Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 00 (2014) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

23rd International Meshing Roundtable (IMR23)

Metric-orthogonal anisotropic mesh generation

Adrien Loseille[a]

[a]*INRIA Paris-Rocquencourt, 78153 Le Chesnay, France*

Abstract

We introduce a procedure to generate metric-orthogonal anisotropic volume meshes. These meshes are obtained when most of the edges are aligned with the eigenvectors of a provided input metric field. The goal is to generate locally structured meshes in an anisotropic context and in the presence of complex geometries. The algorithm relies on two local mesh modification operators. The first one is used to quickly coarsen volume meshes. From a coarse volume mesh, the second operator is used to insert iteratively points in a frontal manner. The local alignment and orthogonality are naturally inherited from the eigenvectors and eigenvalues of an input metric field. We show on several 3D numerical examples that this procedure is robust and can generate anisotropic locally structured meshes.

Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

Keywords: Anisotropic Mesh Adaptation, Cavity-based Operators, Orthogonality, Quasi-structured Meshes, Metric Alignment, Metric-Orthogonal Meshes, Cartesian Core, Generalized Coarsening Operator

1. Introduction

Cartesian or structured grids are usually preferred in the numerical simulation community as they have many interesting properties: natural high-order solutions, fast numerical solvers and high robustness. However, when complex geometries are used, robust Cartesian grids are less straightforward to implement, many research exist on the subject in order to intersect a complex geometry with a Cartesian grid [1]. As far as mesh adaptation is concerned, reaching a high level of anisotropy remains a challenge as the directions are always aligned with the natural axes.

On a different perspective, unstructured mesh generation is now a mature field of research where many methods are able to generate a 3D volume mesh with respect to a prescribed surface mesh. The most common methods are based on the Delaunay-kernel with boundary recovery [4,11,14], on a frontal approach [23,29] or on a combination of both with local reconnections [28]. All these techniques have been extended in the framework of anisotropic unstructured mesh adaptation. For robustness issues, most of the strategies rely on local mesh modification operators [5,7–9,12,15, 18,20,22,31,33] . If these strategies produce highly anisotropic tetrahedra, they fail to naturally produce elements that are aligned with the eigenvectors of the provided metric field. Consequently, the quest of anisotropic mesh generation

* Corresponding author. Tel.: +33 1 39 63 55 39 ; fax: +33 1 39 63 58 82.
E-mail address: Adiren.Loseille@inria.fr

Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

with locally structured elements remains an open problem with only a small number of previous attempts in 2D and 3D [19,32].

In this paper, we show that metric-orthogonal meshes, *i.e.* composed of elements aligned with the eigenvectors of the metric, are just a different set of anisotropic meshes. In order to generate these meshes, we define a strategy that relies on a cavity-based operator and a frontal insertion of points.

The paper is organized as follows. In Section 2, we recall the basics of metric-based mesh adaptation and give the basic principles of our approach. In Section 3, we define the cavity-based operator. It is used to insert points and also to define a generalized coarsening operator that removes almost all volume points of a given mesh. This mesh with almost no volume points is the starting step before the insertion of points in a frontal manner. In Section 4, we review how the points are created and filtered. In Section 5, we illustrate this procedure on 3D numerical examples.

2. Metric-based mesh adaptation and metric-orthogonality

Metric-based mesh adaptation was introduced in the pioneering works [6,16]. It allows to transform any unstructured uniform mesh generator into an anisotropic one. This is done by computing the distance in a Riemannian space instead of the classical Euclidean metric space.

A metric tensor field of Ω is a Riemannian metric space denoted by $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$, where $\mathcal{M}(\mathbf{x})$ is a 3×3 symmetric positive definite matrix. Taking this field at each vertex $\mathbf{x}_i$ of a mesh $\mathcal{H}$ of Ω defines the discrete field $\mathcal{M}_i = \mathcal{M}(\mathbf{x}_i)$. If N denotes the number of vertices of $\mathcal{H}$, the linear discrete metric field is denoted by $(\mathcal{M}_i)_{i=1\dots N}$. As $\mathcal{M}(\mathbf{x})$ and $\mathcal{M}_i$ are symmetric definite positive, they can be diagonalized in an orthonormal frame, such that

$$\mathcal{M}(\mathbf{x}) = {}^t\mathcal{R}(\mathbf{x})\Lambda(\mathbf{x})\mathcal{R}(\mathbf{x}) \text{ and } \mathcal{M}_i = {}^t\mathcal{R}_i\Lambda_i\mathcal{R}_i,$$

where $\Lambda(\mathbf{x})$ and Λ_i are diagonal matrices composed of strictly positive eigenvalues $\lambda(\mathbf{x})$ and λ_i, $\mathcal{R}$ and $\mathcal{R}_i$ orthonormal matrices verifying ${}^t\mathcal{R}_i = (\mathcal{R}_i)^{-1}$. Setting $h_i = \lambda_i^{-1}$ allows to define the sizes prescribed by $\mathcal{M}_i$ along the principal directions given by $\mathcal{R}_i$. Note that the set of points verifying the implicit equation ${}^t\mathbf{x}\,\mathcal{M}_i\,\mathbf{x} = 1$ defines a unique ellipsoid. This ellipsoid is called the unit-ball of $\mathcal{M}_i$ and is used to represent geometrically $\mathcal{M}_i$ as in Figure 1.

The two fundamental operations in a mesh generator are the computation of length and volume. The length of an edge $\mathbf{e} = [\mathbf{x}_i, \mathbf{x}_j]$ and the volume of an element K are evaluated continuously in $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ by:

$$\ell_{\mathcal{M}}(\mathbf{e}) = \int_0^1 \sqrt{{}^t\mathbf{e}\,\mathcal{M}(\mathbf{x}_i + t\,\mathbf{e})\,\mathbf{e}}\,\mathrm{d}t \text{ and } |K|_{\mathcal{M}} = \int_K \sqrt{\det(\mathcal{M}(\mathbf{x}))}\,\mathrm{d}\mathbf{x} \tag{1}$$

From a discrete point view, the metric field needs to be interpolated [10] to compute approximate length and volume. For the volume, we consider a linear interpolation of $(\mathcal{M}_i)_{1\dots N}$ and the following edge length approximation is used:

$$|K|_{\mathcal{M}} \approx \sqrt{\det\left(\frac{1}{4}\sum_{i=1}^{4}\mathcal{M}_i\right)}|K| \text{ and } \ell_{\mathcal{M}}(\mathbf{e}) \approx \sqrt{{}^t\mathbf{e}\,\mathcal{M}_i\,\mathbf{e}}\,\frac{r-1}{r\ln(r)}, \tag{2}$$

where $|K|$ is the Euclidean volume of K and r stands for the ratio $\sqrt{{}^t\mathbf{e}\,\mathcal{M}_i\,\mathbf{e}}/\sqrt{{}^t\mathbf{e}\,\mathcal{M}_j\,\mathbf{e}}$. The approximate length arises from considering a geometric approximation of the size variation along end-points of $\mathbf{e}$: $\forall t\in[0,1]\; h(t) = h_i^{1-t}\,h_j^t$.

The task of the adaptive mesh generator is then to generate a unit-mesh with respect to $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\boldsymbol{\Omega}}$. A mesh is said unit when it is only composed of unit-volume elements and unit-length edges. Practically, these two requirements are combined in a quality function computed in the metric field:

$$Q_{\mathcal{M}}(K) = \frac{3^{\frac{1}{3}}}{36}\frac{\sum_{i=1}^{6}\ell^2_{\mathcal{M}}(\mathbf{e}_i)}{|K|^{\frac{2}{3}}_{\mathcal{M}}} \in [1,\infty]. \tag{3}$$

Many softwares based on the unit mesh concept exist: `Bamg` [15] and `BL2D` [20] in 2D, `Yams` [9] for discrete surface mesh adaptation and `Feflo.a` [26], `Forge3d` [7], `Fun3d` [18], `Gamanic3d` [13], `MeshAdap` [22], `Mmg3d` [8], `Moess` [30], `Mom3d` [33], `Tango` [5] and [31] in 3D.

During an adaptive remeshing process based on (1) and (3), the shape of the created unit elements is not controlled. Indeed, as a unit element is given by $\mathcal{M}^{-\frac{1}{2}}\,R\,K$ [24], where K is the regular triangle or tetrahedron and R a rotation

matrix, the edges of a unit element can span all the directions of space, see Figure 1. In addition, the eigenvectors of $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ are never explicitly used in the distance, volume or quality function. As a result, all the aforementioned anisotropic mesh generators produce anisotropic elements where the edges have no particular orientation. If an equal level of interpolation error (in $\mathbb{L}^1$ norm) can be achieved on these unit elements [24], their dihedral angles or error level in $\mathbb{L}^2$ or $\mathbb{H}^1$ norms are greatly different. These quantities are not optimized in classical anisotropic mesh adaptation although they may unfavorably impact the quality of a numerical simulation. In this paper, we show how to enforce the alignment of the edges with the local eigenvectors of the provided metric. As the eigenvectors are orthogonal, such meshes are called *metric-orthogonal*, *$\mathcal{M}$-orthogonal* or *quasi-structured* meshes [32]. In particular, when an isotropic metric field is provided, *cartesian* grids are recovered with elements aligned with the *x*-*y*-*z* directions.

As we want to force the alignment of the edges, standard local remeshing approaches based on a set of classical operators (insertion, collapse, swap, ...) as in [25,30] seem to be more delicate to use as they iteratively modify the mesh with no specific ordering. On the contrary, frontal methods have been used to generate high-quality isotropic meshes but with little success for anisotropic mesh generation. In this paper, we combine both approaches: only local operators are used in order to ensure robustness and a frontal insertion of points is used in order to control the alignment of vertices along the eigenvectors of $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$. However, contrary to fully frontal mesh generation techniques [23,29] where a front of points/elements is used to fill the computational domain, the points are inserted in an *empty volume mesh*. In this paper, an *empty mesh* is a valid volume mesh composed of a minimum (or a small) number of volume points, while the surface mesh is assumed to be adapted to the input metric. Inserting the points in an empty volume mesh is motivated to avoid the collision of the frontal points with already existing volume points. Note that empty meshes are usually generated after the boundary recovery phase in typical mesh generation algorithm [4,11]. However, instead of starting the process from the empty mesh generated by the mesh generation process, we propose a fast coarsening operator to reach this state quickly. This allows us to store the initial volume mesh and metric and use them as a background information to interpolate the metric for the frontal creation of vertices. The interpolation is based on the logarithmic map [3]. Given a sequence of points $(\mathbf{x}_i)_{i=1...k}$ and their respective metrics $\mathcal{M}_i$, then the interpolated metric in $\mathbf{x}$ verifying :

$$\mathbf{x} = \sum_{i=1}^{k} \alpha_i \, \mathbf{x}_i, \text{ with } \sum_{i=1}^{k} \alpha_i = 1,$$

is

$$\mathcal{M}(\mathbf{x}) = \exp\left(\sum_{i=1}^{k} \alpha_i \ln(\mathcal{M}_i)\right). \tag{4}$$

In the space of metric tensors, logarithm and exponential operators apply on metric eigenvalues directly:

$$\ln(\mathcal{M}_i) = {}^t\mathcal{R}_i \, \ln(\Lambda_i)\, \mathcal{R}_i \quad \text{and} \quad \exp(\mathcal{M}_i) = {}^t\mathcal{R}_i \, \exp(\Lambda_i)\, \mathcal{R}_i.$$

Consequently, in order to reduce the computation of the interpolation, the metric on the background mesh is stored in logarithmic form. This avoids multiple diagonalisation steps. Starting from the initial set of points of the surface mesh, new points are created along eigenvectors directions at unit-length and then inserted in the current mesh. Both coarsening and insertion operator are based on a cavity-based operator described in Section 3.

3. Anisotropic cavity-based operator

We describe in this section a cavity-based operator in an anisotropic context. It is used to insert points of a given mesh when a metric field is provided. Then, we extend this operator to define a generalized fast coarsening operator to generate an *empty volume mesh*.

3.1. Insertion operator

The cavity-based operator is inherited from incremental methods where the current mesh $\mathcal{H}_k$ is modified iteratively through sequences of insertion of a point P:

$$\mathcal{H}_{k+1} = \mathcal{H}_k - \mathcal{C}_P + \mathcal{B}_P, \tag{5}$$

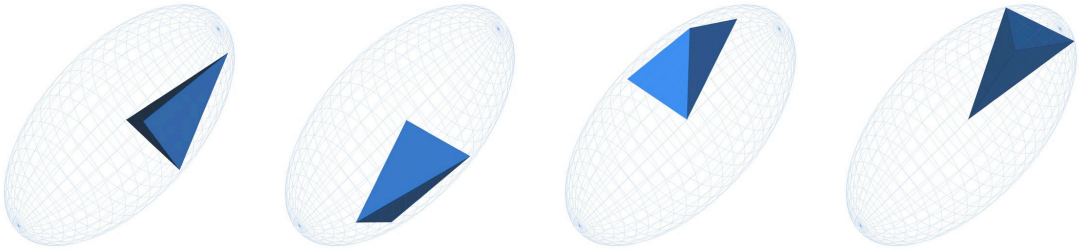

Fig. 1. Illustration of several unit tetrahedron with respect to a 3D metric represented by its unit-ball.

where C_P is the cavity. It is a set of volume and surface elements that will depend on the desired mesh modification operation. For the Delaunay insertion, the cavity is the set of elements of $\mathcal{H}_k$ such that P is contained in their circumcenter. $\mathcal{B}_P$ is the ball of P, i.e., the set of new elements having P as vertex. These elements are created by connecting P to the set of the boundary faces of C_P. This insertion pattern is illustrated in Figure 2 in 2D. Note that if $\mathcal{H}_k$ is a valid mesh (only composed of elements of positive volume) then $\mathcal{H}_{k+1}$ will be valid if and only if C_P is connected (through internal faces of tetrahedron) and $\mathcal{B}_P$ generates only valid elements. The standard Delaunay kernel has been extended to anisotropic mesh adaptation in [8,21]. This consists in generating a mesh having edges of almost unit-length computed in the metric, see Section 1. In that case, the Delaunay cavity criterion is modified to comply with the size of the metric field. It is composed of any element K verifying:

$$\alpha_{\mathcal{M}}(P,K) = \frac{\|OP\|_{\mathcal{M}}}{(r_K)_{\mathcal{M}}} < 1, \tag{6}$$

where P is the point being inserted, O the center of the circumcenter of K computed in $\mathcal{M}$ and r_K the radius computed in $\mathcal{M}$. When $\mathcal{M}$ is varying, Equation (6) is not straightforward to compute, so we use the modified Delaunay criterion defined in [8] instead. It relies only on point-wise metric evaluation:

$$\begin{cases} \alpha_{\mathcal{M}(P)}(P,K) < 1, \\ \displaystyle\sum_{i=1}^{4} \alpha_{\mathcal{M}(P_i)}(P,K) + \alpha_{\mathcal{M}(P)}(P,K) < 5,\ P_i \in K. \end{cases}$$

The previous operator prevents the suppression of edges/faces having an admissible size in $\mathcal{M}$ while having bad angle for the standard Delaunay. Consequently, the creation of slivers is not handled by this operator. We add the following additional control to avoid their creation. The idea consists in controlling the height of the tetrahedron in addition to the volume. Once the anisotropic cavity is computed, the external faces of the cavity leading to a valid tetrahedron (positive volume) are checked to verify that their heights are greater than the minimal possible height given by the metric $\mathcal{M}(P)$:

$$h_{tar} = \max(\lambda_1,\lambda_2,\lambda_3)^{-2}\,\frac{\sqrt{3}}{6}, \tag{7}$$

where $(\lambda_i)_{i=1,3}$ are the eigenvalues of $\mathcal{M}(P)$ and $\frac{\sqrt{3}}{6}$ the height of the regular tetrahedron. The cavity is then reduced to remove negative volume elements and elements that do not verify (7). The previous formula simply states that the worst height of a unit tetrahedron is found when the height vector is aligned with the eigenvector of minimal size. This modification reduces the number of slivers and also reduces the amount of optimization (swaps).

3.2. Generating an empty volume mesh

Our strategy relies on inserting the points in a specific frontal manner in order to force the quasi-structured aspect of the mesh. Consequently, we start the frontal insertion procedure in an almost empty volume mesh. In order to be as robust as possible, we prefer to start from a given valid 3D mesh and remove (almost) all volume points. We define

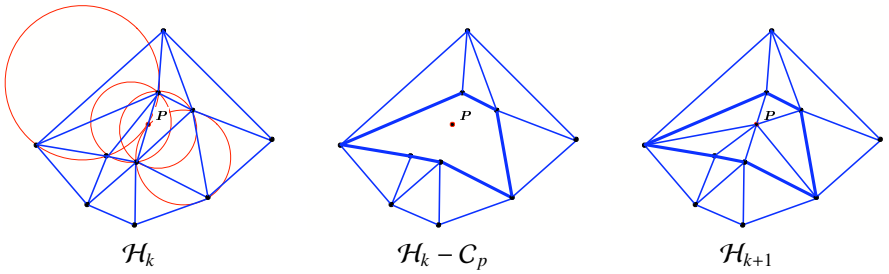

Fig. 2. Illustration of the incremental point insertion (5) in the case of the 2D Delaunay point insertion.

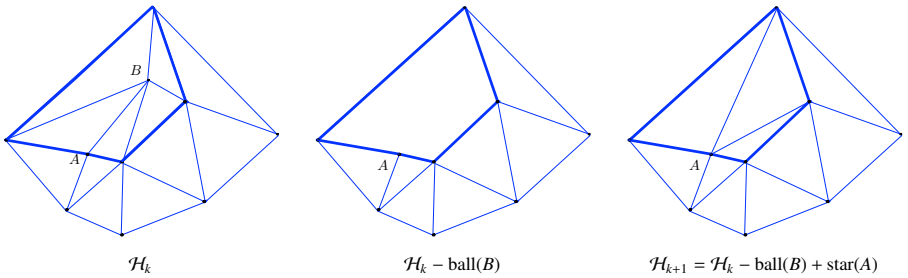

Fig. 3. Standard collapse recasting as a cavity-based insertion: To remove B, A is reinserted.

Test case	Generalized coarsening		Standard collapse		Ghs3d
landing gear	169 162 points removed	3.4s	151 313 points removed	6.8s	30.8s
scud geometry	738 623 points removed	6.8s	708 150 points removed	18s	9.3s
aircraft with boundary layer	2 198 702 points removed	16s	2 156 122 points removed	54s	600s
seeb model	9 749 932 points removed	63s	9 645 854 points removed	261s	*failure*

Table 1. CPU times to generate an *empty volume mesh* using the new generalized fast collapse operator, the standard collapse and ghs3d-tetmesh industrial mesh generator on a sequence of different 3D meshes.

in this section a generalized coarsening operator that allows us to quickly reach this state. This new collapse is based on the previous insertion and outperforms the standard collapse operator or the boundary recovery techniques of mesh generators.

In order to work in the cavity framework, we need to recast the standard collapse within a cavity-based operation. To do so, to collapse an edge $[A, B]$, the point A (resp. B) needs to be re-inserted with the list of elements containing B (resp. A) as initial cavity. Note that the standard collapse is often rejected because some faces in the ball of B lead to negative volume when A is re-inserted. In order to define a generalized coarsening operator, we modify the initial cavity by adding neighboring tetrahedra in order to make the insertion possible. Consequently, to remove B, A is inserted with (5), C_A initialized with the ball of B. The cavity is then modified according to Algorithm 1 in order to favor the reinsertion, see Figure 3. Table 1 reports the CPU time to generate an empty volume mesh with this standard collapse, the new fast coarsening operator and the boundary recovery techniques of Ghs3d-Tetmesh [11]. The selected geometries are depicted in Figure 4 and are composed of meshes of different sizes and features. The generalized coarsening operator with the cavity enlargement is 2 to 4 times faster than the traditional collapse. In addition, more points are removed in the volume mesh with the fast coarsening operator. These tests were performed on an Intel Core i7 at 2.7Ghz with 16Gb of RAM.

The fast generalized collapse procedure is always applied first on the input mesh. From the obtained *empty volume mesh*, points are created, filtered and inserted with the previous cavity-based inserter.

Algorithm 1 Cavity enlargement for (re)insertion of P

Volume Part:
For each K in C_P

For each face $[A, B, C]$ of such that $P \notin [A, B, C]$:
if $volume(A, B, C, P) < 0$, then
if P is a surface point then **reject**
else add neighboring tetrahedron to C_P
endif
endif
EndFor

EndFor

if C_P is modified goto **Volume Part**.

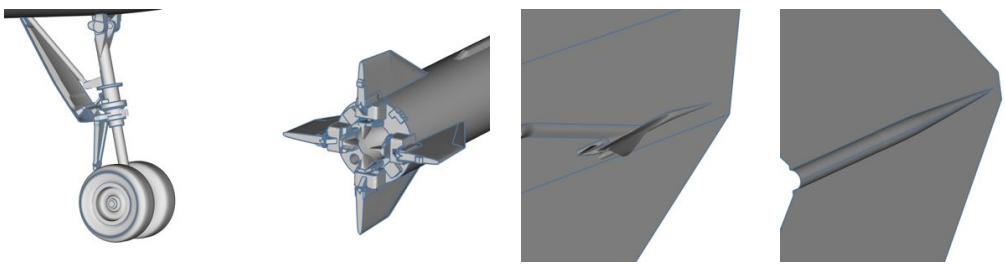

Fig. 4. Illustration of the geometries used to test the generalized coarsening operator: from left to right, landing gear geometry, scud geometry, supersonic aircraft and a seeb model.

4. Metric-orthogonal anisotropic mesh generation

Starting from a provided input 3D valid volume mesh and metric, the generation of an adapted metric-orthogonal mesh is a combination of the previous operators with a frontal algorithm to propose the points to be inserted. The complete procedure is composed of the following steps:

- Store the initial mesh-metric couple as a background information,
- Adapt the surface mesh in the standard way [25],
- Generate an *empty volume mesh* with the fast collapse operator,
- Create, filter and insert points with the cavity-based inserter.

In this section, we describe how the new points are created and filtered before being inserted.

4.1. Frontal creation of vertices

In order to favor orthogonality and metric alignment of the final mesh, a frontal approach is used. However, contrary to standard frontal approaches, we use a front of vertices instead of a front of faces. From a practical point of view, the new points are proposed by vertex and not by face. In an anisotropic context, the new points depend only on the eigenvectors and eigenvalues of the metric of the front point. The initial front of points is given by the list of the surface points. Given a point $\mathbf{x}_o$ and its metric $\mathcal{M}_o$ of the current front with eigenvectors $(\mathbf{u}_i)_{i=1,3}$ and eigenvalues $(\lambda_i)_{i=1,3}$, six points are proposed:

$$\mathbf{x}_i = \mathbf{x_o} \pm \lambda_i^{-\frac{1}{2}} \mathbf{u}_i. \tag{8}$$

When the metric is isotropic, we force the eigenvectors to be aligned with the natural axis of $\mathbb{R}^3$. Note that these points are just a first guess and several additional checks are performed before trying insertion. The first check consists in verifying that the new points are in the current volume mesh by using a simple mesh localization algorithm. This check is also performed on the background mesh. The back mesh localization also provides the metric $\mathcal{M}_i$ of $\mathbf{x}_i$.

In order to take into account the variation of the metric, the final position of $\mathbf{x}_i$ and metric $\mathbf{x}_i$ is updated. The procedure is based on a dichotomy along the segment $[\mathbf{x}_o, \mathbf{x}_i]$ in order to make sure that the Riemannian length evaluation of the vector $[\mathbf{x}_o\mathbf{x}_i]$ is unit:

$$\int_0^1 \sqrt{{}^t[\mathbf{x}_o\mathbf{x}_i]\,\mathcal{M}(t)\,[\mathbf{x}_o\mathbf{x}_i]}\,\mathrm{d}t = 1,$$

where $\mathcal{M}(t)$ is a geometric interpolation between metrics $\mathcal{M}(0) = \mathcal{M}_o$ and $\mathcal{M}(1) = \mathcal{M}_i$. Note that the original guess (8) only guarantees:

$${}^t[\mathbf{x}_o\mathbf{x}_i]\,\mathcal{M}_o\,[\mathbf{x}_o\mathbf{x}_i] = 1.$$

Consequently, we seek for an optimal point $\mathbf{x}_{opt}$ with back-mesh interpolated metric $\mathcal{M}_{opt}$ lying along the initial direction $\mathbf{x}_o\mathbf{x}_i$. Note that we need to iterate, because we interpolate the metric from the background mesh. If $\mathcal{M}_{opt}$ were interpolated by $\mathcal{M}_o$ and $\mathcal{M}_i$, an analytical formula exists depending on the metric interpolation scheme used. This list of points is then filtered in order to suppress from insertion points that are too close in the distance computed in the metric. The filtering process gives the list of points to be inserted. This list of points defines the next front. This algorithm is applied until the list of points to be inserted becomes empty. We mention that different procedures may be used to generate the list of points to be inserted. In [27], the list is issued from a front of faces instead of a front of points.

4.2. Anisotropic filtering

By using the previous point creation procedure, neighboring points in the front can generate similar points, so it is important to filter out the points that are too close (in the metric). To do so, we use an octree of points. Each octant can contain up to 10 points before being subdivided. Initially, the octree contains only the surface points (that are constrained and define the initial front). The rejection test is based on the discrete distance computation in a Riemannian metric space (1). Note that the full (costly) evaluation of (1) is not always necessary. If ${}^t[AB]\,\mathcal{M}(A)\,[AB]$ and ${}^t[AB]\,\mathcal{M}(B)\,[AB]$ are greater than 2 then $\ell_{\mathcal{M}}(A, B) > 1$. This simple observation avoids a lot of computation. To validate the insertion of a point, we first check the distances between every points that are in the octant containing the point to be inserted. If no rejection occurs, then the current octree is intersected with the bounding box of the metric. All the intersected octants are checked starting from the octants closer to the point being inserted. Then, each point that is accepted for insertion is inserted in the octree along with its metric.

4.3. Overview and comments

In this section, we illustrate the global overview of the procedure and we perform a complexity study of the operator.

An overview of the complete procedure is depicted in Figure 5. In this example, a metric-orthogonal mesh is generated around a city with a blast wave propagating in the city. In Figure 5, we depict some on the non-empty octants of the initial and final octree of points. The initial octree only contains the surface points while anisotropic features only appear in the final octree composed of all the points to be inserted. The empty volume mesh is depicted in Figure 5 (bottom left), and the final volume mesh is depicted in Figure 5 (bottom middle). A closer view to the volume mesh near a shock wave is represented in Figure 5 (bottom right). It shows how the elements are locally structured.

We then give the timing and a simple complexity study of the operator. We generate a sequence of uniform meshes with respect to uniform metrics of decreasing sizes from 0.02 to 0.0025 for a unit cube. Table 2 reports the CPU times. These meshes were generated on serial on a DELL PowerEdge R900 with 4 Intel Xeon E7 with 10-cores at 2Ghz and 1Tb of RAM. According to Table 2, this procedure has a linear complexity up to 35 million points.

size	cpu time	# of points inserted	# number of elements created
0.02	19.2s	211 935	1 155 874
0.01	4m38s	1 540 099	9 139 202
0.005	28m43s	8 238 559	51 976 676
0.0025	4h18m	38 314 040	219 903 811

Table 2. CPU times to generate uniform meshes of decreasing sizes on a unit cube.

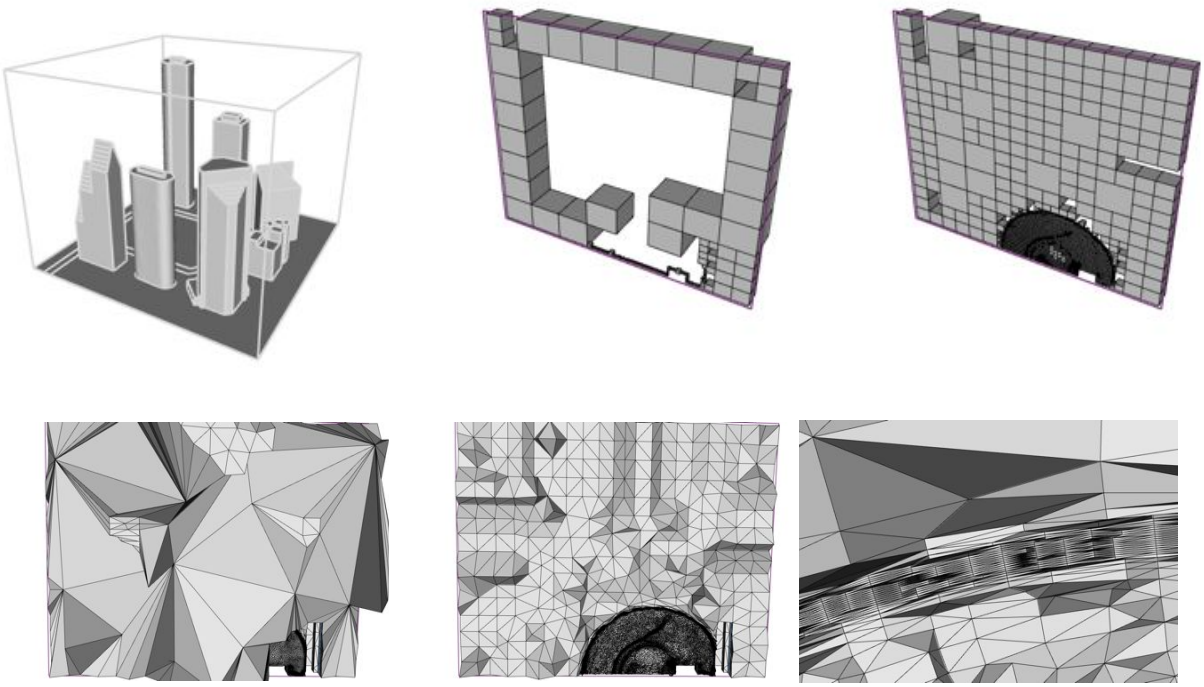

Fig. 5. Illustration of the procedure: from left to right, top to bottom, the geometry of generic city, the initial octree containing only the surface points, the final octree on the final set of points, a cut in the volume empty mesh, a cut in the final metric-orthogonal mesh and a closer view near a shock wave showing the locally structured aspect of the mesh.

5. Numerical examples

The first example is a tailored mesh where isotropic source terms are used to generate an axes-aligned mesh around and in the wake of an f117 aircraft. The aircraft has an angle of attack of 20 degrees. The scope is to study the vortical flow generated by the delta shaped wing. For this example, the use of locally structured grids is preferred as the smooth vortices interact behind the aircraft. Reducing the solver diffusion is thus a key condition to observe the full picture of the physic. The flow solver is WOLF [2], and an unsteady inviscid simulation is performed. The total CPU time is 12 hours on 8 cores of an i7 at 2.7Ghz. The mesh composed of 7 532 632 vertices and 45 721 814 tetrahedra. The CPU time to generate this mesh is 25 minutes. In Figure 6 (middle left), we see how the fronts merge smoothly with nothing but filtering as specific treatments. For the standard approach, the dihedral angles follow a Gaussian distribution centered on the mean dihedral angle of the regular tetrahedron whereas the distribution is centered around this mean value and the right angle value for the metric-orthogonal approach, see Figure 6 (bottom left).

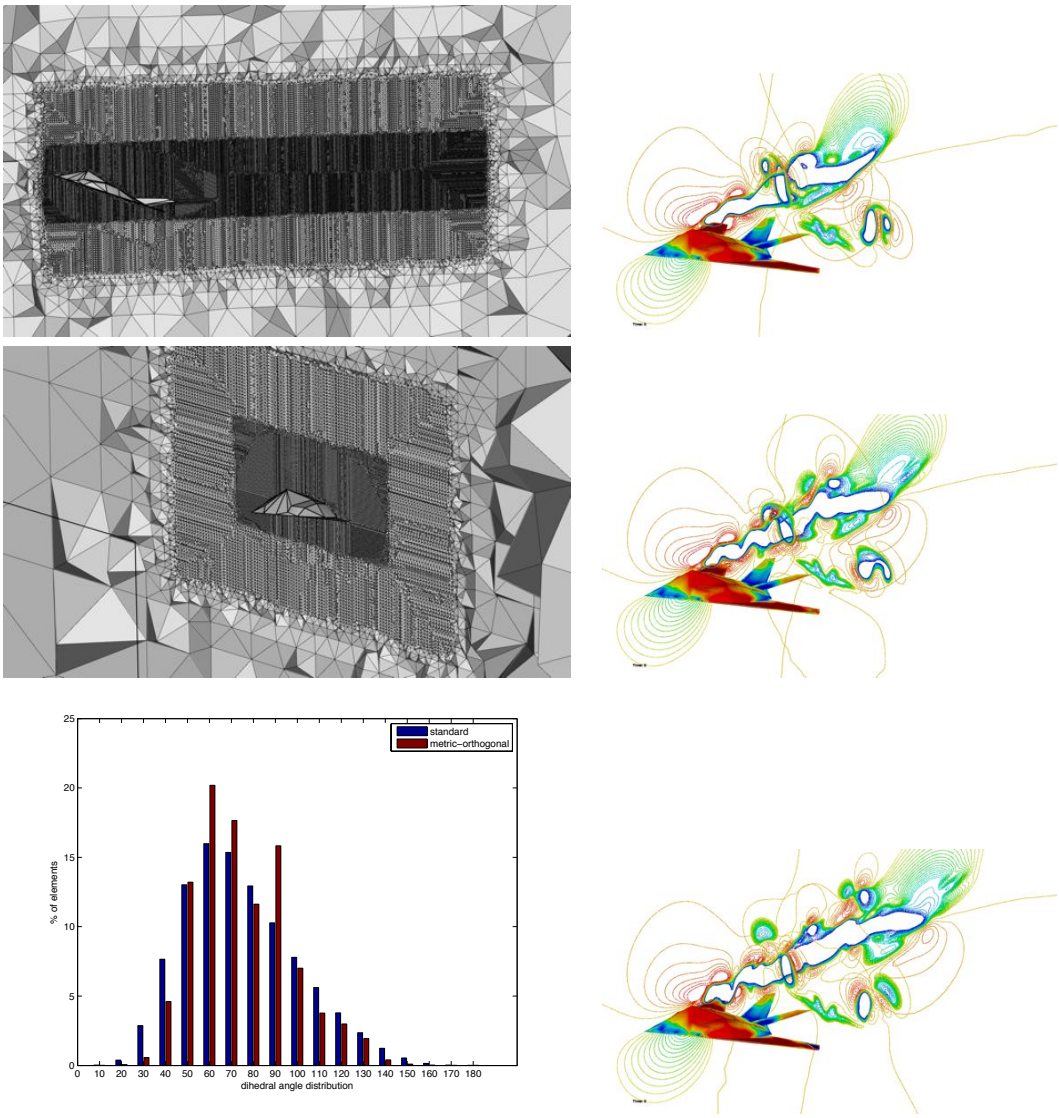

Fig. 6. Example of metric-orthogonal mesh generated from an analytical isotropic metric. Different views of the 3D mesh (left) and iso-lines of the vortices generated by the f117 at different times (right). Distribution of the dihedral angles with the standard and metric-orthogonal approaches (bottom left).

If the previous example is isotropic, this approach can be used to generate anisotropic meshes as well. We consider a transonic flow computation around a generic Falcon geometry at Mach 0.8 with an angle of attack of 3 degrees. We adapt the solution to the Mach number by controlling the interpolation error in $\mathbf{L}^2$ norm [25], the final mesh is obtained after 30 iterations and is composed of 1 110 735 vertices and 6 546 789 tetrahedra. The total cpu time is 40 minutes on an Intel Core i7 laptop at 2.7Ghz. Local orthogonal features appear clearly in the wake, see Figure 8 (bottom left). For all meshes, more than 95% of the edges have a unit length in the metric. A comparison with a standard mesh adaptation approach is given in Figure 9 (left) where dihedral angles are compared between the metric-orthogonal approach and the standard one. For the metric-orthogonal approach, we see that more than 25% (resp. 5% for the standard approach) of the elements contain right dihedral angles while minimizing the number of large dihedral angles. The number of elements with small angles is also greater with the standard approach. This reveals that the level of anisotropy is even higher for the metric-orthogonal approach.

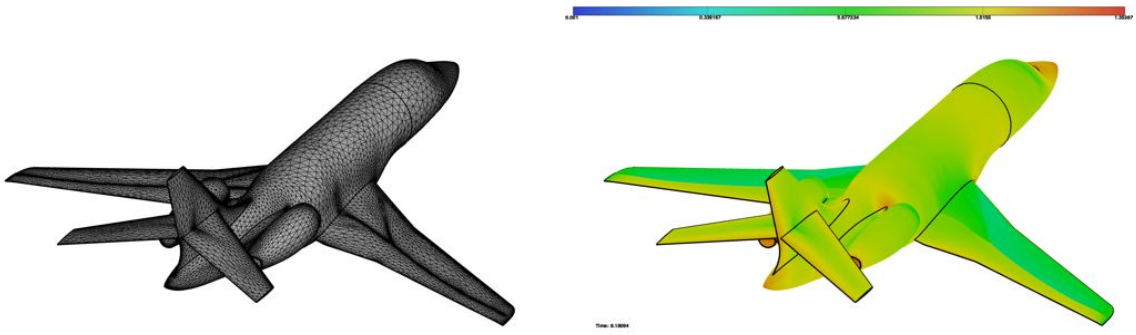

Fig. 7. Final adapted surface mesh on the falcon geometry (left) and density iso-values (right).

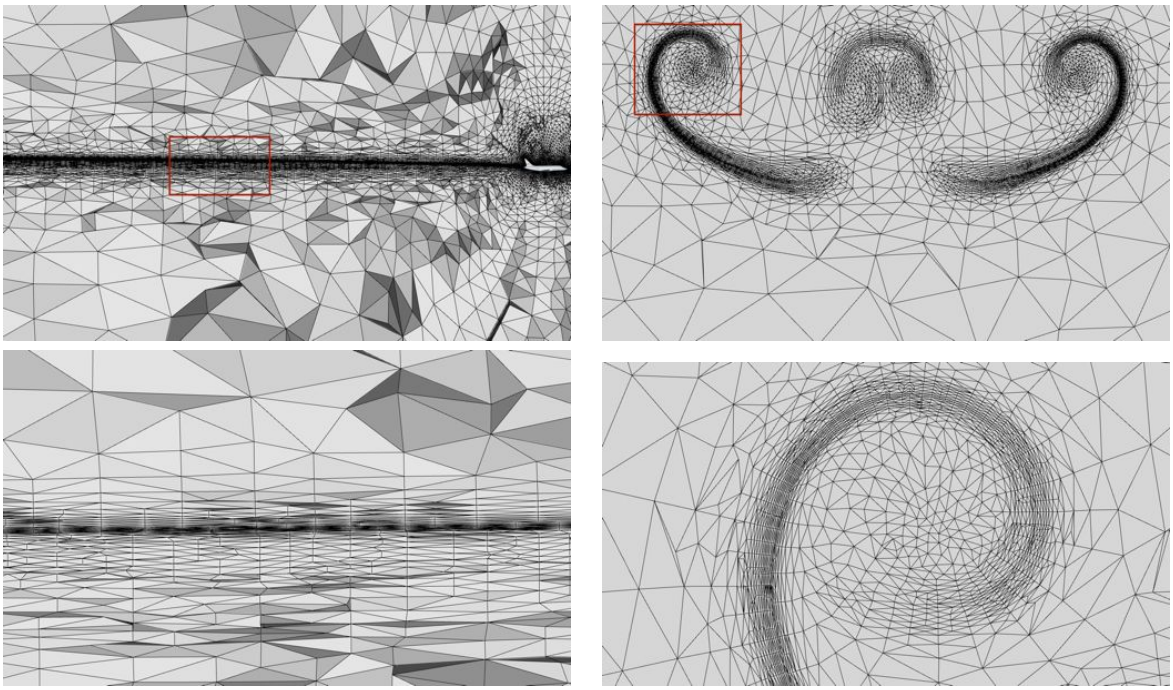

Fig. 8. Cuts in the final adapted volume mesh showing the aligned anisotropic elements: in the wake (top left), the vortex 100m behind the falcon (top right). Closer view in the red rectangles of the wake (bottom left) and the vortex (bottom right)

We now consider the example of a dam break on a rectangular obstacle. In this simulation, the compressible bi-fluid solver ANANAS© [17] is used. The error estimate is based on a level-set metric in order to capture and predict accurately the interface between the water and the air. The total CPU time for a physical time of $15s$ is $36h$ for the fully unstructured and $24h$ for the Cartesian approach. Both simulations were run on 4-cores of an Ivy-Bridge i7 at $3.4Ghz$. We observe in Figure 10 the dynamic of the flow. As the metric-orthogonal approach tends to insert less points (especially in the transverse direction), the CPU time is lower than the classical anisotropic approach. The distribution of the dihedral angles for the standard and metric-orthogonal approach are reported in Figure 9 (right) for time $0.85s$. The histogram shows that the angle distribution is centered around small angles and right angle whereas a uniform distribution is observed for the standard approach. Some cuts in the volume mesh are reported in Figure 11 at different times. It shows how the mesh around the interface is locally structured and how the edges are automatically aligned.

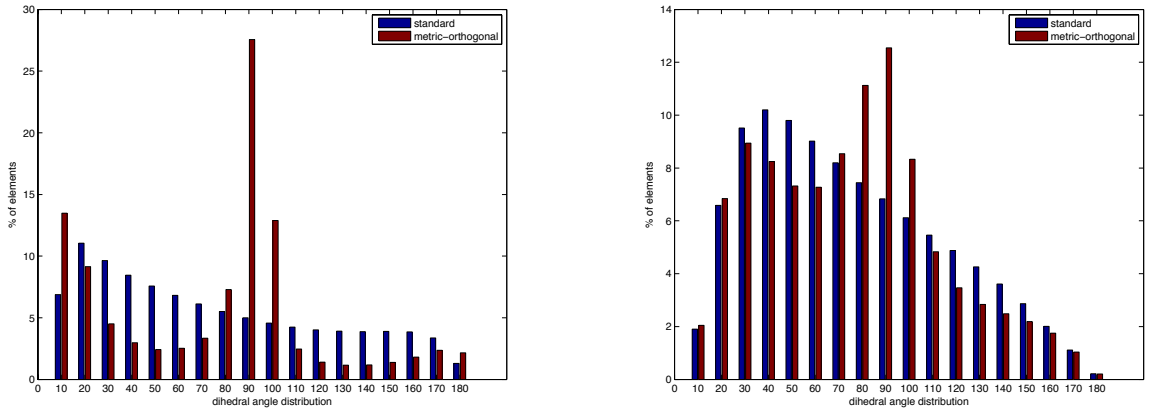

Fig. 9. Comparison of the dihedral angle distribution of the standard anisotropic mesh adaptation with the metric-orthogonal approach: falcon case (left) and dam break case (right).

6. Conclusion

We have proposed a strategy to generate metric-orthogonal anisotropic meshes. It is an extension of classical anisotropic mesh adaptation. In addition to being unit with respect to a metric, metric-orthogonal meshes are composed of elements that are aligned with the eigenvectors. This allows to recover locally structured meshes while keeping the same level of anisotropy. Dihedral angles are also optimized : the distributions are concentrated around small angles (needed for anisotropy) and right angles. The procedure is based on a combination of a cavity-based operator with a frontal algorithm.

The effectiveness of the method strongly depends on the quality and the properties of the input metric. Consequently, the current work is directed at improving usual anisotropic metric to comply with this metric-orthogonal kernel. In particular, size and orientation smoothing should be performed with care in order to improve the orthogonality and alignment properties of the final mesh. This strategy is currently extended to surface mesh generation.

References

[1] M. J. Aftosmis, M. J. Berger, and J. E. Melton. Adaptive cartesian mesh generation, 1999.
[2] F. Alauzet and A. Loseille. High-order sonic boom modeling based on adaptive methods. *J. Comput. Phys.*, 229(3):561–593, February 2010.
[3] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Log-Euclidean metrics for fast and simple calculus on diffusion tensors. *Magnetic Resonance in Medicine*, 56(2):411–421, August 2006.
[4] T. Baker. Three-dimensional mesh generation by triangulation of arbitrary point sets. *AIAA Paper*, 1987-1124, 1987.
[5] C. L. Bottasso. Anisotropic mesh adaption by metric-driven optimization. *Int. J. Numer. Meth. Engng*, 60:597–639, 2004.
[6] M. J. Castro-Díaz, F. Hecht, B. Mohammadi, and O. Pironneau. Anisotropic unstructured mesh adaptation for flow simulations. *Int. J. Numer. Meth. Fluids*, 25:475–491, 1997.
[7] T. Coupez. Génération de maillages et adaptation de maillage par optimisation locale. *Revue Européenne des Éléments Finis*, 9:403–423, 2000.
[8] C. Dobrzynski and P. J. Frey. Anisotropic delaunay mesh adaptation for unsteady simulations. In *Proc. of 17th Int. Meshing Rountable*, pages 177–194. Springer, 2008.
[9] P. J. Frey. `Yams`, a fully automatic adaptive isotropic surface remeshing procedure. RT-0252, INRIA, 2001.
[10] P.J. Frey and P.-L. George. *Mesh generation. Application to finite elements*. ISTE Ltd and John Wiley & Sons, 2nd edition, 2008.
[11] P.-L. George and H. Borouchaki. *Delaunay triangulation and meshing : application to finite elements*. Hermès Science, Paris, Oxford, 1998.
[12] P.-L. George and H. Borouchaki. Back to edge flips in 3 dimensions. In *Proceedings of the 12th International Meshing Roundtable*, pages 393–402, Santa Fe, NM, USA, 2003.
[13] P.L. George. `Gamanic3d`, adaptive anisotropic tetrahedral mesh generator. Technical Note, INRIA, 2003.
[14] P.L. George, F. Hecht, and E. Saltel. Fully automatic mesh generator for 3d domains of any shape. *Impact of Comuting in Science and Engineering*, 2:187–218, 1990.
[15] F. Hecht. `BAMG`: bidimensional anisotropic mesh generator. Available from http://www-rocq.inria.fr/gamma/cdrom/www/bamg/eng.htm, INRIA-Rocquencourt, France, 1998.
[16] F. Hecht and B. Mohammadi. Mesh adaptation by metric control for multi-scale phenomena and turbulence. *AIAA Paper*, 97-0859, 1997.

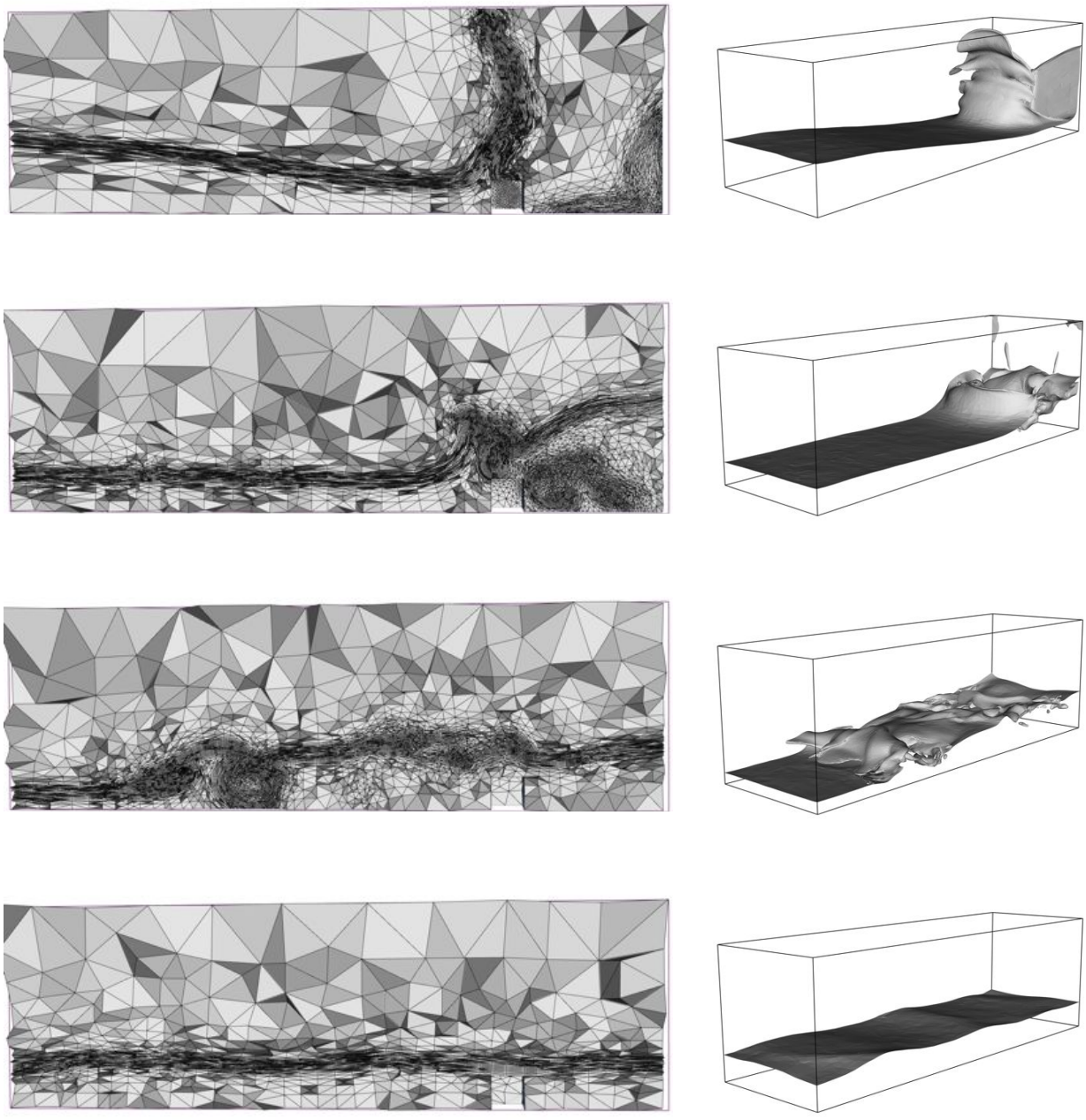

Fig. 10. Dam break simulation: cuts in the volume meshes for times $0.85s$, $1s50$, $2.50s$, and $7s50$ (left) and representation of the interface water/air (right).

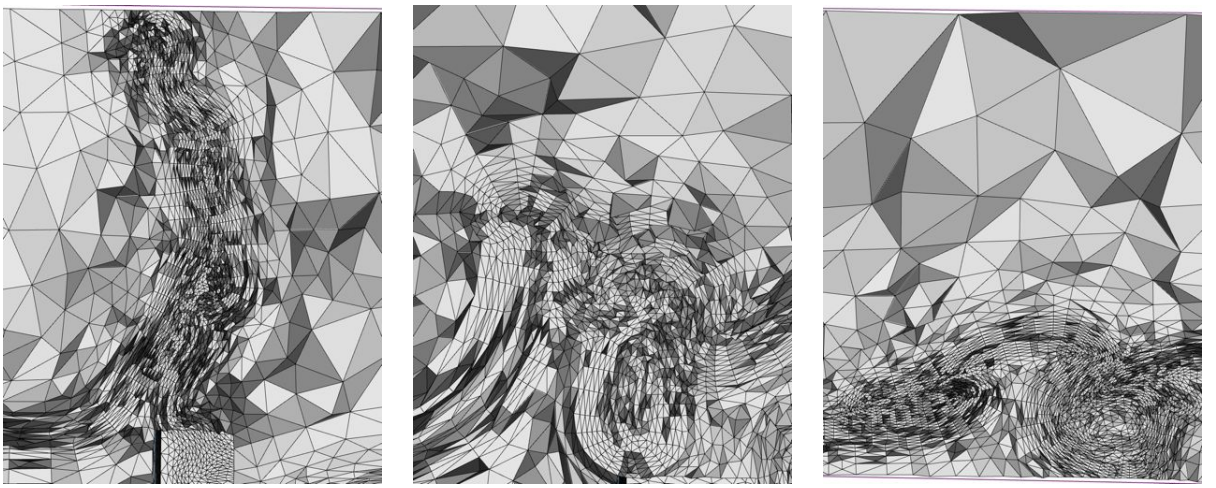

Fig. 11. Dam break simulation: Close view in the volume meshes for time $0.85s$, and $1s50$ and $2.50s$.

[17] Lemma Ing. ANANAS `http://www.lemma-ing.com`.

[18] W. T. Jones, E. J. Nielsen, and M. A. Park. Validation of 3D adjoint based error estimation and mesh adaptation for sonic boom prediction. *AIAA Paper*, 2006-1150, 2006.

[19] Jens Krause and Paul Louis George. Construction d'un maillage 3-D anisotrope localement structuré. Rapport de recherche RR-4834, INRIA, 2003.

[20] P. Laug and H. Bourochaki. `BL2D-V2`, mailleur bidimensionnel adaptatif. RR-0275, INRIA, 2003.

[21] B. Lévy and N. Bonneel. Variational anisotropic surface meshing with voronoi parallel linear enumeration. In Xiangmin Jiao and Jean-Christophe Weill, editors, *Proceedings of the 21st International Meshing Roundtable*, pages 349–366. Springer Berlin Heidelberg, 2013.

[22] X. L. Li, M. S. Shephard, and M. W. Beall. 3D anisotropic mesh adaptation by mesh modification. *Comput. Methods Appl. Mech. Engrg.*, 194(48-49):4915–4950, 2005.

[23] R. Löhner and P. Parikh. Three-dimensionnal grid generation by the advancing-front method. *Int. J. Numer. Meth. Fluids*, 8(8):1135–1149, 1988.

[24] A. Loseille and F. Alauzet. Continuous mesh framework part i: Well-posed continuous interpolation error. *SIAM Journal on Numerical Analysis*, 49(1):38–60, 2011.

[25] A. Loseille and R. Löhner. On 3d anisotropic local remeshing for surface, volume, and boundary layers. In *Proceedings of the 18th International Meshing Roundtable*, pages 611–630. Springer, 2009.

[26] A. Loseille and R. Löhner. Adaptive anisotropic simulations in aerodynamics. *AIAA Paper*, 2010-169, 2010.

[27] D. Marcum and F. Alauzet. Aligned metric-based anisotropic solution adaptive mesh generation. In *Proceedings of the 23rd International Meshing Roundtable*. 2014.

[28] D. L. Marcum. Efficient generation of high-quality unstructured surface and volume grids. *Engrg. Comput.*, 17:211–233, 2001.

[29] D.J. Mavriplis. An advancing front delaunay triangulation algorithm designed for robustness. *J. Comp. Phys.*, 117:90–101, 1995.

[30] T. Michal and J. Krakos. Anisotropic mesh adaptation through edge primitive operations. *AIAA Paper*, 2011-0159, 2011.

[31] C. C Pain, A. P. Umpleby, C. R. E. de Oliveira, and A. J. H. Goddard. Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations. *Comput. Methods Appl. Mech. Engrg.*, 190:3771–3796, 2001.

[32] M. Sharbatdar and C. Ollivier Gooch. Anisotropic mesh adaptation: recovering quasi-structured meshes. *AIAA Paper*, 2013-0149, 2013.

[33] A. Tam, D. Ait-Ali-Yahia, M. P. Robichaud, M. Moore, V. Kozel, and W. G. Habashi. Anisotropic mesh adaptation for 3D flows on structured and unstructured grids. *Comput. Methods Appl. Mech. Engrg.*, 189:1205–1230, 2000.