Anisotropic Delaunay Mesh Adaptation for Unsteady Simulations

C. Dobrzynski[1] and P. Frey[2,3]

[1] IMB, UMR 5251, Université Bordeaux 1, 351 cours de la Libération, 33405 Talence cedex, France and INRIA Futurs, Scalapplix team
cecile.dobrzynski@math.u-bordeaux1.fr
[2] UPMC Univ Paris 06, UMR 7598, Laboratoire J.L. Lions, F-75005 Paris, France
frey@ann.jussieu.fr
[3] Universidad de Chile, UMI 2807, Centro de Modelamiento Matemático, Santiago, Chile

Summary. Anisotropic mesh adaptation is a key feature in many numerical simulations to capture the physical behavior of a complex phenomenon at a reasonable computational cost. It is a challenging problem, especially when dealing with time dependent and interface capturing or tracking problems. In this paper, we describe a local mesh adaptation method based on an extension of the Delaunay kernel for creating anisotropic mesh elements with respect to adequate metric tensors. In addition, we show that this approach can be successfully applied to deal with fluid-structure interaction problems where parts of the domain boundaries undergo large displacements. The accuracy and efficiency of the method is assessed on various numerical examples of complex three-dimensional simulations.

1 Introduction

In the context of numerical simulations based on variational methods, adaptive and anisotropic triangulations have proven to be very effective for solving complex physical and biomedical problems described by a set of partial differential equations; see for instance [19, 30, 32]. Actually, many applications (e.g., in solid and fluid dynamics, combustion, heat transfer, etc.) require localized regions of the computational domain to have a larger mesh density, *i.e.*, closely-spaced vertexs, to capture the singular or nearly singular solutions that develop in such regions and to resolve large solution variations sufficiently accurately. Solving these equations with a uniform mesh would require a huge number of mesh vertexs, often out of reach of the current computer technology. Indeed, the aim of mesh adaptation is twofold: improving the efficiency of the method for better accuracy and stability at a lower computational expense.

On the other hand, dynamically evolving surfaces arise in numerous computational applications, such as free surfaces in multiphase flows or moving and deforming interfaces in fluid-structure interactions, biomedical surfaces, etc. These applications require or involve potentially large displacements or deformations of the domain geometry in time. Furthermore, this moving geometry is generally

part of the solution of a set of partial differential equations and thus it is not known by or described by an analytical model. Such surfaces can be successfully handled by level set formulations [29] or implicit surfaces. In many cases, it is again more efficient to use anisotropic elements and to adapt the mesh to capture the interface or to follow the severe deformations of the geometry. The goal of this paper is to provide such mesh adaptation features for unstructured simplicial meshes in view of time-dependent and geometry evolving simulations.

In general, anisotropic mesh adaptation aims at equidistributing the approximation error by adjusting locally the mesh density according to a metric tensor field based on the Hessian of the numerical solution [16, 18]. It relies on the ability to control the *size*, the *shape* and the *orientation* of the mesh elements. In addition to improving the accuracy of the solution, anisotropy allows to preserve the order of convergence of the computational schemes [19]. It has already been largely shown that highly stretched mesh elements can interpolate a smooth function much more accurately than an isotropic mesh with regular elements [3, 33, 34]. As we will emphasize here, anisotropic elements also have the advantage of introducing regularity in the approximation of interfaces between physical domains.

The contributions of this paper are the following. First, it provides a general context for anisotropic Delaunay-based mesh adaptation in three dimensions, based on a local point insertion procedure. In this respect, it can be considered as an extension of previous works on anisotropic meshing for complex domains [7, 9, 12, 13, 22, 32]. It can also be related to computational geometry results on Delaunay insertion of Steiner points in a triangulation, sometimes called point placement methods [11, 26]. However, here we provide a straightforward and convergent algorithm to locally adapt the mesh elements to any anisotropic metric tensor. Furthermore, we show how a slight modification of the Delaunay kernel allows us to prevent the generation of badly-shaped elements. Since our approach is based on local mesh modifications, the termination of the algorithm is straightforward, although the resulting triangulation fulfill the Delaunay criterion only locally. Second, this paper explains how to build an anisotropic metric tensor for level set interface tracking, following the ideas of [14]. Then, we show how this method can be efficiently used to resolve fluid-structure interactions and moving mesh problems where the computational domain geometry undergoes large displacements. We demonstrate benchmark and simulation results on rigid-body and on fluid dynamics simulations. For the sake of simplicity however, we will assume in all these simulations that the initial surface mesh is not concerned by adaptation. The local adaptation of surface meshes has been largely discussed in other papers over the last years (surveys can be found in [17, 27], for instance).

The remainder of this paper is organized as follows. In Section 2, we review the main issues of anisotropic mesh adaptation based on Riemannian metric tensors. This notion of an anisotropic metric tensor has been described in a general purpose book [18] and in many research papers. In particular, we outline the definition of a metric tensor, the notion of metric intersection and interpolation. In Section 3, we show how the classical Delaunay mesh generation procedure can

be extended to the anisotropic context. In particular, we show how to compute the Delaunay kernel used for point insertion in the Bowyer-Watson approach to mesh generation. In Section 4, we describe our local mesh adaptation algorithm. Numerical examples of simulations are given in Section 5 to show the efficiency of the approach.

2 Anisotropy and Metric Related Issues

As pointed out already [18, 23], anisotropic meshing is closely related to differential geometry concepts and numerical error estimates. Here, we will briefly review the theoretical material (curvature estimates, metric tensor field, mesh quality) for anisotropic meshing strictly required to understand our approach, as it has already been thoroughly described in various papers [1, 3, 7, 12, 13, 15, 22, 19].

2.1 Basic Definitions

We introduce the following notations: Ω denotes a simply connected open bounded domain in $\mathbb{R}^3$, $\overline{\Omega}$ is the closure of Ω and $|\Omega|$ is the d-dimensional measure or the volume of Ω. We suppose we are given a conforming triangulation T_h on Ω, h representing the characteristic element size. Each element $K \in T_h$ is a closed subdomain of Ω and we assume that $\overline{\Omega} = \bigcup_{K \in T_h} K$ and that the usual finite element requirements are satisfied (*i.e.*, non-overlapping and intersecting elements are disallowed).

A *uniform mesh* of Ω is then a mesh in which all elements are equally-sized and regular. In such case, if $|T_h|$ represents the number of mesh elements and $h_K = diam(K)$ the diameter of K, the size $h = \max\limits_{K \in T_h} h_K$ is given by the relation:

$$h_K \approx \left(\frac{|\Omega|}{|T_h|} \right)^{1/d} \qquad \forall\, K \in T_h\,.$$

A *quasi-uniform* mesh is a mesh for which (i) there exists a constant τ such that

$$\frac{h_K}{\rho_K} \leq \tau \qquad \forall K \in \bigcup_h T_h\,,$$

where, for any open ball $B_i \subset K$, $\rho_K = \sup_i\{diam(B_i)\}$ is the in-diameter of K and (ii) the variation of h is bounded by a constant. Notice that this notion does not assume a constant mesh size over the domain.

2.2 Metric Tensors

Essential to mesh adaptation is the ability to control the *size*, the *shape* and the *orientation* of mesh elements. This specification is usually based on an error estimate or an error indicator. Typically, it uses a matrix-valued field for anisotropic mesh adaptation.

On the continuous level, it is suitable to consider that mesh elements are represented by ellipsoids. In this geometric representation, the size of the element is its volume, the shape is associated with the ratio of the lengths of its semi-axes and the orientation is provided by its principal axis vectors. Then, their control can be achieved by specifiying a metric tensor $M(x)$ to indicate the size, shape and orientation of mesh elements on the whole domain. Here, $M(x)$ is a 3×3 symmetric positive definite matrix, sometimes referred to as a *monitor function* [23]. The function $\rho(x) = \sqrt{\det(M(x))}$ is called the *adaptation function*. This metric tensor is used to generate a quasi-uniform mesh of Ω in the metric related to M. The volume of an element $K \in T_h$ is unitary:

$$\int_K \sqrt{det(M(x))}\, dx = \int_K \rho(x)\, dx \;=\; 1\,, \qquad \forall K \in T_h$$

and corresponds to the discrete formulation:

$$|K|\, \sqrt{det(M_K)} \;= 1\,, \qquad \forall K \in T_h\,,$$

where M_K is an average of $M(x)$ on K. By extension, the length of a curve γ in a metric M given by $M(x)$ for any $x \in \Omega$ is defined as:

$$|\gamma|_M = \int_0^1 \sqrt{\langle \gamma'(t), M(\gamma(t))\gamma'(t)\rangle} dt\,,$$

where $\gamma(t) : [0,1] \to \mathbb{R}^d$ is a parametrization of γ. By analogy, the length of a mesh edge $\mathbf{e}$ is defined as:

$$l_M(\mathbf{e}) = \int_0^1 \sqrt{\mathbf{e}^t M(t) \mathbf{e}}\, dt\,.$$

The edge length value represents also the number of subdivisions of the edge required to match the mesh size prescribed by the metric tensor. Since the metric tensor $M(x)$ is supposed a symmetric positive definite matrix, the spectral decomposition theorem allows to decompose M as:

$$M = P\,\Lambda\,P^t \;=\; \sum_{i=1}^{d} \lambda_i \mathbf{e_i} \mathbf{e_i}^t\,,$$

where the normalized eigenvectors of M are the columns of matrix $P = [\mathbf{e_1}, \ldots, \mathbf{e_d}]$ such that $P\,P^t = P^t\,P = I_d$ and Λ is the diagonal matrix of the eigenvalues $\lambda_i = {h_i}^{-1}$, where the h_i are the sizes in each eigendirection. It is obvious to see that the matrix P prescribes the orientation, and the matrix Λ prescribes the size and shape of any element K.

Metric intersection and interpolation

Metric intersection

Suppose that two metric tensors are specified at a vertex $p \in T_h$. For mesh generation purposes, we would like to deal with a single metric at the vertex. To

this end, we define a metric intersection procedure. Geometrically speaking, it consists in defining the largest ellipsoid E included in the intersection of the two ellipsoids E_1 and E_2 associated with the two metric tensors. From the algebraic point of view, we use the simutaneous reduction of the two underlying quadratic forms to find a basis of the vector space in which the matrices M and N associated with E_1 and E_2 are represented respectively by I_3 and D, a diagonal matrix of $M_3(\mathbb{R})$.

Denoting by $P = \begin{pmatrix} e_1 & e_2 & e_3 \end{pmatrix}$ the invertible matrix of $GL_3(\mathbb{R})$ formed by the eigenvectors $e_i, i = 1, 3$ of the matrix $N' = M^{-1}\,N$ leads to define the intersection matrix $M_\cap$ as: $M_\cap = (P^t)^{-1}\,\Lambda\,P^{-1}$, where Λ is the diagonal matrix with the coefficients $\max(\lambda_i, \mu_i)$.

Metric interpolation

Consider a parametrization of a mesh edge pq as $c : [0, 1] \to \mathbb{R}^3$, $c(t) = (1-t)p+tq$ and two metric tensors M_p and M_q associated with the endpoints. We are looking for the metric tensor at t, hence for a matrix $M(t)$ defined along the segment $c(t)$ for any value of the parameter $t \in [0, 1]$. The definition of this matrix $M(t)$ involves the interpolation of the eigenvalues of the matrices M_p and M_q. This procedure allows one to define a continuous metric field along the segment. To this end, we suggest the following linear interpolation scheme:

$$M(t) = \left((1-t)M_p^{-\frac{1}{2}} + tM_q^{-\frac{1}{2}}\right)^{-2}, \quad 0 \le t \le 1\,. \tag{1}$$

Finding the interpolated metric tensor M_t requires us to express the two matrices in a basis $\{e_i\}$ in which both are congruent to a diagonal matrix and then to deduce the metric tensor at point t. In other words, this scheme is similar to reducing simultaneously the two quadratic forms associated with the metrics.

2.3 Mesh Quality Measures

There are several reasons for assessing a mesh and controlling its quality. In particular, it is useful to know if the mesh elements are aligned with the physical solution, especially in the anisotropic context. Then, in the adaptive context, it is important to know how closely the equiditribution and alignment conditions are satisfied by the mesh. Finally, in three dimensions, a quasi-uniform mesh is not easy to produce if only controlling the lengths of mesh edges. It is well known that *slivers* (null volume) elements may occur and their creation cannot be prevented by simply checking the their edge lengths.

The topics of mesh quality and mesh assessment have been studied in the context of finite element methods in numerous papers. We refer the reader to [2] and the references therein for more details on the related issues. Classical quality measures include minimal angle condition [35] or maximal angle condition [4] as well as aspect ratio considerations. Other works include error estimates to account for the shape of the element and the solution behavior [6]. The adequacy

between anisotropic elements and the anisotropy of the solution can be measured by a *matching function* [25].

Given a metric tensor field $M(x)$, it is natural to define critera to measure how closely the mesh elements are aligned and equidistributed with respect to $M(x)$. For practical reasons, we introduce a single measure to evaluate the quality of an element K (given here in dimension d):

$$Q_{ani}(K) = \alpha_d \frac{\left(\sum_{i=1}^{k} \langle \mathbf{e_i}, M_K \mathbf{e_i} \rangle\right)^d}{|K|\sqrt{\det(M_K)}},$$

where $\mathbf{e_i}$ represents any of the k egdes of K and α_d is a normalisation constant such that $Q_{ani}(K) = 1$ for a regular element. Notice that $Q_{ani} \geq 1$ for all $K \in T_h$ and thus the larger $\max_K Q_{ani}(K)$ is, the less the triangulation T_h matches the metric specifications. This quality measure is able to indicate how well a mesh element match the metric specification, both in terms of size (edge lengths) and of shape (aspect ratio).

In addition, we define an *efficiency index* τ that provides a single scalar value to evaluate how well a mesh complies with the metric requirements:

$$\tau = \exp\left(\frac{1}{ne} \sum_{1 \leq i \leq ne} \ell_M(\mathbf{e}_i)\right), \quad \ell_M(\mathbf{e}_i) = \begin{cases} l_M(\mathbf{e}_i) - 1 \text{ if } l_M(\mathbf{e}_i) < 1 \\ l_M^{-1}(\mathbf{e}_i) - 1 \quad \text{ else} \end{cases} \tag{2}$$

and ne denotes the total number of mesh edges. The objective is to generate meshes for which the efficiency index is close to the optimal value of one. In the simulations, a value $\tau \geq 0.85$ will be considered as an acceptable lower bound.

2.4 Error Estimates

A function $e(h, u_h, f)$ is called an *a posteriori error* if it provides an upper bound on the approximation error: $\|u - u_h\|_W \leq e(h, u_h, f)$. In the context of numerical simulations, the aim is to obtain an anisotropic bound where the physical derivatives are related to the size, the shape and the orientation of mesh elements. Research has been very active these recent years to develop mathematically-based error estimates and several references are provided in the bibliography section of this paper. From the numerical point of view, it is interesting to obtain estimates for the classical L^1 and L^2 norms or the H^1 seminorm. Such estimates have been provided for the interpolation error on linear Lagrange finite elements and involve the eigenvalues and eigenvectors of the Jacobian matrix of the affine mapping between the reference element and a mesh element [15, 31] or are based on the Hessian matrix of the solution [18, 1]. A local error model can be defined at a mesh vertex p as follows:

$$e_M(p) = \sum_{i=1}^{d} h_i^2 \left| \frac{\partial^2 u}{\partial \alpha_i^2} \right|,$$

where the α_i are the coefficients of the diagonal matrix Λ (see above) and the h_i indicate the local sizes in the directions of the eigenvectors of the Hessian matrix. For our purposes, we define a local error indicator the L_∞ norm as:

$$e_K = \|u - P_h u\|_{\infty,K} \leq C \max_{y \in K} \max_{\mathbf{c} \subset K} \langle \mathbf{v}, |D^2 u(y)| \mathbf{v} \rangle \,,$$

where $D^2 u$ represents the Hessian matrix of the function u and C is a constant independent of h.

3 Anisotropic Delaunay Mesh Adaptation

Our approach suggests a modification of the classical point insertion procedure in a Delaunay triangulation to account for the anisotropic metric specifications. At first, we briefly recall the Delaunay kernel in the isotropic context, and then we introduce the modifications in the anisotropic context and for local mesh adaptation.

3.1 The Delaunay Kernel

In the classical isotropic context, mesh points are inserted in an existing Delaunay triangulation using the so-called *Delaunay kernel* [18]. This incremental yet practical method provides an efficient means for constructing a Delaunay triangulation.

Theorem 1. (Delaunay kernel) *Let T_i be the Delaunay triangulation of the convex hull of a set of points $\mathcal{S} \subset \mathbb{R}^d$, and let $p \notin \mathcal{S}$ be a vertex enclosed in T_i. The* Delaunay kernel *procedure can be written as:*

$$T_{i+1} = T_i \setminus \mathcal{C}(p) + \mathcal{B}(p) \,, \tag{3}$$

and provides T_{i+1}, a Delaunay triangulation of the convex hull of $\mathcal{S} \cup \{p\}$.

The proof is obtained by duality with the Voronoï diagram and can be found in [21, 18]. In this fundamental result, $\mathcal{C}(p)$ stands for the *cavity* of point p: the set of simplices in T_i such that their circumspheres contain point p and $\mathcal{B}(p)$ denotes the *ball* of point p: the set of simplices formed by joining p to the external faces of $\mathcal{C}(p)$. Practically, this result ensures that the cavity $\mathcal{C}(p)$ is a star-shaped polytope with respect to point p.

However, in numerical simulations, at least two specific problems arise, related to the necessity of:

i) inserting specific entities in the triangulations (a given set of edges defining the domain boundary, for instance) and
ii) creating additional vertices in the triangulation that are not part of the initial set $\mathcal{S}$ (for instance internal vertices during mesh adaptation).

Regarding the last requirement, the following result provides the existence of a triangulation:

Lemma 1. *Let T_i be an arbitrary triangulation and let $p \in \mathbb{R}^d$ be a point enclosed in T_i, p not being a vertex of T_i. Then, a valid conforming triangulation T_{i+1} having p as vertex can be created using the Delaunay kernel, Theorem 1.*

The proof relies on the star-shaped character of the region $\mathcal{C}(p)$ with respect to point p [18]. This is only possible if the cavity is created incrementally from a *seed* tetrahedron (see above). Notice that due to this restriction, the resulting triangulation T_{i+1} may not be a Delaunay triangulation, although it is conforming to finite elements requirements [10].

Point insertion issues

Defining the cavity $\mathcal{C}(p)$ of a point p requires identifying all simplices having a circumsphere that contains point p. The radius of the sphere circumscribed to a simplex K corresponds to the distance between any vertex p_i of K and the circumcenter O of K: $r_K = d(p_i, O) = \|Op_i\|$. The circumcenter O can be computed as the solution of a linear system of the form:

$$\|Op_1\| = \|Op_i\|\,,\ \forall p_i \in K\,,\ p_i \neq p_1,\ i = 2, \ldots, d\,.$$

Hence, given a point p, an element K belongs to the cavity $\mathcal{C}(p)$ iff the following relation holds:

$$\alpha(p, K) = \frac{\|Op\|}{r_K} < 1\,,$$

where $\alpha(p, K)$ is the *Delaunay measure* of point p with respect to the simplex K. First, we have to find the simplex or, in some peculiar cases, the set $\{K \in T_h\,, p \in K\}$ of all simplices containing p the point to be inserted. Then, by incorporating all adjacent simplices K_i such that $\alpha(p, K_i) < 1$, the set $\mathcal{C}(p)$ is obtained. Each time a new simplex K_i is added to $\mathcal{C}(p)$, the star-shapedness of this set is checked and the simplex is eventually removed if $\mathcal{C}(p)$ loses its property.

In addition, to avoid the generation of badly-shaped elements like *slivers*, we introduced a minimal volume requirement: all new simplices obtained by connecting p to the external faces of $\mathcal{C}(p)$ must have a measure larger than a given lower bound: $|K| > \varepsilon$, with ε set to 1.10^{-5} in the numerical experiments. This simple check has revealed especially and quite surprinsingly efficient in preventing the generation of most of the slivers, thus impacting favorably the optimization stage and the mesh quality histograms (cf. Section 4).

3.2 Anisotropic Delaunay Kernel

As expected, the extension to the anisotropic case consists of introducing a metric tensor, hence a symmetric positive definite matrix M_p at each point $p \in \mathbb{R}^d$. This will allow us to consider the Euclidean norm of any vector in $\mathbb{R}^d$ given the inner product $\langle \cdot, \cdot \rangle_M$. Hence, all distance checks involved in the Delaunay measure will be replaced by length checks according to the given matrix M_p, namely:

$$\alpha_{M_p}(p, K) = \frac{\|Op\|_{M_P}}{(r_K)_{M_p}} < 1\,.$$

The Delaunay kernel (Theorem 1) relies on distance evaluations to define the cavity of the point p to be inserted in an existing Delaunay triangulation. Using the Delaunay measure α_{M_p}, it is possible to define a valid the cavity *i.e.*, star-shaped with respect to p. In [21, 18], we have the following result:

Lemma 2 (anisotropic Delaunay kernel). *Given a metric tensor M_p at a point $p \in \mathbb{R}^d$, the anisotropic Delaunay measure, $\alpha_{M_p}(p, K) < 1$, provides a valid anisotropic Delaunay kernel.*

Again, the proof of the validity relies on the definition of a cavity, star-shaped with respect to p. Notice however that the resulting triangulation T_{i+1} may not be a Delaunay triangulation [26], but it is still conforming to finite elements requirements. In this respect, this result provides a practical way of inserting a point in an anisotropic triangulation.

Numerical issues

Obviously, taking into account only the metric tensor at the given point p is not sufficient and accurate in many applications as the mesh element size may change rapidly from one point to another in space. Therefore, we advocate considering all metric tensors related to the vertices of element K [20]. But doing so however, leads to solving a nonlinear system of equations. The center O_K of the topological sphere circumscribed to tetrahedron $K = \{p_1, \ldots, p_4\}$ is the solution of a set of equations:

$$l_{M_p}(O_K, p_1) = l_{M_p}(O_K, p_i) \qquad i = 2, \ldots, 4\,,$$

where the length of the edge $O_K p_i$ in the metric M_p is given by:

$$\begin{aligned} l_{M_p}(O_K, p_i) = {} & 2\, m_{12}\, (x_i - O_x)(y_i - O_y) + m_{11}\, (x_i - O_x)^2 \\ & + 2\, m_{13}\, (x_i - O_x)(z_i - O_z) + m_{22}\, (y_i - O_y)^2 \\ & + 2\, m_{23}\, (y_i - O_y)(z_i - O_z) + m_{33}\, (z_i - O_z)^2, \end{aligned}$$

with $p_i = (x_i, y_i, z_i)^t$, $O_K = (O_x, O_y, O_z)^t$, and $M_p = (m_{ij})_{1 \le i,j \le 3}$. Finding the center O_K simply leads to solving the following linear system:

$$\begin{pmatrix} a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \\ a_4 & b_4 & c_4 \end{pmatrix} \begin{pmatrix} O_x \\ O_y \\ O_z \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix},$$

with the coefficients:

$$\begin{aligned} a_i &= 2\,(m_{11}\,(x_i - x_1) + m_{12}\,(y_i - y_1) + m_{13}\,(z_i - z_1)), \\ b_i &= 2\,(m_{22}\,(y_i - y_1) + m_{12}\,(x_i - x_1) + m_{23}\,(z_i - z_1)), \\ c_i &= 2\,(m_{33}\,(z_i - z_1) + m_{13}\,(x_i - x_1) + m_{23}\,(y_i - y_1)), \\ d_i &= m_{11}\, x_i^2 + 2\, m_{12}\, x_i\, y_i + 2\, m_{13}\, x_i\, z_i + m_{22}\, y_i^2 + 2\, m_{23}\, y_i\, z_i + m_{33}\, z_i^2 \\ &\quad -(m_{11}\, x_1^2 + 2\, m_{12}\, x_1\, y_1 + 2\, m_{13}\, x_1\, z_1 + m_{22}\, y_1^2 + 2\, m_{23}\, y_1\, z_1 + m_{33}\, z_1^2). \end{aligned}$$

Furthermore, numerical experiments revealed that it is interesting to account for all metric tensors at the vertices p_i in order to define the cavity. We found efficient to assess the cavity with two inequalities:

$$\begin{cases} \alpha_{M_p}(p,K) < 1, \\ \sum_{i=1}^{4} \alpha_{M_{p_i}}(p,K) + \alpha_{M_p}(p,K) < 5. \end{cases} \tag{4}$$

Like in the generic case, we avoided the creation of most *slivers* by introducing a minimal volume requirement $|K| > \varepsilon$, the measure of the element being computed in the relevant metric.

4 Local Anisotropic Mesh Adaptation

Adaptive meshing methods belong to one of two categories depending on whether they proceed by global or local remeshing of the computational domain at each iteration. Global remeshing techniques consist in constructing a new mesh of the domain at each iteration, to ensure that the elements are in good agreement with the anisotropic metric-related prescriptions. The latter are supplied at the vertices of the previous mesh that is then acting as a control space. Obviously, the order of complexity of the meshing method remains the same throughout the whole adaptation scheme. In steady-state adaptative simulations, the number of modifications usually decreases with the iterations since a fixed point of the pair (mesh,solution) is targeted. In other words, once mesh features have been identified and captured by adjusting the local vertex density, numerical accuracy is only a matter of introducing a few more vertexs in critical regions while most of the mesh is kept unchanged. Hence, remeshing at each iteration the whole domain results in a loss of efficiency (even if improvements have been proposed [9, 19]). It seems more advantageous that fewer vertices are inserted in critical regions over time, as this will help to minimize the run time of the algorithm while at the same time converging to a solution. Hence, in our approach, we perform local mesh modifications.

4.1 Mesh Modification Operations

Mesh modification are either geometrical (edge split, edge collapse, vertex relocation) or topological (edge flip). In the anisotropic context, we assume that a metric tensor field is provided at the mesh vertices of a given triangulation. The objective is then to modify this triangulation iteratively by local operations in order to obtain a quasi-uniform triangulation wih respect to this field. An important feature consists in modifying the metric specifications to account for a desirable (*i.e.*, user-specified) mesh gradation. This procedure is fully described in [8] and is used here as such.

Our approach is based on the analysis of the edge lengths. Given a metric field prescription, all mesh edges have to belong to the interval $[l_{min}, l_{max}]$ and the mesh element quality has to be close to the optimal unit value. Theoretically, a quasi-uniform triangulation is characterized by the fact that all edges have a unit length, *i.e.*, $l_{min} = l_{max} = 1$. However, it is easy to understand that such restriction is highly improbable and we suggest to set the lower and upper

bounds of the edge lengths to the values $l_{min} = \sqrt{2}^{-1}$ and $l_{max} = \sqrt{2}$. All local mesh modifications can be simulated in order to check whether the resulting configuration is better than the previous one. A configuration is considered as improved if the mesh quality of all the elements involved is better (or almost identical in the case of an edge collapse) than the original configuration.

Short edges $l_M(\mathbf{e}) < l_{min}$ are simply collapsed by merging the two endpoints of such edge at one of the endpoints. All mesh elements surrounding this edge are then removed. If a short edge cannot be collapsed, an edge flip operation attempts to solve the problem. The edge swap consists in remeshing the set of elements surrounding the edge by changing the vertex connectivity. This is an expensive modification as it requires the computation of a large (combinatorial) number of quality measures. For such reason, no edge flip will be checked if the set of elements contains more than 7 elements.

Long edges $l_M(\mathbf{e}) > l_{max}$ are normally removed by splitting them into unit subsegments with respect to the metric specification. This consists in introducing the new vertices using the anisotropic Delaunay kernel. The main advantage of this approach is that the set of elements concerned by the vertex insertion (the cavity) is usually much larger than the set of elements strictly surrounding the edge (the initial cavity). Computationally, it revealed more efficient to proceed like this and the quality of the resulting mesh is better than with the classical edge split operation. Furthermore, the creation of *slivers* is explicitely checked and prevented during this stage. Sometimes, in peculiar cases, an edge flip operation can be used to remove a long edge.

In addition, a mesh optimization stage involves a vertex relocation procedure. This procedure aims at equidistributing the edge lengths, with the objective of improving the efficiency index. Given a mesh vertex p, the objective is to improve the lengths of all edges connected to p. To this end, the vertex is moved in the direction of an optimal vertex location corresponding to the average optimal positions along each edge. This operation does not affect the mesh topology, nonetheless optimal vertex positions have to be checked to prevent invalid configurations.

All mesh modification operations are described in details in the general purpose book [18].

4.2 Mesh Adaptation Scheme

Our mesh adaptation algorithm can then be described as follows: we assume given a triangulation T and a metric tensor field M defined at the mesh vertices,

```
Enforce the desired h-gradation,
do
   1. loop over mesh edges /* edge analysis */
      compute l_M(e) for every edge e
      if (l_M(e) < l_min) then
         collapse e
      else if (l_M(e) > l_max) then
```

```
            split e and insert new vertex (Delaunay kernel)
         endif
      while ( T is modified )
      2. loop over mesh elements /* quality optimization */
         perform edge flips
         perform vertex relocation
      end for
   while ( triangulation T is modified ).
```

Notice that this algorithm (composed of two levels of imbricated loops) always terminates in a finite number of iterations. Indeed, mesh modifications are only applied if they result in the mesh quality improvement. An infinite loop could potentially occurs between several configurations if the analysis is based on the edge lengths only. For instance, if the values of l_{min} and l_{max} are too close, the edge split and the edge collapse operations could possibly lead to a previous configuration. However, since we apply the mesh modification if the quality is strictly improved, this case cannot happen in our scheme. In addition, for efficiency purposes, we have restricted the overall number of iterations to a maximum value.

The complexity of the algorithm is related to the number of loops that are performed. The inner loops are in $O(nt)$, where nt is the number of mesh elements. Since we limit the nmber of outer loop to a maximum value c, the overall complexity is $c\,O(nt)$, *i.e.*, linear in number of mesh elements.

4.3 Application to Large Displacements Problems

As pointed out in the introduction, our local anisotropic mesh adaptation approach allows us to handle rigid-body displacement problems without difficulty. Fluid-structure interactions involves a moving structure, rigid or deformable, and a fluid in flow around a part or the whole structure. The domain boundary is moving, however moving only boundary vertices would quickly result in an invalid mesh. Therefore, internal vertices are also relocated using a linear elasticity analogy [5]. Actually, the whole procedure is straightforward and can be decomposed in three successive stages: given a field of displacements prescribed at the domain boundaries, *e.g.*, resulting from a fluid calculation, (i) solve a linear elasticity equation as suggested by [5] to define a discrete displacement field at all mesh vertices and (ii) move the mesh vertices to the positions prescribed by this field and (iii) optimize the resulting mesh.

This boundary value problem assumes that part of the boundary remains fixed as another part is moving. It reads:

$$\begin{cases} -div\sigma(u) = f \text{ in } \Omega \\ u = 0 \text{ on } \Gamma_{fixed} \\ \sigma(u)\cdot n = 0 \text{ on } \partial\Omega\backslash\Gamma_{fixed} \end{cases}$$

and the Hook's law gives the relation between the stress tensor σ and the linearized strain tensor e as: $\sigma = 2\mu e(u) + \lambda tr(e(u))I_3$, with $e(u) = \frac{1}{2}(\nabla u + \nabla u^t)$, λ and μ are the Lamé coefficients describing the mechanical properties of the material. The variational formulation is solved using affine P_1 Lagrange finite elements. The Lamé coefficients have been set experimentally in our simulations. The linear system is solved using a conjugate gradient routine with a LDL^t preconditioning step. The convergence of the linear system is not affected too much by the anisotropy of the mesh, indicating that the condition number does not degrade too much with the anisotropy.

Notice that our approach preserves the number of mesh vertices during this stage and involves vertex relocation and edge flips only. However, edge splits and edge colapses can also be used to achieve a better efficiency in large displacement problems.

5 Application Examples

Our approach has been tested on numerous test cases and is currently used by several research groups. We provide here a set of numerical examples to assess this method.

5.1 An Analytical Metric

This first test case is provided to illustrate our mesh adaptation scheme on a steady-state problem where the size, the shape and the orientation of the mesh elements are prescribed using the following analytical metric:

$$M = \begin{pmatrix} h_1^{-2} & 0 & 0 \\ 0 & h_{max}^{-2} & 0 \\ 0 & 0 & h_{max}^{-2} \end{pmatrix} \text{ with } \begin{cases} h_1 & = h_{max}|1 - e^{-|x-0.5|}| + 0.003 \\ h_{max} & = 0.2 \end{cases} \tag{5}$$

This metric simulates the capture of a planar shock in the computational domain: $\Omega = [-1\,,\,1]^3$. We started with an initial surface-adapted mesh T_0 containing no internal vertex containing $4,399$ vertices and $13,978$ tetrahedra (Figure 1). The final adapted mesh after 5 iterations contains $51,440$ tetrahedra and its efficiency index is 0.87. Regarding the mesh quality, 99.96 percents of the elements have a quality measure between 1 and 3.

5.2 Interface Capturing

Our approach has also proven useful in accurately tracking and approximating a dynamically evolving interface [14]. In this approach, the metric tensor field is related to the intrinsic properties of the manifold of codimension one that correspond to the interface. We consider an implicitly-defined, scalar valued function u on a domain $\Omega \subset \mathbb{R}^3$ and we denote by Γ the surface associated to the isovalue $u = 0$. The objective is to produce a mesh where the density

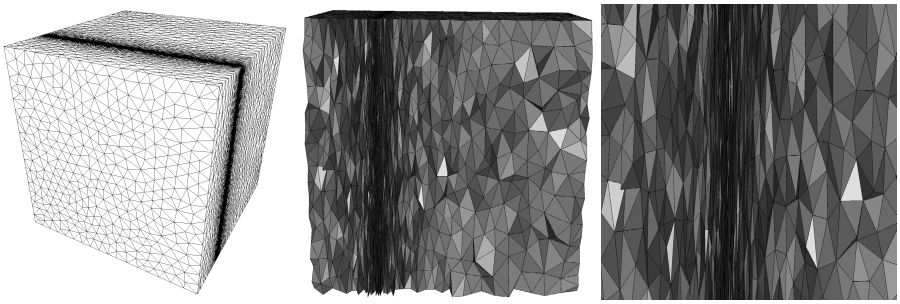

Fig. 1. Initial surface-adapted mesh (left), cut though the final adapted mesh (middle) and local enlargment (right)

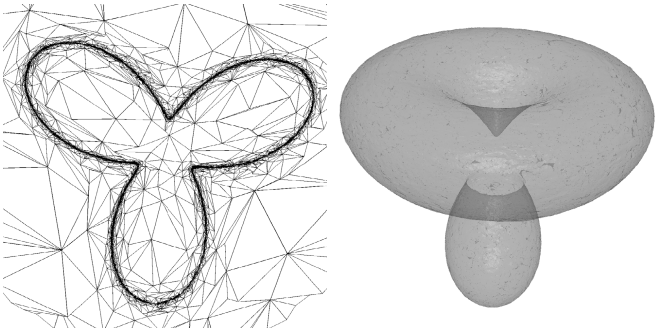

Fig. 2. Example of mesh adaptation to capture an implicitly defined surface: cutting plane through tetrahedral elements (left) and isosurface reconstruction (right)

is high in the vicinity of the isosurface so as to minimize the piecewise affine approximation of this interface. To this end, we defined the following metric tensor at the vertices of all mesh elements intersected by the manifold:

$$M = R \begin{pmatrix} 1/\varepsilon^2 & 0 & 0 \\ 0 & |\lambda_2|/\varepsilon & 0 \\ 0 & 0 & |\lambda_3|/\varepsilon \end{pmatrix} {}^tR,$$

with $R = \begin{pmatrix} \nabla u \; v1 \; v2 \end{pmatrix}$, where (v_1, v_2) is a basis of the tangent plane to the surface and λ_i are the eigenvalues of the Hessian of u. At all other vertices, we define the metric αI_3 with $\alpha \in \mathbb{R}^+$. We consider the analytical surface defined in spherical coordinates as: $r = 0.45 + 0.3\,\sin(3\phi)$, with $\theta \in [0; 2\pi]$ and $\phi \in [-\frac{\pi}{2}; \frac{\pi}{2}]$. We started from an initial uniform mesh of size $h = 0.2$ (Figure 2). The final mesh after 8 iterations contains 10^5 vertices for a minimal element size $h = 10^{-3}$. The approximation error in the L_∞ norm between the surface and its piecewise affine discretization is lesser than 10^{-4}.

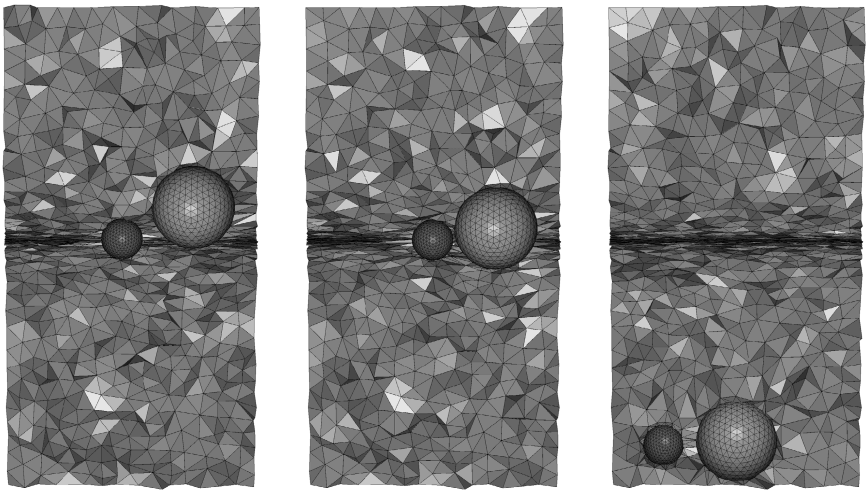

Fig. 3. Two moving spheres are crossing an anisotropic field

5.3 Moving Spheres in Anisotropic Field

We consider two spheres of different radii moving in a computational domain $\Omega = [-2\,,\,2] \times [-2\,,\,6] \times [-10\,,\,1.5]$ (Figure 3). The initial mesh is adapted to a planar shock corresponding to a metric specification very similar to that given by Equation (5). The main difficulty is to preserve the anisotropic refinement throughout the simulation. To this end, we have allowed edge split and edge collapse operations in addition to vertex relocation and edge flips. Nevertheless, the number of vertices remains almost constant: $12,200$ for he inital mesh and $11,282$ at the 34th time step. The efficiency index is also constant $\tau = 0.86$.

5.4 Airflow around a Rotating Helicopter Propeller

The flow is governed by the classical compressible Euler equations of the fluid dynamics and the numerical resolution uses the software `Fluidbox` based on finite volume and Arbitrary Lagrangian Eulerian (ALE) method [28] that imposes a constant number of vertices and the connectivity to remains identical to that of the initial mesh. The solver uses implicit time stepping scheme. Furthermore, we consider that the propeller turns with a constant angular velocity $\theta = 3.6deg$, *i.e.*, 10 full rotations per second. One mesh is generated at each time step $dt = 1/6,000$ seconds and 100 meshes are needed to achieve a complete revolution. However, the constant topology and number of vertices constraints have been relaxed: a few edge flips were introduced to remove badly-shaped elements and the solution had then to be interpolated on the optimized mesh (Figure 4). All meshes contain about 3.10^5 elements for an efficiency index close to 0.85 (more than 98 percent of the elements have a quality better than 3).

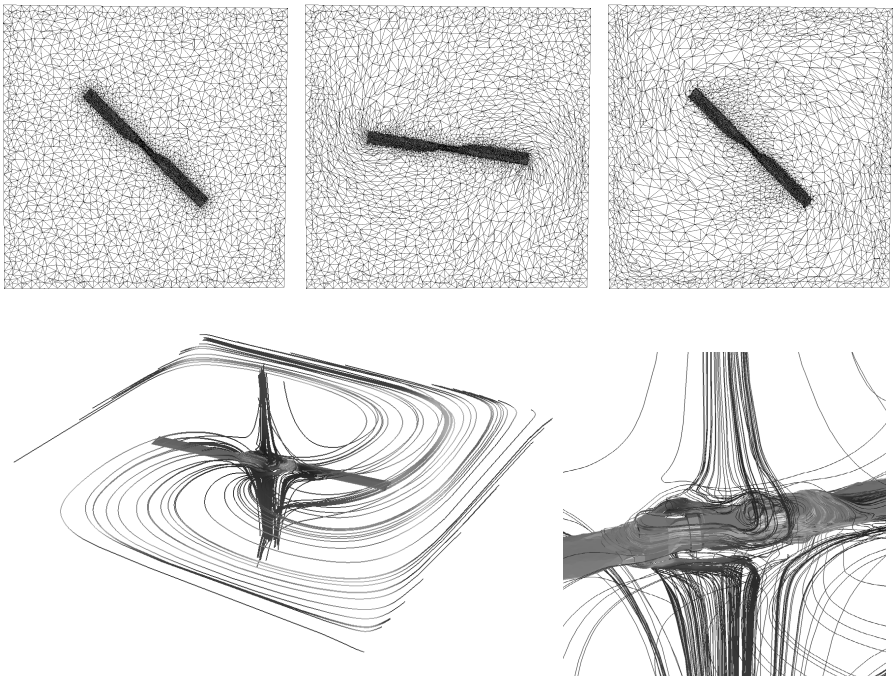

Fig. 4. Airflow simulation: adapted meshes at iterations 1, 99 and 500 (top); streamlines and velocity modulus (bottom)

6 Conclusions and Perspectives

In this paper, we have presented an efficient method for obtaining anisotropic adapted meshes based on Riemannian metric specifications. This approach is based on a modification of the classical Delaunay kernel and involve local mesh modification operations. The results obtained so far in the numerical simulations are promising and confirm the cogency of the local modification strategy. The next stage will be to handle dynamically evolving domains where both the geometry and the topology of the domains change in time.

References

1. Alauzet, F., Loseille, A., Dervieux, A., Frey, P.: Multi-dimensional continuous metric for mesh adaptation. In: Proc. 15th Int. Meshing Roundtable, pp. 191–214 (2006)
2. Apel, T., Berzins, M., Jimack, P.K., Kunert, G., Plaks, A., Tsukerman, I., Walkley, M.: Mesh shape and anisotropic elements: Theory and practice. In: Whiteman, J.R. (ed.) The Mathematics of Finite Elements and Applications X, pp. 367–376. Elsevier, Oxford (2000)

3. DAzevedo, E.F., Simpson, R.B.: On optimal triangular meshes for minimizing the gradient error. Numer. Math. 59, 321–348 (1991)
4. Babuška, I., Aziz, A.K.: On the angle condition in the finite element method. SIAM J. Numer. Anal. 13, 214–226 (1976)
5. Baker, T.J.: Mesh Movement and Metamorphosis. Eng. Comput. 18(3), 188–198 (2002)
6. Berzins, M.: A solution-based triangular and tetrahedral mesh quality indicator. SIAM J. Sci. Comput. 19, 2051–2060 (1998)
7. Bossen, F.J., Heckbert, P.S.: A pliant method for anisotropic mesh generation. In: Proc. 5th Int. Meshing Roundtable, pp. 63–74 (1996)
8. Borouchaki, H., Hecht, F., Frey, P.: Mesh gradation control. Int. J. Numer. Methods Engng. 43(6), 1143–1165 (1998)
9. Castro-Diaz, M.J., Hecht, F., Mohammadi, B., Pironneau, O.: Anisotropic unstructured mesh adaption for flow simulations. Int. J. Numer. Meth. Fluids 25, 475–491 (1997)
10. Ciarlet, P.G.: The Finite Element Method for Elliptic Problems. North-Holland, Amsterdam (1978)
11. Chernikov, A., Crisochoides, N.: Three-dimensional semi-generalized point placement method for Delaunay mesh refinement. In: Proc. 16th Int. Meshing Roundtable, Seattle, pp. 25–44 (2007)
12. Dolejsi, V.: Anisotropic mesh adaptation for finite volume and finite element methods on triangular meshes. Computing and Visualisation in Science 1, 165–178 (1998)
13. Dompierre, J., Vallet, M.-G., Bourgault, Y., Fortin, M., Habashi, W.G.: Anisotropic mesh adaptation: Towards user-indepedent, mesh-independent and solver-independent CFD. Part III: Unstructured meshes. Int. J. Numer. Meth. Fluids 39, 675–702 (2002)
14. Ducrot, V., Frey, P.: Contrôle de l'approximation géométrique d'une interface par une métrique anisotrope. C.R. Acad. Sci. 345, Série I, 537–542 (2007)
15. Formaggia, L., Perotto, S.: New anisotropic a priori error estimates. Numer. Math. 89, 641–667 (2001)
16. Fortin, M.: Etude numérique d'estimations d'erreur a posteriori. REEF 9, 467–486 (2000)
17. Frey, P.: About surface remeshing. In: Proc. 9th Int. Meshing Roundtable, pp. 123–136 (2000)
18. Frey, P.J., George, P.L.: Mesh generation. Application to finite elements, 2nd edn. Wiley, Chichester (2008)
19. Frey, P.J., Alauzet, F.: Anisotropic mesh adaptation for CFD simulations. Comput. Methods Appl. Mech. Engrg. 194(48-49), 5068–5082 (2005)
20. George, P.L.: Premières expériences de maillage automatique par une méthode de Delaunay anisotrope en trois dimensions, RT-0272, Inria (2002)
21. George, P.L., Borouchaki, H.: Delaunay triangulation and meshing. Applications to finite elements, Hermès Science, Paris (1998)
22. Gruau, C., Coupez, T.: 3d unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric. Comput. Methods Appl. Mech. Engrg. 194(48-49), 4951–4976 (2005)
23. Huang, W.: Metric tensors for anisotropic mesh generation. J. Comput. Phys. 204, 633–665 (2005)
24. Jiao, X., Colombi, A., Ni, X., Hart, J.: Anisotropic mesh adaptation for evolving triangulated surfaces. In: Proc. 15th Int. Meshing Roundtable, pp. 173–190 (2006)

25. Kunert, G.: A local problem error estimator for anisotropic tetrahedral finite element meshes. SIAM J. Numer. Anal. 39, 668–689 (2001)
26. Labelle, F., Shewchuk, J.: Anisotropic Voronoi Diagrams and Guaranteed-Quality Anisotropic Mesh Generation. In: Proc. 19th Annual Symposium on Computational Geometry, pp. 191–200. ACM, San Diego (2003)
27. Löhner, R.: Regridding surface triangulations. J. Comput. Physics 26(1), 1–10 (1996)
28. Nkonga, B.: On the conservative and accurate CFD approximations for moving meshes and moving boundaries. Comput. Methods Appl. Mech. Engrg. 190, 1801–1825 (2000)
29. Osher, S., Fedkiw, R.: Level Set Methods and Dynamic Implicit Surfaces, vol. 153. Springer, New York (2002)
30. Peraire, J., Vahdati, M., Morgan, K., Zienkiewicz, O.C.: Adaptive remeshing for compressible flow computations. J. Comput. Phys. 72, 449–466 (1997)
31. Picasso, M.: An anisotropic error indicator based on Zienkiewicz-Zhu error estimator: Application to elliptic and parabolic problems. SIAM J. Sci. Comput. 24, 1328–1355 (2003)
32. Remacle, J., Li, X., Shephard, M.S., Flaherty, J.E.: Anisotropic adaptive simulation of transient flows using discontinuous Galerkin methods. Int. J. Numer. Meth. Engr. 62(7), 899–923 (2003)
33. Rippa, S.: Long and thin triangles can be good for linear interpolation. SIAM J. Numer. Anal. 29, 257–270 (1992)
34. Shewchuk, J.R.: What Is a Good Linear Element? Interpolation, Conditioning, and Quality Measures. In: 11th Int. Meshing Roundtable, pp. 115–126 (2002)
35. Zlámal, M.: On the finite element method. Numer. Math. 12, 394–409 (1968)