goal-oriented mesh adaptation procedure, but in both these cases a deeper analysis on the embedding map has to be done.

Acknowledgements. The authors would like to thank the "Deutscher Akademischer Austauschdienst" and, in particular, its academic exchange service that makes possible this research.

References

[1] G. D. Cañas and S. J. Gortler. Surface remeshing in arbitrary codimensions. *The Vis. Comp.*, 22(9-11):885–895, 2006.
[2] B. Lévy and N. Bonneel, Variational anisotropic surface meshing with Voronoi parallel linear enumeration, Proceedings 21st International Meshing Roundtable, 2012, pp. 349–366.
[3] F. Dassi and H. Si, A curvature-adapted anisotropic surface re-meshing method, Tetrahedron IV Proceedings, 2013, to appear.
[4] F. Dassi A. Mola and H. Si, Curvature-adapted Remeshing of CAD Surfaces, Proceedings 23rd International Meshing Roundtable, Proc. Eng., 2014, 82, pp. 349–366.
[5] L. Formaggia and S. Perotto. New anisotropic a priori error estimates. *Numer. Math.*, 89(4):641–667, 2001.
[6] L. Formaggia and S. Perotto. Anisotropic error estimates for elliptic problems. *Numer. Math.*, 94(1):67–92, 2003.
[7] 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.
[8] 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.
[9] 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.
[10] 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.
[11] V. Selmin and L. Formaggia. Simulation of hypersonic flows on unstructured grids. *Int. J. Numer. Meth. Eng.*, 34(2):569–606, 1992.
[12] 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.
[13] J. Peraire, M. Vahdati, K. Morgan, and O. Zienkiewicz. Adaptive remeshing for compressible flow computations. *J. Comput. Phys.*, 72(2):449 – 466, 198.
[14] F. Alauzet, Adaptation de maillage anisotrope en trios dimensions. Application aux simulations instationnaires en mécanique des fluides *Ph.D. Thesis Univerité Montpellier II*, 2003.
[15] 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.
[16] 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.
[17] 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.
[18] P. Frey and H. Borouchaki. Surface meshing using a geometric error estimate. *Int. J. Numer. Methods Engng.*, 58(2):227–245, 2003.
[19] J. Xu and Z. Zang. Analysis of recovery type a posteriori error estimators for middly structured grids. *In Math. Comp.* **73**: pp. 1139–1152, 2004.
[20] C. L. Lawson. Software for C^1 surface interpolation. *Mathematical Software III, Academic Press*, pages 164–191, 1977.
[21] F. J. Bossen and P. S. Heckbert. A pliant method for anisotropic mesh generation. In *5th Intl. Meshing Roundtable*, pages 63–74. Citeseer, 1996.
[22] F. Hecht. New development in FreeFem++. In *J. Numer. Math.*, 20(3-4):251–265, 2012.
[23] F. Hecht. BAMG: Bidimensional anisotropic mesh generator. www.ann.jussieu.fr/hecht/ftp/bamg.
[24] J. Fuhrmann, H. Langmach, T. Streckenbach and M. Uhle. http://www.wias-berlin.de/software/pdelib.–
[25] O. C. Zienkiewicz, R. L. Taylor and J. Z. Zhu. The Finite Element Method: Its Basis and Fundamentals, Elsevier, 6th Edition, 2005.
[26] P. Frey and F. Alauzet. Anisotropic mesh adaption for CFD computations. *Comput. Methods Appl. Mech. Engrg.*, 194:5068–5082, 2005.
[27] C. Dobrzynski and P. Frey. Anisotropic Delaunay mesh adaptation for unsteady simulations. In *17th International Meshing Roundtable*, 2008.
[28] G. Kunert. A posteriori error estimation for anisotropic tetrahedral and triangular grids: Some numerical results using adaptive techniques *Ph.D. thesis*, Von der Frakultät für Mathematik der Technischen Universität Chemnitz, 1999.