Available online at www.sciencedirect.com
SCIENCE DIRECT®
Comput. Methods Appl. Mech. Engrg. 194 (2005) 5068–5082
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Anisotropic mesh adaptation for CFD computations

P.J. Frey [a,*], F. Alauzet [b]

[a] Laboratoire Jacques Louis Lions, Université Pierre et Marie Curie (Paris VI), 175 rue de Chevaleret, F-75252 Paris Cedex 05, France
[b] INRIA, Projet Gamma, Domaine de Voluceau, Rocquencourt, BP 105, F-78153 Le Chesnay Cedex, France

Received 8 December 2003; received in revised form 24 June 2004; accepted 26 November 2004

Abstract

Unstructured mesh adaptation is now widely used in numerical simulations to improve the accuracy of the solutions as well as to capture the behavior of physical phenomena. In this paper, we propose a general purpose error estimate based on the interpolation error that produces an anisotropic metric map used to govern the mesh element creation. Application examples of CFD computations are given to emphasize the efficiency of this approach.

Keywords: Anisotropic mesh generation; CFD; Metric; A posteriori estimate

1. Introduction

Nowadays, in the context of numerical simulations based on finite elements (or finite volumes) methods, unstructured mesh adaptation has largely proved its efficiency for improving the accuracy of the numerical solution as well as for capturing the behavior of physical phenomena. In principle, this technique allows (i) to substantially reduce the number of degrees of freedom, thus impacting favorably the cpu time and (ii) to compute the numerical solution within a desired accuracy. Both these considerations combined with the always increasing performances of the computers allow to envisage the simulation of complex three-dimensional phenomena on current workstations with a reasonable level of accuracy and without having to resort to concurrent algorithms on parallel architectures.

Schematically, a mesh adaptation method relies on an error estimate that converts the numerical (approximation) error into a piece of information that can be efficiently used by the other stages of the mesh adaptation scheme [3]. More specifically, if we consider *h*-methods, this information corresponds to local

* Corresponding author.
E-mail address: frey@ann.jussieu.fr (P.J. Frey).

0045-7825/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2004.11.025

sizing and stretching constraints for the mesh elements. In other words, there is an a posteriori correlation between the error on the numerical solution and the local mesh element size [4]. The aim is thus to equidistribute the error over the mesh by adjusting locally the element size. By repeating this procedure, a "limit" mesh is obtained that leads to a numerical solution at the desired accuracy.

In this paper, we will focus on the construction of a general purpose error estimate that allows to define an anisotropic metric map [10]. With this map, anisotropic mesh elements can be created that lead to reduce the number of degrees of freedom to solve the problem as compared to the equivalent isotropic mesh (at the same level of accuracy).

Formally speaking, we consider a general PDE problem defined on a bounded domain Ω of $\mathbb{R}^3$ and we denote u the exact solution. The solution u_h is obtained using a classical finite element or finite volume method based on a spatial discretization $\mathscr{T}_h$ of the domain, a mesh. Regarding mesh adaptation, the problem consists in computing the approximation error $e_h^i = u - u_h^i$ between the exact solution and the numerical solution at step i on the mesh $\mathscr{T}_h^i$. Using this error, we are willing to control the generation of a mesh $\mathscr{T}_h^{i+1}$ on which the error e_h^i (computed on this mesh) is bounded by a given tolerance value.

From the practical point of view, the approximation error is usually significantly difficult to quantify and, moreover, is intrinsically dependent of the problem at hand (the analysis must be totally adapted even if the equation slightly changes). Therefore, we advocate an indirect approach to estimate the approximation error. Let $\Pi_h u$ denotes the linear interpolate of u on an element of $\mathscr{T}_h$ and let $\tilde{e}_h = u - \Pi_h u$ represents the interpolation error. Using Cea's lemma (established for elliptic problems, although we will show that this approach works well for hyperbolic problems, too), we know that [13]:

$$\|u - u_h\| \leqslant c\|u - \Pi_h u\|.$$

Hence, it will be possible to control the approximation error by controlling the interpolation error, as the numerical solution converges towards the exact solution as the mesh size vanishes. This being stated, the main stages of this approach concern:

- the construction of a suitable interpolation error estimate,
- the definition of an anisotropic metric map,
- the adaptation of a mesh to the sizing and stretching constraints of the metric map.

All these points will be discussed in the following sections and examples of computational meshes will be supplied to illustrate the efficiency of the proposed approach.

2. An estimate based on the interpolation error

In many engineering applications, one of the most challenging problem is to reduce the number of degrees of freedom while preserving the desired level of accuracy for the numerical solution. Following the analysis suggested by Anglada et al. [2], we will indicate how to bound the interpolation error for piecewise linear simplicial elements in $\mathbb{R}^3$. This bound will be used later to characterize the elements of an adapted mesh. This estimate can be described as a geometric error estimate, as it is independent from the nature of the problem to solve and can be used for various types of equations (Navier–Stokes, advection–diffusion, thermal conductivity, waves propagation, etc.).

2.1. An upper bound on the interpolation error

As mentioned before, we used an indirect approach to measure the error $\|u - u_h\|$ [1]. For elliptic problems, it has been proved (Céa's lemma [13]) that the approximation (finite element) error can be bounded by the

interpolation error $e_i : \|e_i\| \leqslant c\|u - \Pi_h u_i\|$, where $\Pi_h u_i$ is the interpolate of u on the mesh $\mathscr{T}_i$, $\|\cdot\|$ is a norm of $\mathbb{R}^3$ and c a constant independent of the current mesh $\mathscr{T}_i$. Here, we assume that this relation still holds in the class of problems envisaged (e.g., hyperbolic). Actually, similar analysis based on the interpolation error show (practically) that the link between the interpolation error and the approximation error is even stronger than the bound given by Céa's lemma. The interpolation error is indeed a reasonable way of defining an error estimate according to [15]. As the analysis and construction of a two-dimensional error estimate has been thoroughly detailed in a research report [1], we will then focus on the three-dimensional case here.

We consider a simplicial unstructured mesh with the following notations:

- $K = [a, b, c, d]$ is a tetrahedron, such that its diameter is not too small (i.e., $h_{\max} > \varepsilon$),
- $u : \mathbb{R}^3 \to \mathbb{R}$ is a (regular) function that represents the solution of our problem,
- $\Pi_h u$ is the interpolate of u on the element K defined by the parameterization $\Pi_h u = (1 - \lambda - \mu - \nu)u(a) + \lambda u(b) + \mu u(c) + \nu u(d)$, with $0 \leqslant \lambda + \mu + \nu \leqslant 1$,
- u and $\Pi_h u$ are supposed to coincide at the vertices of K.

The aim is to bound the error $u - \Pi_h u$ on K. Following the analysis in two-dimensions, we use a Taylor expansion with integral rest of the function e at a vertex of K (for instance a) with respect to any interior point x in K:

$$(u - \Pi_h u)(a) = (u - \Pi_h u)(x) + \langle \vec{xa}, \nabla(u - \Pi_h u)(x)\rangle + \int_0^1 (1-t)\langle \vec{ax}, H_u(x + t\vec{xa})\,\vec{ax}\rangle \,\mathrm{d}t, \tag{1}$$

where $\nabla u(x)$ (resp. $H_u(x)$) denotes the gradient (resp. Hessian) of the variable u at point x. Actually, as we assume that the maximal error is achieved at the point x (closer to a than to b, c or d), then:

$$\nabla(u - \Pi_h u)(x) = 0$$

and this is equivalent to

$$\langle \vec{v}, \nabla(u - \Pi_h u)(x)\rangle = 0, \quad \forall \vec{v} \subset K.$$

Thus, if $e(x)$ stands for the error, we have:

$$|e(x)| = \left| \int_0^1 (1-t)\langle \vec{ax}, H_u(x + t\vec{xa})\,\vec{ax}\rangle \,\mathrm{d}t \right|.$$

Let a' represents the point corresponding to the intersection of the line ax with the face opposite to a. It exists a real number λ such that $\vec{ax} = \lambda\vec{aa'}$. As a is closest to x than any other vertex of K, then $\lambda \leqslant 3/4$. This yields to write:

$$\begin{aligned} |e(x)| &= \left| \int_0^1 (1-t)\lambda^2 \langle \vec{aa'}, H_u(a + t\vec{xa})\vec{aa'}\rangle \,\mathrm{d}t \right|, \\ &\leqslant \frac{9}{16} \max_{y \in aa'} \left| \langle \vec{aa'}, H_u(y)\vec{aa'}\rangle \right| \left| \int_0^1 (1-t)\,\mathrm{d}t \right| \end{aligned} \tag{2}$$

and then:

$$|e(x)| \leqslant \frac{9}{32} \max_{y \in K} \left| \langle \vec{aa'}, H_u(y)\vec{aa'}\rangle \right|.$$

Finally, we can introduce the L_∞ norm of the interpolation error:

$$\|u - \Pi_h u\|_{\infty,K} \leqslant \frac{9}{32} \max_{y \in K} \left| \langle \vec{aa'}, H_u(y)\,\vec{aa'}\rangle \right|.$$

At this point, it can be of practical interest to introduce a metric tensor in the previous relationship. To this end, the 3×3 Hessian matrix being symmetric, its absolute value can be decomposed as

$$|H_u| = \mathscr{R}|\Lambda|\mathscr{R}^{-1}, \text{ with } |\Lambda| = \begin{pmatrix} |\lambda_1| & 0 & 0 \\ 0 & |\lambda_2| & 0 \\ 0 & 0 & |\lambda_3| \end{pmatrix}, \tag{3}$$

where $\mathscr{R}$ is the eigenvectors matrix and $\{\lambda_i\}_{i=1,\ldots,3}$ denote the eigenvalues of this matrix. We obtain the following bound:

$$\|u - \Pi_h u\|_{\infty,K} \leqslant \frac{9}{32} \max_{y\in K} \left\langle \vec{aa'}, |H_u(y)|\vec{aa'} \right\rangle. \tag{4}$$

Notice that the previous relationship is not very useful in practice as the bound depends on the extermum x that is not known a priori. However, it can be reformulated as follows [1]:

$$\|u - \Pi_h u\|_{\infty,K} \leqslant \frac{9}{32} \max_{y\in K} \max_{\vec{v}\subset K} \langle \vec{v}, |H_u(y)|\vec{v}\rangle. \tag{5}$$

This expression provides a bound in the case where the maximum error is achieved inside the element K. If the maximum error is obtained on the element face, then we obtain:

$$\|u - \Pi_h u\|_{\infty,K} \leqslant \frac{2}{9} \max_{y\in[a,b,c]} \max_{\vec{v}\subset[a,b,c]} \langle \vec{v}, |H_u(y)|\vec{v}\rangle. \tag{6}$$

Similarly, for a maximum value obtained along an element edge, we have:

$$\|u - \Pi_h u\|_{\infty,K} \leqslant \frac{1}{8} \max_{y\in ab} \left\langle \vec{ab}, |H_u(y)|\,\vec{ab} \right\rangle. \tag{7}$$

In conclusion, relation (5) provides a proper bound for the interpolation error on an element K.

2.2. Numerical computation of the interpolation error

The previous section has led to an upper bound on the interpolation error on the following form:

$$\|u - \Pi_h u\|_{\infty,K} \leqslant c_{\mathrm{d}} \max_{x\in K} \max_{\vec{v}\subset K} \langle \vec{v}, |H_u(x)|\vec{v}\rangle, \tag{8}$$

where c_{d} is constant related to the space dimension. However, this bound is not very useful as the two maximum values that are involved are tedious to compute in practice. Hence, we introduce the edges of K to replace the maximum value associated with all vectors included in K. As any vector $\vec{v}$ of K can be written as linear combination of the edges of K [1], it yields:

$$\forall \vec{v} \subset K, \quad \|\vec{v}\|_{|H_u(x)|} \leqslant \max_{e\in E_K} \|\vec{e}\|_{|H_u(x)|},$$

where E_K is the set of edges of K. This allows us to write the previous bound in the following manner:

$$\|u - \Pi_h u\|_{\infty,K} \leqslant c_{\mathrm{d}} \max_{x\in K} \max_{e\in E_K} \langle \vec{e}, |H_u(x)|\vec{e}\rangle. \tag{9}$$

However, the righ-hand side term remains difficult to compute numerically. To overcome this problem, we assume that it exists a metric tensor $\overline{\mathscr{M}}(K)$ such that:

$$\max_{x\in K} \langle \vec{e}, |H_u(x)|\vec{e}\rangle \leqslant \langle \vec{e}, \overline{\mathscr{M}}(K)\vec{e}\rangle, \quad \forall e \in E_K$$

and such that the region defined by: $\{\langle \vec{v}, \overline{\mathscr{M}}(K)\vec{v}\rangle | \forall \vec{v} \subset K\}$ is minimal in volume. In the following section, we will detail how to compute precisely the metric field $|H_u|$ and how to construct the metric tensor $\overline{\mathscr{M}}(K)$. Consequently, the interpolation error ε_K on an element K is given by the following relationship:

$$\varepsilon_K = c \max_{e \in E_K} \langle \vec{e}, \overline{\mathscr{M}}(K)\vec{e}\rangle. \tag{10}$$

This relation relates the interpolation error to the square of the largest edge length in K (i.e., the diameter of K) with respect to the metric $\overline{\mathscr{M}}(K)$. In other words, controlling the length of the mesh edges will allow to control the interpolation error on the mesh.

Remark 2.1. In the previous analysis, the computation of the interpolation error corresponds to the computation of the distance between a piecewise linear approximation of a Cartesian surface and the underlying surface. This is the justification for the term of *geometric* error estimate.

2.3. Application to mesh adaptation

From the estimation of the interpolation error given by relation (10), we can state that if the interpolation error is fixed, the only variable left is the edge length. Hence, in mesh adaptation, we will try to adjust mesh edges so that the interpolation error will be equidistributed over the adapted mesh. In other words, we are willing to generate an *optimal* mesh, i.e., to obtain the desired error level with a minimum number of degrees of freedom under fixed constraints [4]. We will now explain how to cope with this requirement.

Let ε represents the maximum level of error tolerated on the mesh elements. The mesh edges must be such that:

$$\varepsilon = c\langle \vec{e}, \overline{\mathscr{M}}(K)\vec{e}\rangle, \quad \forall e \in E_K,$$

where E_K is the set of edges of K. Let $\mathscr{M}(K) = \frac{c}{\varepsilon}\overline{\mathscr{M}}(K)$ be the desired metric tensor, the previous relationship leads to have:

$$\langle \vec{e}, \mathscr{M}(K)\vec{e}\rangle = 1, \quad \forall e \in E_K. \tag{11}$$

This relation is equivalent to the definition of the edge length with respect to the metric $\mathscr{M}(K)$. Actually, the Euclidean norm of a vector $\vec{u}$ for a metric $\mathscr{M}$ (i.e., the scalar product) is defined as

$$\|\vec{u}\|_{\mathscr{M}} = \sqrt{\langle \vec{u}, \vec{u}\rangle_{\mathscr{M}}} = \sqrt{{}^t\vec{u}\mathscr{M}\vec{u}}$$

and the distance between two points is then given by

$$d_{\mathscr{M}}(A,B) = l_{\mathscr{M}}(\overrightarrow{AB}) = \|\overrightarrow{AB}\|_{\mathscr{M}} = \sqrt{{}^t\overrightarrow{AB}\,\mathscr{M}\,\overrightarrow{AB}}.$$

Therefore, the relation (11) is equivalent to

$$(l_{\mathscr{M}(K)}(\vec{e}))^2 = 1, \tag{12}$$

which prescribes edges of unit length for any edge of K in order to bound the interpolation error on K by a value ε.

To briefly summarize this analysis in one sentence, we can act that the relation (12) clearly introduces the notion of *unit mesh* as the desired optimal mesh, i.e., the mesh on which the interpolation error is equidistributed and bounded by a tolerance value ε. Still, we have to explain how to construct the metric tensor and the link between this metric tensor and mesh generation algorithms. This is the aim of the next section.

2.4. Construction of the metric tensor

A metric map on a bounded domain Ω is a data of infinite dimension that cannot be used for numerical purposes as such. Hence, we use a discrete approximation of the metric based on the mesh as geometric support. Usually, in the applications envisaged, the metric $\mathscr{M}$ is defined at the mesh vertices rather than at mesh elements. From a discrete metric map defined at the vertices, a continuous metric field can be defined using an interpolation scheme.

Let ε denotes the desired (i.e., tolerated) interpolation error on the mesh elements and let $h_{\min}$ (resp. $h_{\max}$) be the minimal (resp. maximal) edge size for the mesh elements. According to the previous sections, we define a 3×3 metric tensor at the mesh vertices as follows:

$$\mathscr{M} = \mathscr{R}\widetilde{\Lambda}\mathscr{R}^{-1}, \text{ with } \widetilde{\Lambda} = \begin{pmatrix} \widetilde{\lambda}_1 & 0 & 0 \\ 0 & \widetilde{\lambda}_2 & 0 \\ 0 & 0 & \widetilde{\lambda}_3 \end{pmatrix}, \tag{13}$$

and

$$\widetilde{\lambda}_i = \min\left(\max\left(\frac{c|\lambda_i|}{\varepsilon}, \frac{1}{h_{\max}^2}\right), \frac{1}{h_{\min}^2}\right),$$

where $\mathscr{R}$ is the eigenvector matrix and the coefficients λ_i are the eigenvalues of the Hessian matrix H_u, Relation (3), and c is the constant defined in the inequality (9). Notice that this decomposition is made unique by imposing conditions on the eigenvectors. The metric tensor defined by this relation is anisotropic by nature. The principal directions are given by the eigenvectors and the associated sizes are prescribed by the coefficients $\widetilde{\lambda}_i$. Moreover, we have introduced bounds on minimal and maximal sizes in order to avoid unrealistic metric specifications (for instance, elements of infinite size may be required in regions where the solution is linear, i.e., where the gradient is vanishing, leading to a null eigenvalue). The minimal size reveals also useful for explicit schemes as the smallest mesh size defines the time step.

2.4.1. Relative error

The relation (9) gives an absolute bound on the interpolation error. However, for practical reasons, we would rather like to consider a relative bound on this error and thus to define an estimate of the relative error. The reasons for doing so are related to the nature and type of the variables considered for the metric definition and also to the need to combine various variables together. In order to have dimensionless variables, we follow the ideas suggested by [11,22] and introduce a relative error as

$$\left\| \frac{u - \Pi_h u}{|u|_\epsilon} \right\|_{\infty,K} \leqslant c_{\mathrm{d}} \max_{x \in K} \max_{\vec{e} \in E_K} \left\langle \vec{e}, \frac{|H_u(x)|}{|u(x)|_\epsilon} \vec{e} \right\rangle,$$

where $|u|_\epsilon = \max(|u|, \epsilon \|u\|_{\infty,\Omega})$. Moreover, in numerical simulations, solutions vary from several orders of magnitude (e.g., multi-scale phenomena, recirculations, shocks, etc.). It is often difficult to capture the weakest phenomena *via* mesh adaptation, and even harder to do it when, for instance in CFD, shocks are located within the flow region. A local error estimation can overcome this problem. Following the previous idea, the error estimate is also normalized using the local value of the gradient norm of the variable u, weak phenomena can be captured even in presence of strong shocks. To this end, we suggest the following error estimate:

$$\left\| \frac{u - \Pi_h u}{\alpha|u|_\epsilon + \bar{h}\|\nabla u\|_2} \right\|_{\infty,K} \leqslant c \max_{x \in K} \max_{\vec{e} \in E_K} \left\langle \vec{e}, \frac{|H_u(x)|}{\alpha|u(x)|_\epsilon + \bar{h}\|\nabla u(x)\|_2} \vec{e} \right\rangle, \tag{14}$$

where $\bar{h}$ is the diameter (i.e., the length of its largest edge) of element K and $0 < \alpha < 1$.

2.5. Operations on metrics

When several metrics are specified at the same vertex, a single metric tensor must be defined taking into account all given metrics. To this end, a metric intersection procedure is used [1] and briefly recalled here. Let $\mathscr{M}_1$ and $\mathscr{M}_2$ be two metric tensors given at a vertex P (represented by two ellipsoids), the metric tensor $\mathscr{M}_{1\cap 2}$ corresponding to the intersection of $\mathscr{M}_1$ and $\mathscr{M}_2$ must be such that the interpolation error for each variable is bounded by the given tolerance value. To this end, we use the simultaneous reduction of the quadratic forms associated with the two metrics (cf. Fig. 1). The two metric tensors being geometrically represented by the associated ellipsoids $\mathscr{E}_{\mathscr{M}_i}$, the ellipsoid $\mathscr{E}_{\mathscr{M}}$ of maximal volume included in the (geometric) intersection of these two ellipsoids defines the desired metric tensor. Formally speaking, let us consider the set M_{d} of all metric tensors in $\mathbb{R}^d$ and let us define the ellipsoid associated with the metric $\mathscr{M}$ by

$$\mathscr{E}_{\mathscr{M}} = \left\{ M \middle| \sqrt{{}^t\overrightarrow{PM}\,\mathscr{M}\,\overrightarrow{PM}} = 1 \right\}.$$

The metric $\mathscr{M}_{1\cap 2}$ is then defined as

$$\sup_{\mathscr{M}_i \in M_{\mathrm{d}}} \left\{ M \middle| \sqrt{{}^t\overrightarrow{PM}\,\mathscr{M}_i\,\overrightarrow{PM}} = 1 \right\} \subset \mathscr{E}_{\mathscr{M}_1} \cap \mathscr{E}_{\mathscr{M}_2},$$

where the sup symbol stands for the metric having the largest ellipsoid volume.

We now turn on to the following problem: let $\gamma = [P_1P_2]$ be a segment (i.e., a mesh edge) and $\mathscr{M}_1$ and $\mathscr{M}_2$ be two metrics associated with the endpoints of γ. We are looking for an interpolated metric $\mathscr{M}(t)$ defined along the parametrized segment $\gamma(t) = P_1P_2$ for all $t \in [0,1]$. In addition, this metric must vary in a monotonous manner along the segment. The construction of such metric boils down to interpolate the metrics $\mathscr{M}_1$ and $\mathscr{M}_2$. This process allows to define a continuous metric field along the segment. It is easy to see that such procedure will be useful to define a continuous metric field over the whole domain, from the discrete metric map defined at the mesh vertices. Here, we suggest a linear interpolation scheme, for which the metric at point $\gamma(t)$ is given by

$$\mathscr{M}(t) = \left((1-t)\mathscr{M}_1^{-\frac{1}{2}} + t\mathscr{M}_2^{-\frac{1}{2}} \right)^{-2}, \quad 0 \leqslant t \leqslant 1. \tag{15}$$

To find the interpolated metric, we must first express the two metrics in a basis in which both associated matrices are diagonal, i.e., perform the simultaneous reduction of the quadratic forms and apply the previous interpolation scheme. To this end, we consider $\mathscr{N} = \mathscr{M}_1^{-1}\mathscr{M}_2$ and $\mathscr{P} = (e_1e_2e_3)$ the matrix such that its columns are formed by the eigenvectors of $\mathscr{N}$. We compute $(\lambda_i)_{i=1,3}$ and $(\mu_i)_{i=1,3}$ the eigenvalues of $\mathscr{M}_1$ and $\mathscr{M}_2$ in the basis (e_1,e_2,e_3). Then, we define $(h_{1,i} = \frac{1}{\sqrt{\lambda_i}})_{i=1,3}$ and $(h_{2,i} = \frac{1}{\sqrt{\mu_i}})_{i=1,3}$. The metric $\mathscr{M}(t)$ is given by

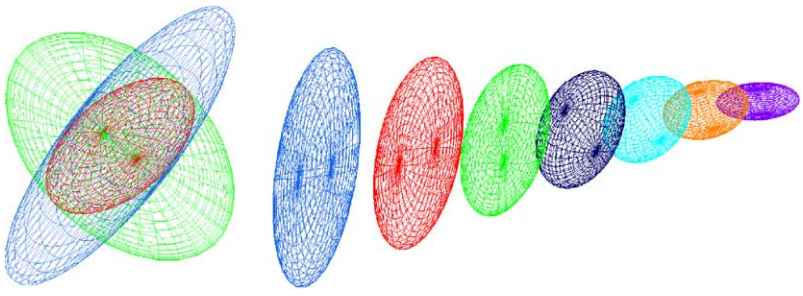

Fig. 1. Intersection of two metric tensors (left-hand side) and metric interpolation along a segment (right-hand side).

$$\mathscr{M}(t) = {}^t\mathscr{P}^{-1} \begin{pmatrix} \frac{1}{H_1^2(t)} & 0 & 0 \\ 0 & \frac{1}{H_2^2(t)} & 0 \\ 0 & 0 & \frac{1}{H_3^2(t)} \end{pmatrix} \mathscr{P}^{-1}, \quad 0 \leqslant t \leqslant 1,$$

where $H_i(t)$, $i = 1,3$, are linear functions such that: $H_i(t) = (1-t)h_{1,i} + th_{2,i}$ (with $H_i(0) = h_{1,i}$ and $H_i(1) = h_{2,i}$).

Remark 2.2. Similar interpolation schemes can be defined to interpolate the metric at a point of a mesh tetrahedra.

3. Anisotropic mesh adaptation

As claimed in the introduction, in many engineering applications, it is desirable to generated anisotropic meshes presenting highly stretched elements in arbitrary directions. Numerous papers have been published on mesh adaptation for numerical simulations in computational solid or fluid mechanics (see [15] for a survey). Among these papers, only a few have addressed the problem of creating unstructured anisotropic meshes [5]. In the following sections, we will detail our approach to anisotropic mesh generation based on the notion of *unit mesh*. Schematically, this approach can be decomposed in two successive steps: at first the surface mesh is adapted using local mesh modifications [17], then the volume mesh is adapted using a constrained Delaunay algorithm extended to the anisotropic case [21]. Both steps are aimed at creating a so-called *unit mesh*.

3.1. Notion of unit mesh

The generation of a (adapted) mesh is based on the specification of a discrete anisotropic metric tensor at each mesh vertex of the current mesh, this tensor allowing to compute the lengths of the mesh edges. The notion of length in a metric space is directly related to the notion of metric and thus to a suitable definition of the scalar product in the vector space considered. Let us briefly recall here that defining a metric tensor at a vertex P consists in defining a 3×3 symmetric positive definite (i.e., non-degenerated) matrix [16]. The scalar product in $\mathbb{R}^3$ is defined according to a metric $\mathscr{M}$ as

$$\langle \vec{u}, \vec{v} \rangle_{\mathscr{M}} = {}^t\vec{u}\mathscr{M}\vec{v} \in \mathbb{R}.$$

(It is indeed a scalar product as the matrix $\mathscr{M}$ is symmetric positive definite.) With this notion, it is easy to define the associated Euclidean norm:

$$\|\vec{u}\|_{\mathscr{M}} = \sqrt{\langle \vec{u}, \vec{u} \rangle_{\mathscr{M}}} = \sqrt{{}^t\vec{u}\mathscr{M}\vec{u}},$$

which actually measures the length of the vector $\vec{u}$ (i.e., the distance between the two endpoints). The tensor $\mathscr{M}$ can be diagonalized as it is symmetric. Hence, $\mathscr{M}$ can be decomposed as: $\mathscr{M} = \mathscr{R}\Lambda\mathscr{R}^{-1}$, where $\mathscr{R}$ is the eigenvectors matrix and Λ is the matrix of the eigenvalues of $\mathscr{M}$. Although this decomposition is usually not unique, it can be made unique by considering specific conditions on the eigenvectors [1].

For the sake of simplicity, it is possible to define the metric tensor so as to prescribe a unit edge length. At each vertex, a different expression of the metric $\mathscr{M}$ leads to a different expression of the scalar product.

Let P be a vertex and let $\mathscr{M}(P)$ be the metric at P. The desired edge PX must have a length close to one with respect to $\mathscr{M}(P)$:

$$l_{\mathscr{M}(P)}(\overrightarrow{PX}) = \sqrt{{}^t\overrightarrow{PX}\,\mathscr{M}(P)\,\overrightarrow{PX}} = 1.$$

As the metric varies in the domain (is not constant in an element), we need to consider the metrics at the edge endpoints as well as all intermediate metrics along the edge. To this end, we use the length of a parametrized segment $\gamma(t)$ (e.g., a mesh edge PX: $\gamma(t) = P + t\overrightarrow{PX}$) as

$$l(\gamma) = \int_0^1 \|\gamma'(t)\|\mathrm{d}t, = \int_0^1 \sqrt{{}^t\gamma'(t)\mathscr{M}(t)\gamma'(t)},$$

where $\mathscr{M}(t)$ stands for the metric at the point $P + t\overrightarrow{PX}$ along the edge PX, thus leading to the following relation:

$$l_{\mathscr{M}}(\overrightarrow{PX}) = \int_0^1 \sqrt{{}^t\overrightarrow{PX}\,\mathscr{M}(t)\,\overrightarrow{PX}}. \tag{16}$$

If we remember that the metric $\mathscr{M}$ has been defined such that $l_{\mathscr{M}}(\gamma) = 1$ (see Section 2.3), the desired adapted mesh is then a *unit mesh*, i.e., a mesh such that for each edge $\vec{e} \in E_K$, $l_{\mathscr{M}}(\vec{e}) \approx 1$ (compare this relation to relation (12)).

3.2. Surface mesh adaptation

In our approach, the surface is assumed to be represented by an initial surface triangulation without any link to a CAD modelling system or an analytical (explicit) parametrization. The reason of such choice has been dictated by the wide range of applications envisaged, including biomedical simulations for which the domain is reconstructed from an initial point cloud [19].

Given a discrete surface (a piecewise linear approximation of the domain boundaries) and a discrete metric field associated with the mesh vertices, the aim is to generate a mesh adapted to this metric map. To this end, the approach we have successfully used consists in modifying iteratively the initial surface mesh so as to achieve a unit mesh. Obviously, as the mesh is intended for numerical (finite element) computations, the mesh gradation is also a major concern [9]. The ingredients to achieve this goal typically include mesh enrichment, mesh coarsening and local mesh optimization procedures. The local mesh modifications operators involved are: edge flipping, edge collapsing, edge splitting and node removal, node repositioning and degree relaxation [14,24].

As no CAD information is supposedly available, an internal (at least) C^1 continuous geometric support is constructed, using local quadrics (defined at the mesh vertices). Then, a geometric metric tensor $\mathscr{G}$ is defined at the mesh vertices via this support, based on the local principal curvatures and directions. The geometric metric $\mathscr{G}$ is defined in the local frame (associated to the tangent plane at a vertex), the third direction corresponding to the local normal to the surface. The analysis of the surface curvatures allows to minimize the deviation between the tangent planes of the piecewise linear interpolation and that of the true (underlying) surface. This means that the local size for the mesh elements must be proportional to the radii of curvature:

$$\mathscr{G} = (\,h_1 \quad h_2 \quad h_3\,)\begin{pmatrix} \alpha\kappa_1^{-2} & 0 & 0 \\ 0 & \beta\kappa_2^{-2} & 0 \\ 0 & 0 & 1 \end{pmatrix}{}^t(h_1 \quad h_2 \quad h_3), \tag{17}$$

where h_1, h_2 and h_3 represent the desired sizes. Actually, this metric tensor has to be intersected with the computational metric $\mathscr{M}$ (when specified), so as to cope at best with the two anisotropic requirements. In turn, this metric $\mathscr{G} \cap \mathscr{M}$ must be modified to account for the desired mesh gradation [9]. The resulting metric $\widetilde{\mathscr{M}}$ is finally used to govern all mesh modifications.

Once the geometric metric has been defined, the surface meshing algorithm is pretty straightforward: edge lengths are computed with respect to the metric $\widetilde{\mathscr{M}}$ and small edge are collapsed while long edge are splitted into unit length segments. Edge flips and node repositioning operations are performed to improve the overall mesh quality (in terms of shape and size) [6,7,17].

3.3. Volume mesh adaptation

Once the surface mesh has been adapted, a unit volume mesh is generated with respect to the modified metric $\widetilde{\mathscr{M}}$. In our approach a constrained Delaunay procedure is used to build first an empty mesh (with no internal vertices). Then, based on an edge length analysis, internal nodes are added into the current mesh (most of them coming from the background mesh, at the previous iteration) using the *Delaunay kernel*, extended to the anisotropic case [20,21].

As pointed out, the classical distance evaluation is replaced using an evaluation related to the local anisotropic metric. Hence, let A and B be two points, the distance between A and B, $d(A,B)$ is now replaced by $l_{\mathscr{M}}(A,B)$ as defined by Eq. (16). Then, O_K (the circumcenter of a tetrahedron) is computed as the solution of the system:

$$l_{\mathscr{M}}(O_K,P_i) = l_{\mathscr{M}}(O_K,P_j) \quad \forall i,j = 1,4, \;\; i \neq j$$

and r_K, the circumradius of K, is computed as

$$r_K = l_{\mathscr{M}}(O_K,P_j).$$

Finally, the Delaunay measure:

$$\alpha_{\mathscr{M}}(P,K) = \frac{l_{\mathscr{M}}(O_K,P)}{l_{\mathscr{M}}(O_K,P_j)} < 1,$$

where $\alpha_{\mathscr{M}}(P,K)$ is used to define the *cavity* of K. However, as we face a non-linear system to compute O_K and as the metric is discrete, it is then not so easy to compute the desired length. Therefore, approximations are needed to return to the Euclidean context. To this end, we fix the metric and various approximations can be used.

Notice that most of the vertices are preserved in order to reduce the round-off and computational errors when interpolating the solutions from one mesh to another in the adaptation scheme.

4. Application examples

To conclude our exposé and to illustrate the efficiency of the proposed approach, we will now present two application examples of three-dimensional CFD simulations. As the construction of the metric tensor used in mesh adaptation requires computing the Hessian of the variable considered for adaptation, we first recall a numerical technique to evaluate the second derivatives of the variable at mesh vertices.

4.1. Evaluation of the Hessian matrix

One of the key point in the mesh adaptation scheme is related to the construction of the metric tensor and, more precisely, to the evaluation of the Hessian matrix (see relation (13)). Indeed, constructing a good

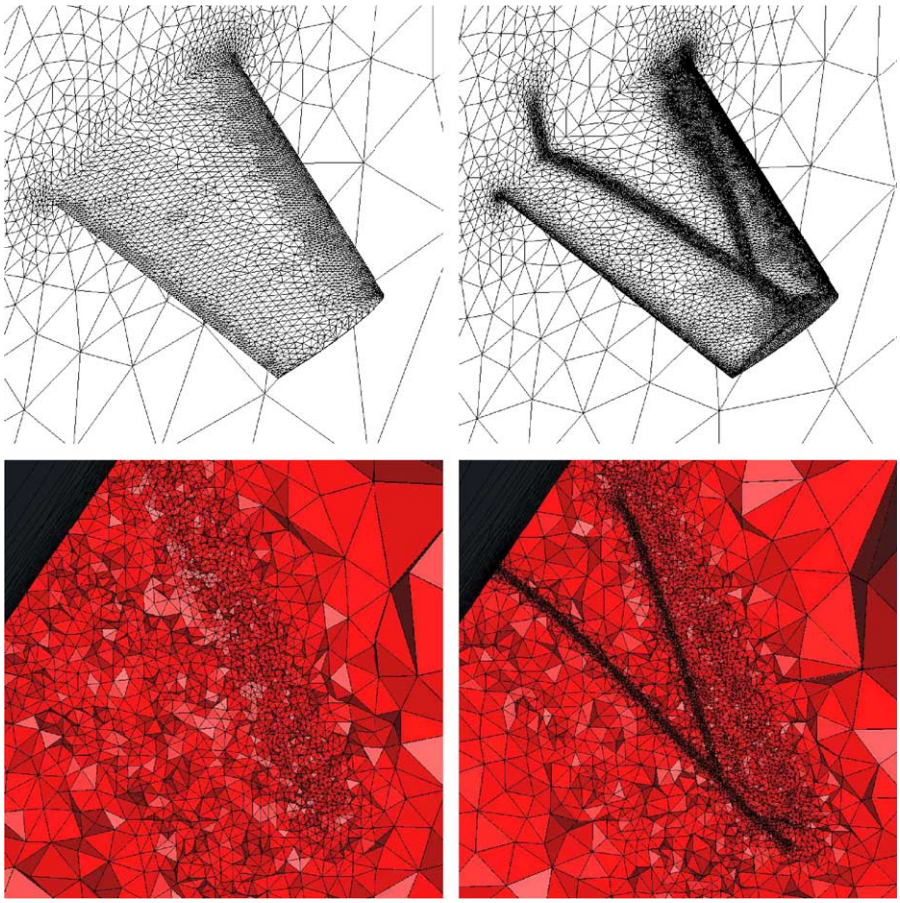

Fig. 2. Onera M6 wing test case: isotropic surface and cut through the volume mesh at iterations 1 and 9 of the adaptation scheme.

(reasonable) metric requires computing the Hessian matrix accurately. Various techniques have been investigated so far, based on a variational (finite element) formulation [25], the Green formula [27] or a Taylor expansion and the solving of a linear system by means of a least-squares approximation. Here, we have considered the last technique and we will briefly recall its main principle.

Let P be a mesh vertex and let u be the solution taken into account to define the metric. By considering a Taylor expansion of u at a vertex P_i connected to the vertex P (i.e., $P_i \in \mathscr{B}(P)$, the ball of P) and truncated at the order 2, we can write the following relation:

$$u_i = u + \overrightarrow{PP_i} \cdot \nabla u(P) + \tfrac{1}{2}\langle {}^t\overrightarrow{PP_i}, H_u(P)\overrightarrow{PP_i}\rangle \iff \tfrac{1}{2}\langle {}^t\overrightarrow{PP_i}, H_u(P)\overrightarrow{PP_i}\rangle = u_i - u - \overrightarrow{PP_i} \cdot \nabla u(P)$$

with the notations $u = u(P)$ and $u_i = u(P_i)$. This relation can be developed using the notations:

$$\overrightarrow{PP_i} = {}^t(x_i y_i z_i), \nabla u(P) = {}^t(\alpha\beta\gamma), H_u(P) = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix},$$

as follows:

$$\tfrac{1}{2}\left(ax_i^2 + 2bx_iy_i + 2cx_iz_i + dy_i^2 + 2ey_iz_i + fz_i^2\right) = u_i - u - (\alpha x_i + \beta y_i + \gamma z_i). \tag{18}$$

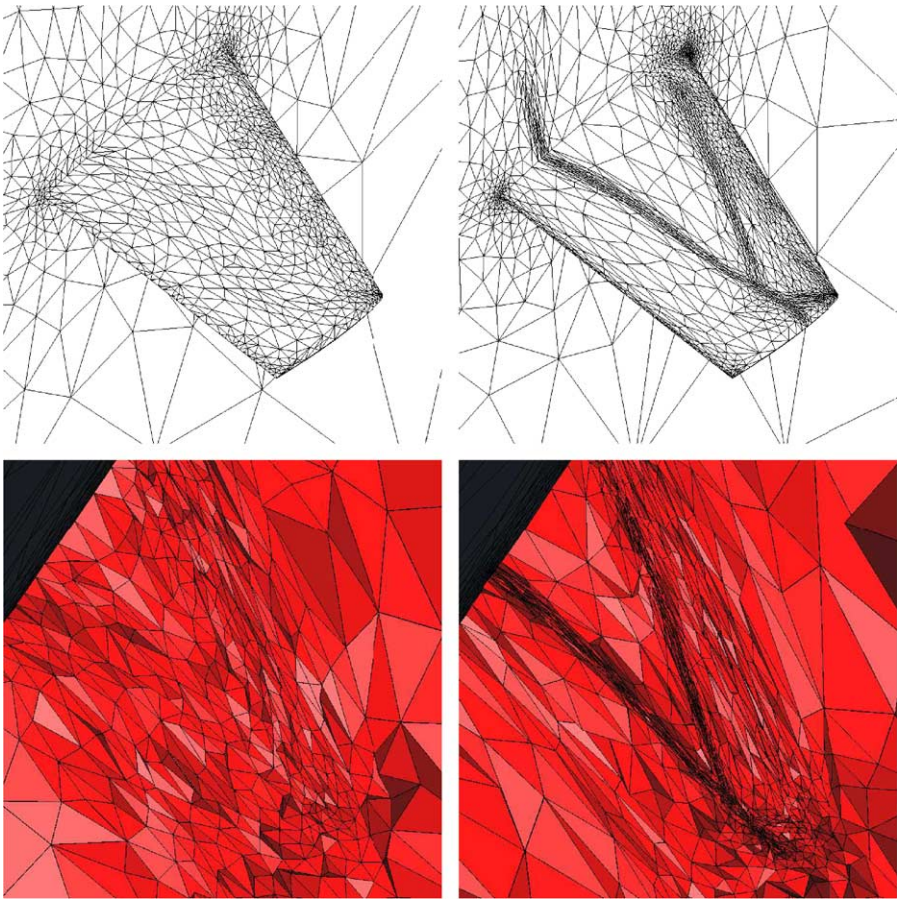

Fig. 3. Onera M6 wing test case: anisotropic surface and cut through the volume mesh at iterations 1 and 9 of the adaptation scheme.

This leads to a usually over-determined system[1] of the form:

$$AX = B, \quad \text{with } {}^tX = (\,a \quad b \quad c \quad d \quad e \quad f\,),$$

where A is a $n \times 6$ matrix ($n = \mathrm{Card}(\mathscr{B}(P))$) function of (x_i, y_i, z_i) and B is a vector of dimension n given by the righ-hand side of the relation (18), and function of $(\alpha, \beta, \gamma, x_i, y_i, z_i, u, u_i)$. This system is solved using a least-square approximation, i.e., it consists in minimizing the distance between the vectors AX and B of $\mathbb{R}^n$ by minimizing the square of the Euclidean norm of their difference. The problem is then to

$$\text{Find } u \in \mathbb{R}^6 \text{ such that } \|AX - B\|^2 = \inf_{Y \in \mathbb{R}^6} \|AY - B\|^2.$$

It can be shown that the solution of this problem is the solution of the linear 6×6 system of normal equations [12]:

$${}^tAAX = {}^tAB.$$

The latter is then solved using a standard Gauss method.

[1] The system is overdetermined as 6 coefficients must be computed and the vertex P is usually connected to more than 6 vertices P_i in three-dimensions.

Remark 4.1. If, in some peculiar cases, the system is under-determined (i.e., $\mathrm{Card}(\mathscr{B}(P)) < 6$), additional vertices connected to the vertices of $\mathscr{B}(P)$ can be taken into account.

4.2. CFD examples

The first example concerns a classic numerical simulation of transonic air flow around the ONERA M6 wing. A Euler solution is computed for Mach number equal to 0.8395 with an angle of attack of 3.06°. This transonic simulation case gives raise to a well-known lambda-shock. The initial mesh is a relatively coarse mesh containing 7815 vertices, 5848 boundary triangles and 37,922 tetrahedra. The variable used to adapt the mesh is the Mach number. The mesh has been adapted 9 times, every 250 time steps. Fig. 2 (resp. 3) shows the adaptation in the isotropic (resp. anisotropic) case. The final isotropic mesh (iteration 9) contains 231,113 vertices and 1,316,631 tetrahedra and the final anisotropic mesh contains 23,516 and 132,676 tetrahedra. In this example, the maximal aspect ratio achieved for the anisotropic elements is about 10. Nevertheless, the anisotropic metric leads to a dramatic reduction of the number of degrees of freedom, roughly one order less than in the isotropic case (for the same error level). The CPU times required to generate the final surface (resp. volume) mesh is 31 (resp. 132) s, and to compute the Euler solution over 250 time steps is

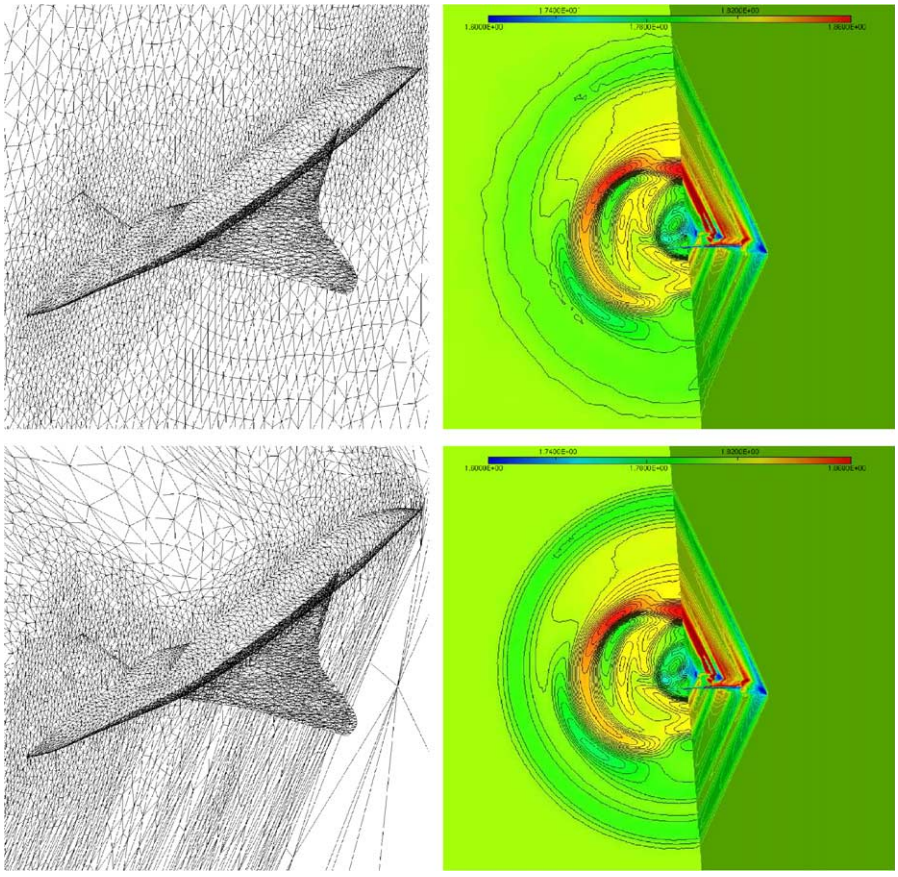

Fig. 4. Supersonic business jet configuration (data courtesy of Dassault Aviation): initial and adapted surface meshes with corresponding isolines of the Mach number.

2,782 s in the isotropic case, on a Pentium 4 3Mhz machine. Similarly, the CPU times required to generate the surface (resp. volume) mesh is 3 (resp. 25) s, and to compute the Euler solution over 250 time steps is 318 s in the anisotropic case (Fig. 3).

The compressible Euler equations have been solved using the NSC3KE software [26]. The surface and volume meshes have been adapted using, respectively, the Yams [18] and Gamanic3d softwares [21].

The second example concerns an Euler simulation of supersonic flow for a future business jet at Mach 1.8 with an angle of incidence of 3° at an altitude of 15,200 m. The design and conception of this future business plane (Dassault Aviation) has led to investigate the control of the sonic boom phenomenon. Beside classical studies aimed at reducing the drag and at increasing the lift of the airplane, the shape optimization process for supersonic civil aircrafts includes another component: the need to reduce the noise at the ground level. In our case, the goal was to analyze the impact on the sonic boom of the optimization of the front part of the airplane geometry. Numerically, this study combine a good near field flow computation (including the computation of the aircraft signature: pressure fields) and a good sonic boom prediction (far field propagation). In the preliminary stage of this project, mesh adaptation already proved to be a very efficient tool. In particular, anisotropic mesh adaptation allows to compute accurately the far-field pressure distribution with a minimal numerical diffusion. Fig. 4 presents initial and adapted meshes for this simulation. In this Figure, the Mach cones are clearly identified in front of the fuselage on the adapted mesh. The initial mesh contains 34,412 vertices, 18,924 boundary triangles and 178,632 tetrahedra. The final mesh (iteration 9) contains 124,320 vertices, 15,792 boundary triangles and 734,046 tetrahedra.

5. Conclusion and future work

In this paper, we have proposed a method for constructing a general purpose error estimate suitable to define an anisotropic metric map. With such a map, adapted meshes can be generated for numerical simulations, for instance in CFD. The error estimate is based on the interpolation rather than on the approximation error. This allows to easily combine metric tensors for variables of different types and natures. Surface and volume mesh generation algorithms have been presented that take advantage of discrete anisotropic requirements. The numerical results presented in this paper have demonstrated the feasibility and efficiency of this approach for solving fluid dynamics problems in an adaptive scheme.

References

[1] F. Alauzet, P.J. Frey, Estimateur d'erreur géométrique et métriques anisotropes pour l'adaptation de maillage. Partie I: aspects théoriques, RR-4759, INRIA Rocquencourt, 2003.
[2] M.V. Anglada, N.P. Garcia, P.B. Crosa, Directional adaptive surface triangulation, Comput. Aided Des. 16 (1999) 107–126.
[3] E.F. D'Azevedo, B. Simpson, On optimal triangular meshes for minimizing the gradient error, Numer. Math. 59 (4) (1991) 321–348.
[4] I. Babuška, W.C. Rheinboldt, A posteriori error estimates for the finite element method, Int. J. Numer. Methods Engrg. 12 (1978) 1597–1615.
[5] T.J. Baker, Mesh adaptation strategies for problems in fluid dynamics, Finite Elem. Anal. Des. 25 (3–4) (1997) 243–273.
[6] R.E. Bank, Mesh smoothing using a posteriori estimates, SIAM J. Numer. Anal. 34 (3) (1997) 979–997.
[7] M. Berzins, Mesh quality: A function of geometry, error estimates or both? Engrg. Comput. 15 (1999) 236–247.
[9] H. Borouchaki, F. Hecht, P.J. Frey, Mesh gradation control, Int. J. Numer. Methods Engrg. 43 (6) (1998) 1143–1165.
[10] H. Borouchaki, D. Chapelle, P.L. George, P. Laug, P.J. Frey, Estimateurs d'erreur géométriques et adaptation de maillage, in: P.L. George (Ed.), Maillage Et Adaptation, Série Mécanique Et Ingénierie Des Matériaux, Méthodes Numériques, Hermès Science, Paris, 2001.
[11] M.J. Castro-Diaz, F. Hecht, B. Mohammadi, O. Pironneau, Anisotropic unstructured mesh adaptation for flow simulations, Int. J. Numer. Methods Engrg. 25 (1997) 475–491.
[12] P.G. Ciarlet, Introduction à L'analyse Numérique Matricielle Et à L'optimisation, Masson, Paris, 1982.

[13] P.G. Ciarlet, Basic error estimates for elliptic problems, in: P.G. Ciarlet, J.L. Lions (Eds.), Handbook of Numerical Analysis, vol. II, Finite Element Methods (Part 1), North Holland, 1991, pp. 17–352.
[14] H.L. deCougny, M.S. Shephard, Surface meshing using vertex insertion, in: Proc. 5th Int. Meshing Roundtable, Pittsburgh, PA, October 10–11, 1996, pp. 243–256.
[15] M. Fortin, Estimation d'erreur a posteriori et adaptation de maillages, Rev. Eur. Élém. Finis 9 (4) (2000).
[16] P.J. Frey, P.L. George, Mesh Generation. Application to Finite Elements, Hermès Science Publ., Paris, Oxford, 2000.
[17] P.J. Frey, About surface remeshing, in: Proc. of 9th Int. Meshing Roundtable, New Orleans LO, USA, 2000, pp. 123–136.
[18] P.J. Frey, Yams: A fully Automatic adaptive isotropic surface remeshing procedure, Rapport Technique INRIA, RT-0252, November 2001.
[19] P.J. Frey, Generation and adaptation of computational surface meshes from discrete anatomical data, Int. J. Numer. Methods Engrg. 60 (2004) 1049–1074.
[20] P.L. George, Improvement on Delaunay based 3D automatic mesh generator, Finite Elem. Anal. Des. 25 (3–4) (1997) 297–317.
[21] P.L. George, `Gamanic3d`, Adaptive anisotropic tetrahedral mesh generator, Technical Report INRIA, 2002.
[22] F. Hecht, B. Mohammadi, Mesh adaptation by metric control for multi-scale phenomena and turbulence, AIAA Paper, 97-0859, 1997.
[24] R. Löhner, Regridding surface triangulations, J. Comput. Phys. 126 (1996) 1–10.
[25] B. Lucquin, O. Pironneau, Introduction Au Calcul Scientifique, Masson, Paris, 1996.
[26] B. Mohammadi, Fluid dynamics computation with NSC2KE—an user-guide, Release 1.0, Technical Report INRIA, RT-0164, 1994.
[27] B. Mohammadi, P.L. George, F. Hecht, E. Saltel, 3D mesh adaptation by metric control for CFD, Rev. Eur. Élém. Finis 9 (4) (2000) 439–449.