Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 82 (2014) 428 – 444

Procedia Engineering

www.elsevier.com/locate/procedia

23rd International Meshing Roundtable (IMR23)

# Aligned Metric-Based Anisotropic Solution Adaptive Mesh Generation

David Marcum[a], Frédéric Alauzet[b,*]

[a]*Center for Advanced Vehicular Systems and Dept. of Mechanical Engineering, Mississippi State University, USA*
[b]*Gamma3 Team, INRIA Paris-Roquencourt, Le Chesnay, France*

**Abstract**

A new aligned metric-based anisotropic mesh generation procedure is presented. The approach uses an advancing-point type point placement that aligns the elements with the underlying metric field. It is based on well-proven methodology combined with new novel point-placement strategies that are aligned with an underlying metric field. This simple strategy produces solution-adapted anisotropic meshes that are optimal in size and alignment with the solution gradients upon which the metric field is based. The overall methodology is implemented in a 2D process that is fully extendable to 3D. Results for several analytical and actual CFD simulation examples are presented that demonstrate that aligned high-quality anisotropic meshes that are optimal for the solution process can be reliably generated.



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

*Keywords:* Unstructured mesh generation; advancing front; metric-based anisotropic mesh; mesh alignment; solution-adaptive mesh

## 1. Introduction

The benefits and fundamental concepts of metric-based mesh adaptation for dealing with anisotropic physical phenomena are well established [16]. Several successful examples [3–5] with real-life problems have proven its ability to efficiently improve the ratio between solution accuracy and the number of degrees of freedom. This success has been based on the following key points:

- Efficient adaptive anisotropic mesh generator that can handle extreme anisotropy
- Accurate metric-based anisotropic error estimates: feature-based or adjoint-based
- Appropriate operator on metrics: interpolation, intersection and gradation
- Accurate solution transfer for transient problems

There are several solver and meshing tools that utilize a solution adaptive metric-based concept. Some examples in 3D include Epic, Feflo.a, Forge3d, Fun3d, Gamanic3d, MAdLib, MeshAdap, Mmg3d, Mom3d, Tango, and AdapLibrary [4,7,8,11–13,18,22]. It is worth mentioning that many of these codes include adaptive remeshers based on local

* Frédéric Alauzet. Tel.: +33 1 39 63 57 93.
*E-mail address:* Frederic.Alauzet@inria.fr


Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23)
doi:10.1016/j.proeng.2014.10.402

mesh modifications and others are mesh generation codes using Delaunay kernels or related approaches. Existing approaches assume that mesh quality and edge length sufficiency are fully satisfied if true in metric space. However, element shape and alignment with the metric field also impacts accuracy and efficiency of the solution process. In addition, during mesh generation without alignment, a local loss of anisotropy can occur when an element is generated that is not aligned (as an ideal equilateral element in metric space can vary in shape from isotropic to highly anisotropic as it is rotated). There is a need to investigate approaches that can provide improvement in this aspect and the present work seeks to do that.

The approach used in this work is based on an existing advancing-front method with local-reconnection/edge,face-swapping for mesh connectivity optimization [19–21], which is known to generate very-high quality unstructured meshes as compared to Delaunay type approaches. A new extension of this methodology to metric-based anisotropic mesh generation wherein the advancing-front type point placement is modified to satisfy a metric space definition is used in our approach. All geometric operations are performed in metric space with a consistent operational framework as defined in [16].

In addition a novel point placement algorithm has been developed to align the mesh elements with the solution based metric field. When new points are created using the advancing-front point placement they are by default aligned in physical space with the edge (or face in 3D) that locally represents the front. In metric space the ideal point placement is chosen to form triangular or tetrahedral elements that are ideal in metric space and aligned in physical space. In our approach the alignment is performed with the local metric field, rather than solely with the frontal edge (or face). Further improvements can be obtained if we consider local topology. Using right-angle type advancing-point point placement [19], pseudo-structured type element topology is produced locally that can be fully aligned with the solution gradients by the metric field. With right-angle type placement two-points are created from the edge (or three-points from the face in 3D) that locally represents the front. Alignment with the metric produces elements that are naturally ideal with respect to the metric field and also pseudo-structured in the case of right-angle point placement. Results with our initial work in 2D are presented for analytical and actual cases along with a discussion on the natural extension to 3D.

## 2. Metric-based mesh generation

This section recalls the main concepts of metric-based anisotropic adapted mesh generation. Here the discussion is largely focused on 2D aspects used in the present work. A similar discussion focussed on 3D is presented in [20].

### 2.1. Basics of metric

An **Euclidean metric space** $(\mathbb{R}^2, \mathcal{M})$ is a vector space of finite dimension where the dot product is defined by a symmetric definite positive tensor $\mathcal{M}$, called **metric tensor**:

$$\mathbf{u} \cdot_{\mathcal{M}} \mathbf{v} = \langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{M}} = \langle \mathbf{u}, \mathcal{M}\mathbf{v} \rangle = {}^t\mathbf{u}\, \mathcal{M}\, \mathbf{v}\,, \quad \text{for } (\mathbf{u}, \mathbf{v}) \in \mathbb{R}^2 \times \mathbb{R}^2\,. \tag{1}$$

The dot product defined by $\mathcal{M}$ makes $\mathbb{R}^2$ become a vector space supplied by norm and distance. In these spaces, the length $\ell_{\mathcal{M}}$ of a segment **ab** is given by the distance between its extremities:

$$\ell_{\mathcal{M}}(\mathbf{ab}) = d_{\mathcal{M}}(\mathbf{a}, \mathbf{b}) = \|\mathbf{ab}\|_{\mathcal{M}}. \tag{2}$$

As metric tensor is symmetric, it is thus diagonalizable: $\mathcal{M} = {}^t\mathcal{R}\,\Lambda\,\mathcal{R}$, where $\mathcal{R}$ is an orthonormal matrix the lines of which are composed of the **eigenvectors** $(\mathbf{v}_i)_{i=1,2}$ of $\mathcal{M}$ verifying ${}^t\mathcal{R}\mathcal{R} = \mathcal{R}\,{}^t\mathcal{R} = \mathcal{I}_2$. $\Lambda = \mathrm{diag}(\lambda_i)$ is a diagonal matrix composed of the **eigenvalues** of $\mathcal{M}$, denoted $(\lambda_i)_{i=1,2}$, which are strictly positive. From the previous definition, we deduce that application $\Lambda^{\frac{1}{2}}\,\mathcal{R}$ where $\Lambda^{\frac{1}{2}} = \mathrm{diag}(\lambda_i^{\frac{1}{2}})$ defines the mapping from the physical space $(\mathbb{R}^2, \mathcal{I}_2)$, where $\mathcal{I}_2$ is the identity matrix, to the Euclidean metric space $(\mathbb{R}^2, \mathcal{M})$. We trivially recover: $\mathbf{u} \cdot_{\mathcal{M}} \mathbf{v} = {}^t\left((\Lambda^{\frac{1}{2}}\,\mathcal{R})\mathbf{u}\right) \cdot \left((\Lambda^{\frac{1}{2}}\,\mathcal{R})\mathbf{v}\right) = {}^t\mathbf{u}\mathcal{M}\mathbf{v}$. We are then able to deduce the expression of the 2D cross product, area and angle with respect to $\mathcal{M}$:

$$\mathbf{u} \times_{\mathcal{M}} \mathbf{v} = \sqrt{\det \mathcal{M}}\,(\mathbf{u} \times \mathbf{v})\,, \quad |K|_{\mathcal{M}} = \frac{1}{2}\|\mathbf{u} \times_{\mathcal{M}} \mathbf{v}\| = \sqrt{\det \mathcal{M}}\,|K|_{\mathcal{I}_2} \quad \text{and} \quad \cos(\theta_{\mathcal{M}}) = \frac{\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{M}}}{\|\mathbf{u}\|_{\mathcal{M}} \|\mathbf{v}\|_{\mathcal{M}}}\,,$$

where $|K|_{\mathcal{I}_2}$ is the Euclidean area of triangle $K$ and, $\mathbf{u}$ and $\mathbf{v}$ are two non-zero vectors.

We will often refer to the geometric interpretation of a metric tensor. The set of points that are at distance 1 of a point $\mathbf{a}$ is given by $\mathcal{B}_{\mathcal{M}} = \left\{\mathbf{x} \mid \sqrt{{}^t(\mathbf{x}-\mathbf{a})\,\mathcal{M}\,(\mathbf{x}-\mathbf{a})} = 1\right\}$. It defines an ellipse centered at $\mathbf{a}$ with its axes aligned with the eigen directions of $\mathcal{M}$. Sizes along these directions are given by $h_i = \lambda_i^{-\frac{1}{2}}$. This ellipse depicted in Figure 1 (left). Notice that application $\Lambda^{\frac{1}{2}}\mathcal{R}$ maps $\mathcal{B}_{\mathcal{M}}$ from physical space into the unit ball in the metric space and, conversely, application ${}^t\mathcal{R}\,\Lambda^{-\frac{1}{2}}$ maps the unit ball into $\mathcal{B}_{\mathcal{M}}$, see Figure 1 (right).

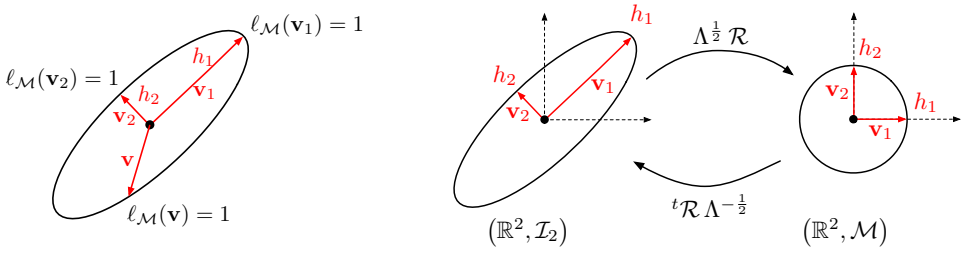


Fig. 1. Left, geometric interpretation of the unit ball $\mathcal{B}_{\mathcal{M}}$ where $\mathbf{v}_i$ are the eigenvectors of $\mathcal{M}$ and $\lambda_i = h_i^{-2}$ are the eigenvalues of $\mathcal{M}$. Right, mappings between physical space $(\mathbb{R}^2, \mathcal{I}_2)$ and Euclidean metric space $(\mathbb{R}^2, \mathcal{M})$.

In the context of mesh adaptation, the metric field is varying throughout the domain. We then work in a **Riemannian metric space** defined by $\mathbf{M} = (\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$. In that specific case, there is no global notion of scalar product. However, the notions of length and area defined for Euclidean metric spaces can be extended to Riemannian metric spaces which will be the main operations performed by the mesh generator. To take into account the variation of the metric along the edge, the edge length is evaluated with an integral formula. For a Riemannian metric space $\mathbf{M} = (\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$, the length of edge $\mathbf{ab}$ is computed using the line parameterization $\gamma(t) = \mathbf{a} + t\,\mathbf{ab}$, where $t \in [0,1]$:

$$\ell_{\mathcal{M}}(\mathbf{ab}) = \int_0^1 \|\gamma'(t)\|_{\mathcal{M}}\,\mathrm{d}t = \int_0^1 \sqrt{{}^t\mathbf{ab}\;\mathcal{M}(\mathbf{a}+t\,\mathbf{ab})\;\mathbf{ab}}\,\mathrm{d}t\,, \tag{3}$$

and, given a bounded subset $K$ of $\Omega$, the area of $K$ computed with respect to $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ is:

$$|K|_{\mathcal{M}} = \int_K \sqrt{\det \mathcal{M}(\mathbf{x})}\,\mathrm{d}\mathbf{x}\,. \tag{4}$$

### 2.2. Metric-based mesh generation

The main idea of metric-based mesh adaptation, initially introduced in [10], is to generate a unit mesh in a prescribed Riemannian metric space. A triangle $K$, defined by its list of edges $(\mathbf{e_i})_{i=1..3}$, is a **unit element** with respect to a metric tensor $\mathcal{M}$ if the length of all its edges is unit in metric $\mathcal{M}$:

$$\forall i = 1, ..., 3, \quad \ell_{\mathcal{M}}(\mathbf{e_i}) = 1 \text{ with } \ell_{\mathcal{M}}(\mathbf{e_i}) = \sqrt{{}^t\mathbf{e_i}\,\mathcal{M}\,\mathbf{e_i}}.$$

If all edges of $K$ are of unit length, then its area $|K|_{\mathcal{M}}$ in $\mathcal{M}$ is constant equal to:

$$|K|_{\mathcal{M}} = \frac{\sqrt{3}}{4} \quad \text{and} \quad |K|_{\mathcal{I}_2} = \frac{\sqrt{3}}{4}(\det(\mathcal{M}))^{-\frac{1}{2}}.$$

The notion of unit mesh is far more complicated than the notion of unit element as the existence of a mesh composed only of unit regular simplices with respect to a given Riemannian metric space is not guaranteed. Therefore, the notion of unit mesh has to be relaxed. A discrete mesh $\mathcal{H}$ of a domain $\Omega \subset \mathbb{R}^n$ is a **unit mesh** with respect to Riemannian metric space $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ if all its elements are quasi-unit. The definition of unity for an element is relaxed by taking

into account technical constraints imposed by mesh generators. For instance, to avoid cycling while analyzing edges lengths in a local remesher, a triangle $K$ defined by its list of edges $(\mathbf{e_i})_{i=1...3}$ is said to be quasi-unit if, $\forall i,\ \ell_{\mathcal{M}}(\mathbf{e_i}) \in [\frac{1}{\sqrt{2}}, \sqrt{2}]$. Consequently, the generated mesh is uniform and isotropic in the Riemannian metric space while it is adapted and anisotropic in the Euclidean space. This idea has turned out to be a huge breakthrough in the generation of anisotropic adapted meshes.

### 2.3. Practical use of metrics

The main advantage when working with metric spaces is the well-posedness of operations on metric tensors which enable us to manage directional sizes.

#### 2.3.1. Metric Interpolation

In practice, the Riemannian metric space or metric field is only known discretely at mesh vertices. The definition of an interpolation procedure on metrics is therefore mandatory to be able to compute the metric at any point of the domain. We consider the log-Euclidean framework introduced in [6]. We first define the notion of metric logarithm and exponential: $\ln(\mathcal{M}) := {}^t\mathcal{R}\ln(\Lambda)\mathcal{R}$ and $\exp(\mathcal{M}) := {}^t\mathcal{R}\exp(\Lambda)\mathcal{R}$, where $\ln(\Lambda) = \mathrm{diag}(\ln(\lambda_i))$ and $\exp(\Lambda) = \mathrm{diag}(\exp(\lambda_i))$. We then define the *logarithmic addition* $\oplus$ and the *logarithmic scalar multiplication* $\odot$:

$$\mathcal{M}_1 \oplus \mathcal{M}_2 := \exp\left(\ln(\mathcal{M}_1) + \ln(\mathcal{M}_2)\right) \quad \text{and} \quad \alpha \odot \mathcal{M} := \exp\left(\alpha.\ln(\mathcal{M})\right) = \mathcal{M}^\alpha\,.$$

The space of metric tensors, supplied with the logarithmic addition $\oplus$ and the logarithmic scalar multiplication $\odot$ is a vector space. We can now easily define the metric interpolation in the log-Euclidean framework. Let $(\mathbf{x}_i)_{i=1...k}$ be a set of vertices and $(\mathcal{M}(\mathbf{x}_i))_{i=1...k}$ their associated metrics. Then, for a point $\mathbf{x}$ of the domain such that: $\mathbf{x} = \sum_{i=1}^k \alpha_i.\mathbf{x}_i$ with $\sum_{i=1}^k \alpha_i = 1$, the interpolated metric is defined by:

$$\mathcal{M}(\mathbf{x}) = \bigoplus_{i=1}^{k} \alpha_i \odot \mathcal{M}(\mathbf{x}_i) = \exp\left(\sum_{i=1}^{k} \alpha_i\ \ln(\mathcal{M}(\mathbf{x}_i))\right). \tag{5}$$

This interpolation is commutative. Moreover, it has been demonstrated in [6] that this interpolation preserves the maximum principle, *i.e.,* for an edge $\mathbf{pq}$ with endpoints metrics $\mathcal{M}(\mathbf{p})$ and $\mathcal{M}(\mathbf{q})$ such that $\det(\mathcal{M}(\mathbf{p})) < \det(\mathcal{M}(\mathbf{q}))$ then we have $\det(\mathcal{M}(\mathbf{p})) < \det(\mathcal{M}(\mathbf{p} + 1t\,\mathbf{pq})) < \det(\mathcal{M}(\mathbf{q}))$ for all $t \in [0, 1]$.

#### 2.3.2. Numerical computation of geometric quantities in Riemannian metric spaces

Approximation can be used to evaluate edge length in Riemannian metric space given by Relation (3). However, an analytical expression can be obtained if we consider that the metric field conforms to a geometric variation law as described above. Let $\mathbf{e} = \mathbf{p_1p_2}$ be an edge of the mesh of Euclidean length $\|\mathbf{e}\|_2$, and $\mathcal{M}(\mathbf{p}_1)$ and $\mathcal{M}(\mathbf{p}_2)$ be the metrics at the edge extremities $\mathbf{p}_1$ and $\mathbf{p}_2$. We denote by $\ell_{\mathcal{M}_i}(\mathbf{e}) = \sqrt{{}^t\mathbf{e}\mathcal{M}(\mathbf{p}_i)\mathbf{e}}$ the length of the edge in metric $\mathcal{M}(\mathbf{p}_i)$. We assume $\ell_{\mathcal{M}_1}(\mathbf{e}) > \ell_{\mathcal{M}_2}(\mathbf{e})$ and we set $a = \dfrac{\ell_{\mathcal{M}_1}(\mathbf{e})}{\ell_{\mathcal{M}_2}(\mathbf{e})}$. Considering a geometric interpolation law of the metric field, we deduce [1]:

$$\ell_{\mathcal{M}}(\mathbf{e}) = \ell_{\mathcal{M}_1}(\mathbf{e})\frac{a-1}{a\ln(a)}\,. \tag{6}$$

The evaluation of a triangle area in a Riemannian metric space consists in computing numerically Integral (4). Higher order approximation can be obtained by using Gaussian quadrature and metric interpolation based on the Log-Euclidean framework. For faster area evaluation, a first order approximation is considered in this work:

$$|K|_{\mathcal{M}} = \int_K \sqrt{\det \mathcal{M}}\mathrm{d}\mathbf{x} \approx \sqrt{\det \mathcal{M}_K}\ |K|_{I_3} \quad \text{where} \quad \mathcal{M}_K = \exp\Big(\frac{1}{3}\sum_{i=1}^{3} \log\left(\mathcal{M}_{\mathbf{p}_i}\right)\Big). \tag{7}$$

and $\mathbf{p}_i$ are the three vertices of triangle $K$.

## 3. Advancing-Front with Local Reconnection Algorithm

The overall Advancing-Front with Local Reconnection (AFRL) mesh generation procedure used in the present work is a combination of automatic point creation, advancing type ideal point placement, and connectivity optimization schemes. A valid mesh is maintained throughout the mesh generation process. This provides a framework for implementing efficient local search operations using a simple data structure. Points are generated using either advancing-front type point placement for traditional equilateral type elements or advancing-point type point placement for right angle elements (cf. Figure 4). The connectivity for new points is initially obtained by direct subdivision of the elements that contain them. Connectivity is then optimized by local-reconnection/edge(face)-swapping with a combined Delaunay/min-max type (minimize the maximum angle, maximize the edge weight, etc.) type criterion. The overall procedure is applied repetitively until a complete field mesh is obtained. The basic steps in the procedure are briefly outlined below and illustrated in Figure 2. More complete details and results are presented in [19,21].

1. Generate a valid initial triangulation of the boundary points only and recover all boundary edges (faces in 3D).
2. Assign a sizing function to each boundary point based on the local point spacing (edge lengths) or a background metric field.
3. For traditional equilateral type elements, generate points using advancing-front type point placement. Points are generated by advancing from the edge (face in 3D) of elements that only satisfy the local sizing function on one edge (face).
4. For right angle type elements, generate points using advancing-point type point placement. Points are generated by advancing as in the previous step, except multiple points are created by advancing along normals from the two (three) points of the edge (face).

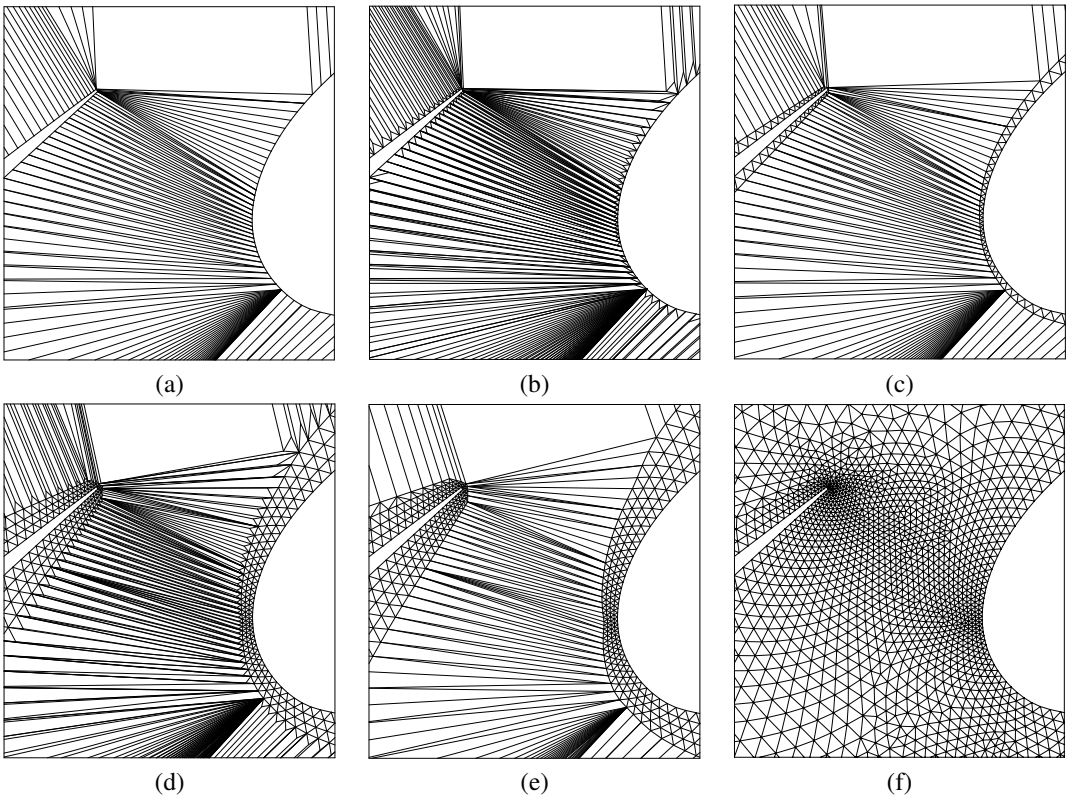


Fig. 2. *AFLR process. (a) resulting mesh after the boundary recovery phase. From (b) to (e), several steps of the point insertion coupled with connectivity optimization where we can see the evolution of the front. (f) Final mesh after mesh quality optimization.*

5. Set the sizing function for new points from interpolation on the existing mesh or the background metric field.
6. Reject or average new points that are too close to other new points.
7. Insert the accepted new points by directly subdividing the elements that contain them. The resulting triangulation is of very poor quality and will need to be improved.
8. Optimize the connectivity using local-reconnection (edge/face swapping). For each element pair, compare the reconnection criterion for all possible configurations and reconnect using the most optimal one. Repeat this local-reconnection process until no elements are reconnected. The only characteristic required of the starting triangulation is that it be valid.
9. Repeat the point generation and local-reconnection process, described above, until no new points are generated.
10. Smooth the coordinates.
11. Optimize the connectivity using the local-reconnection process.

## 4. Aligned metric-based anisotropic mesh generation

### *4.1. Local remeshing quality issues*

As stated in the introduction, local remeshing strategies [4,7,8,11–13,18,22] have been very successful in generating highly anisotropic adapted meshes mainly because they end-up to be very robust. Indeed, the main idea is to modify an existing valid mesh to generate the desired adapted mesh. Therefore, any operations which may create an invalid element is simply rejected. Usually, the local remeshing algorithm consists in analyzing the length of all the mesh's edges: edges that are to long are split into two and edges that are too short are collapsed. Then, the mesh quality is optimized using local reconnection and vertex relocation. One can immediately see that the usual algorithm cannot reach exactly a size of 1 for all edges and it cannot control explicitly control the point insertion location as this depends on the analyzed edge. Moreover, it is very difficult to control the angle of the generated elements.

Figure 3 shows a fully unstructured anisotropic adapted mesh obtained with a well-establish remeshing method, *i.e.,* `Yams` [9]. The mesh globally fulfills the size and the orientation of the metric field and overall quality of the elements is excellent in metric space. However, in physical space a majority of elements have obtuse angles and the overall quality of the elements is poor. While such a mesh will produce good accuracy with typical solvers, the solver efficiency and stability can be adversely impacted by the physical space quality. The issue of physical space quality will be emphasized in Section 5. It is clear that the ideal situation would be to produce meshes that are optimal in both physical and metric spaces. An advancing-front method combined with a proper choice of point placement strategy and metric alignment is a good candidate to solve this issue.

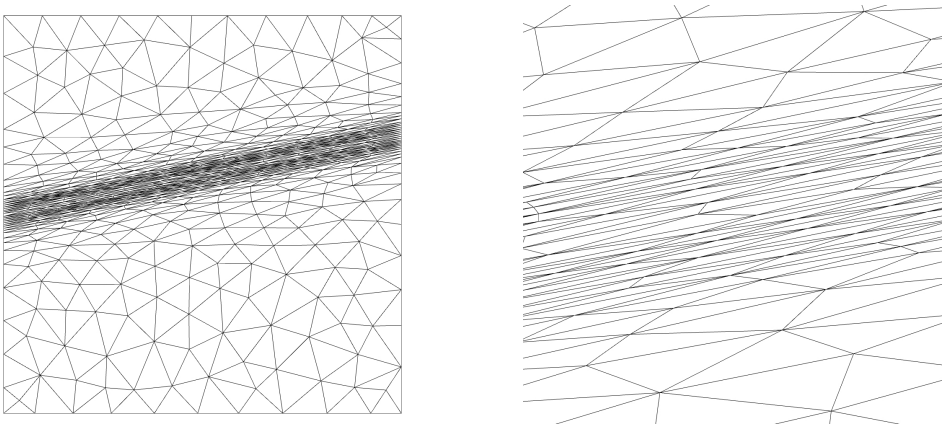

Fig. 3. *Anisotropic adapted mesh obtained with well-establish local remesher. The mesh globally fulfills the size and the orientation of the metric field but locally a majority of elements have obtuse angle.*

### 4.2. Metric-based AFLR issues

The AFLR algorithm - given in Section 3 - can utilize a sizing function that can be defined by a simple isotropic length scale based on local edge lengths or a anisotropic metric field. For anisotropic elements a generalized metric approach is used. Modifications to the original algorithm for anisotropy are accomplished by using a functional approach for calculation of all geometric properties such as dot products, cross products, reconnection criterion, etc. Those modifications are described in Section 2.2 and in [20]. Moreover, a dynamic representation of the metric field is needed during mesh generation. In the present work a metric field is generated from a previous solution and mesh. To maintain the accuracy of the representation of the metric, the logarithm of the metric $\ln(\mathcal{M})$ is stored for each vertex on the given background mesh. Each time, a point is inserted inside the mesh, it is localized in the background mesh and its metric is interpolated according to Formula (5). The background mesh and metric field can also be represented by an analytical function for testing purposes.

However, this leads to a mesh generation method equivalent to a local remeshing approach in resulting physical space quality and doesn't fully take advantage of using an advancing-front meshing strategy. This shape quality issue is due to the **non-alignment** of the edges of the mesh with the metric field. This will be pointed out in Sections 4.3 and 5.1. The unmodified advancing-front approach is lacking an ability to align the elements generated with the metric field. Modifications are required in order to fully control the point placement by adding alignment. This is the purpose of the next section.

### 4.3. Study of point placement strategy in metric spaces

The basics of the two different point insertion strategies previously described in Section 3 and in [19,21] are illustrated in Figure 4. The **advancing-front type placement** leads to generation of equilateral type elements. In this case, each active edge **e** (the local front) proposes one point from the middle of the edge and in the orthogonal direction at a distance $\ell = \frac{\sqrt{3}}{2}\ell(\mathbf{e})$. The **advancing-point type placement** leads to generation of right angle type elements from which quads can be created. In this case, each active edge **e** (the local front) proposes two points, one for each vertex. These points are issued from each edge vertex and in the orthogonal direction at a distance $\ell = \ell(\mathbf{e})$.

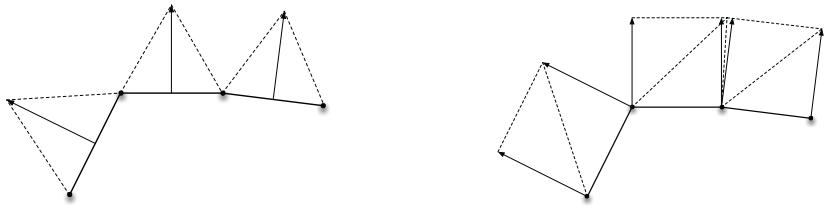

Fig. 4. *AFLR point placement strategies. Left, standard advancing-front type placement. Right, right-angle advancing-point type placement.*

Now, we analyze the behavior of each strategy in the context of metric-based mesh generation depending whether the front - the active edge - is or is not aligned with the metric direction, *i.e.,* the metric eigenvector.

In the case of active edges not aligned with the metric direction, we notice for both methods that if the point placement is not aligned with the metric eigenvectors, *i.e.,* it is still done orthogonally to the active edge, this leads to a loss of anisotropy and the creation of almost isotropic element in physical space while the element is unit in metric space. On the contrary, if the metric eigenvector direction is used to propose the point then anisotropy is preserved, see Figure 5 (middle left vs. far left and far right vs. middle right). This adverse effect is illustrated in Figure 7 (far left and middle right) where adapted meshes have been generated while proposing point in the orthogonal direction of the active edge. We clearly see the creation of isotropic elements. This is not the case with point placement aligned with the metric as shown in Figure 7 (middle left and far right).

In the case of active edges aligned with the metric direction, we observe that the advancing-front type placement creates obtuse angle if the largest triangle side proposes a point in the smallest size direction, see Figure 6 (bottom left) and Figure 7 (middle left). However, nice angles are obtained if the smallest triangle side proposes a point, see Figure 6 (top left). With the advancing-point placement and metric alignment, good angles are generally produced and right angle shape elements are still preserved and quads can be produced, as shown in Figure 6 (top and bottom right) and Figure 7 (far right).

From the previous considerations, the adopted strategy is:

- Point insertion uses the advancing-point type placement.
- Point insertion follows the metric eigenvector direction to propose a new point.
- All geometric operations and quality functions are computed in Riemannian metric spaces.
- Metric field is defined on a background mesh and the log-Euclidean interpolation used to define the metric at new points.

The point placement algorithm with metric alignment must be modified as there is not a single unique metric that can be used for alignment and sizing. An iterative strategy is used wherein the metric at the starting point on the edge is initially used to place the proposed point. The metric at the proposed point is then evaluated from the background mesh. If it differs from the starting metric then the metric used for alignment and sizing is evaluated according to Formula (5) with the starting and proposed point metrics. If this produces a distinct difference in the proposed point location (as it does when the local metric field variation is large) then the metric is re-evaluated at the proposed point. This can be iterated upon until convergence. Alternative strategies such as mid-point evaluation (between the starting and proposed points) can also be used.

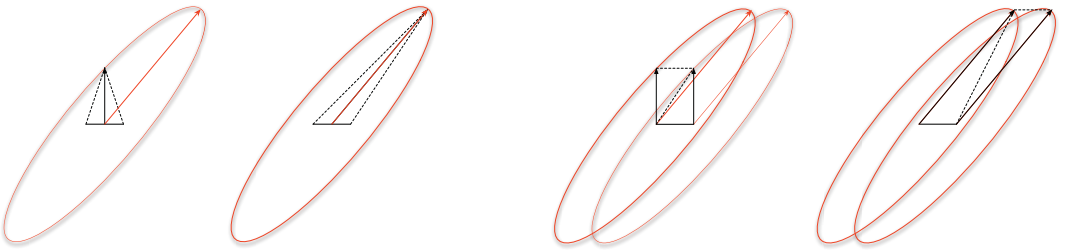

Fig. 5. *If the active edge is not aligned with the metric direction, then orthogonal point placement produces a local loss of anisotropy for both point placement strategies (far left and middle right). No loss of anisotropy is produced with point placement aligned with the metric (middle left and far right).*

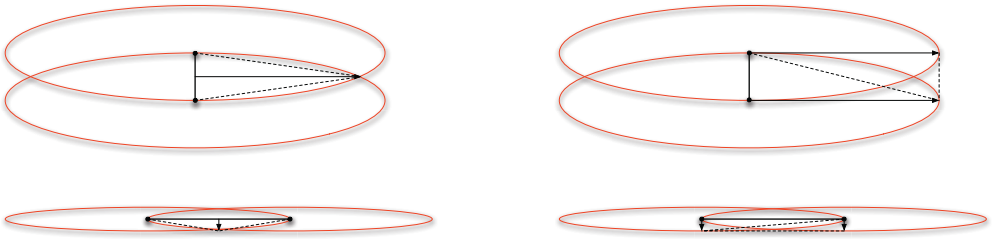

Fig. 6. *If the active edge is aligned with the metric direction, then good angles are obtained (top left, top right, and bottom right), except with advancing-front when a triangle with an obtuse angle is created (bottom left).*

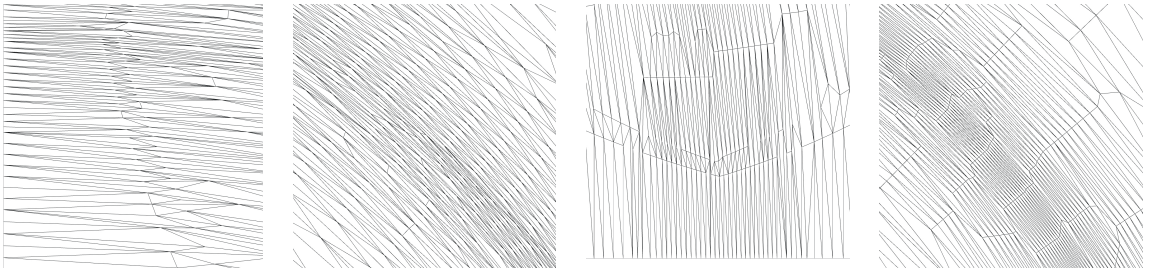

Fig. 7. *Illustration of the loss of anisotropy for a typical case using physical space alignment with the advancing-front placement (far left) and advancing-point placement (middle right). With metric alignment, obtuse angles can still be created with the advancing-front placement (middle left) and good angles are generally produced with advancing-point placement (far right).*

## 5. Numerical examples

Several 2D examples are presented to illustrate the effectiveness of the proposed metric-aligned point placement strategy for generating high-quality aligned anisotropic adapted meshes. For the presented results, the adapted meshes shown are obtained directly from the aligned anisotropic AFLR method without final quality improvement. This means that there is no edge collapse, smoothing or final mesh optimization. This better illustrates the effectiveness of the point placement. Further improvement is expected with those operations included. In addition, for the quad dominant meshes shown, no metric specific modifications have been incorporated and the merging of triangles can clearly be improved upon in this respect. The original quad formation algorithm [19] used in the results presented here is based on boundary/geometry driven alignment and in these cases we have metric-driven alignment. It also relies heavily on quad based smoothing that is not used in the present cases. The final optimization and quad formation algorithms were considered add-ons for the present work and while they will be added for production type usage, they were not considered essential to the present effort which is focused on metric-aligned point placement.

For the results presented in this section, CPU usage for mesh generation with metric alignment is minimal and of order 15 seconds or less on a MacBookPro laptop computer. No significant effort was put into optimization for efficiency and the CPU time with further work is expected to decrease significantly.

### 5.1. Analytic metric fields

In this section, the metric is defined analytically, meaning that each time a new point is inserted its exact metric value is evaluated. For these analytic examples, results using the metric-aligned advancing-front and advancing-point placement strategies are shown to emphasized the issues relative to the chosen strategy as mentioned in Section 4.2.

**Cross function.** The first metric function considered represents two straight anisotropic features that intersect and form a cross. The direction of these features is aligned with the unit square boundary geometry. The function is

$$\mathcal{M}_{cross}(x,y) = \begin{pmatrix} h_x^{-2} & 0 \\ 0 & h_y^{-2} \end{pmatrix} \quad \text{where} \quad \begin{cases} h_x = \min(2^{\alpha_x} \times h_{\min}, h_{\max}) \\ h_y = \min(2^{\alpha_y} \times h_{\min}, h_{\max}) \end{cases}$$

with $h_{\min} = 0.005$, $h_{\max} = 0.1$, $\alpha_x = 20 \times |x - 0.5|$ and $\alpha_y = 20 \times |y - 0.5|$. Thus, the initial front of active edges at the boundaries is perfectly aligned with the metric field and correspond to the smallest edge size. Thus, both AFLR metric-aligned point placement strategies should act properly.

Indeed, the resulting adapted meshes are perfectly aligned in the anisotropic regions, see Figure 8. We also observe that symmetry is preserved by both methods and the resulting adapted meshes are composed of only high-quality elements. The resulting angle distribution in physical space is excellent and appropriate with respect to the chosen point placement strategy as illustrated in Figure 8 (top right). In the quad dominant case, aligned anisotropic quads appears naturally. The quality at the junction is somewhat abrupt as no optimization has been used.

**Circle function.** The second metric function represents a curved anisotropic feature having the shape of a quarter circle. The domain is still a unit square. The analytical function reads:

$$\mathcal{M}_{circle}(x,y) = \begin{pmatrix} h_1^{-2}\cos^2\theta + h_2^{-2}\sin^2\theta & (h_1^{-2} - h_2^{-2})\cos\theta\sin\theta \\ (h_1^{-2} - h_2^{-2})\cos\theta\sin\theta & h_1^{-2}\sin^2\theta + h_2^{-2}\cos^2\theta \end{pmatrix} \quad \text{where} \quad \begin{cases} h_1 = \min(0.002 \times 5^{\alpha}, h_{\max}) \\ h_2 = \min(0.05 \times 2^{\alpha}, h_{\max}) \end{cases}$$

with $\theta = \arctan(x, y)$, $h_{\max} = 0.1$, $\alpha = 10 \times \left|0.75 - \sqrt{x^2 + y^2}\right|$.

The main difficulty of this example is to properly account for the change in direction of the metric field. We observe that the resulting adapted meshes are largely aligned with the metric field, see Figure 9. We notice - as expected - that the advancing-front placement creates large obtuse angle while the advancing-point placement preserves the right-angle triangle shape for the elements. This is emphasized in the angle distribution in physical space histogram shown in Figure 9 (top right) where the metric-aligned advancing-point placement achieved clearly a better distribution. In particular, we observe that 24% of the angle are greater than 150 degrees for the advancing-front placement while it drops to less than 2% for the advancing-point placement. In the quad dominant case, aligned anisotropic quads again appears naturally. The quality of the quad dominant mesh can clearly be improved by optimization and accounting for the metric alignment during quad formation.

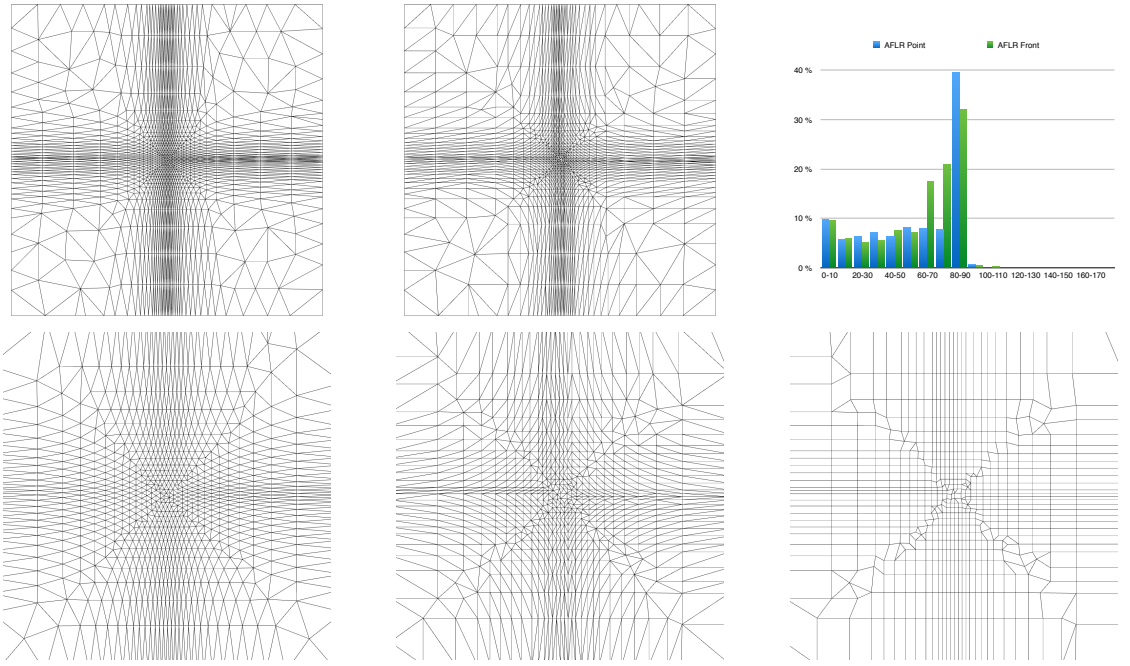


Fig. 8. *Cross analytical metric. Adapted meshes generated with advancing-front placement (left), advancing-point placement (middle) and its resulting quad dominant mesh (right). Top right, angle (in physical space) distribution histograms for the metric-aligned advancing-front placement (in green) and for the metric-aligned advancing-point placement (in blue) approaches.*

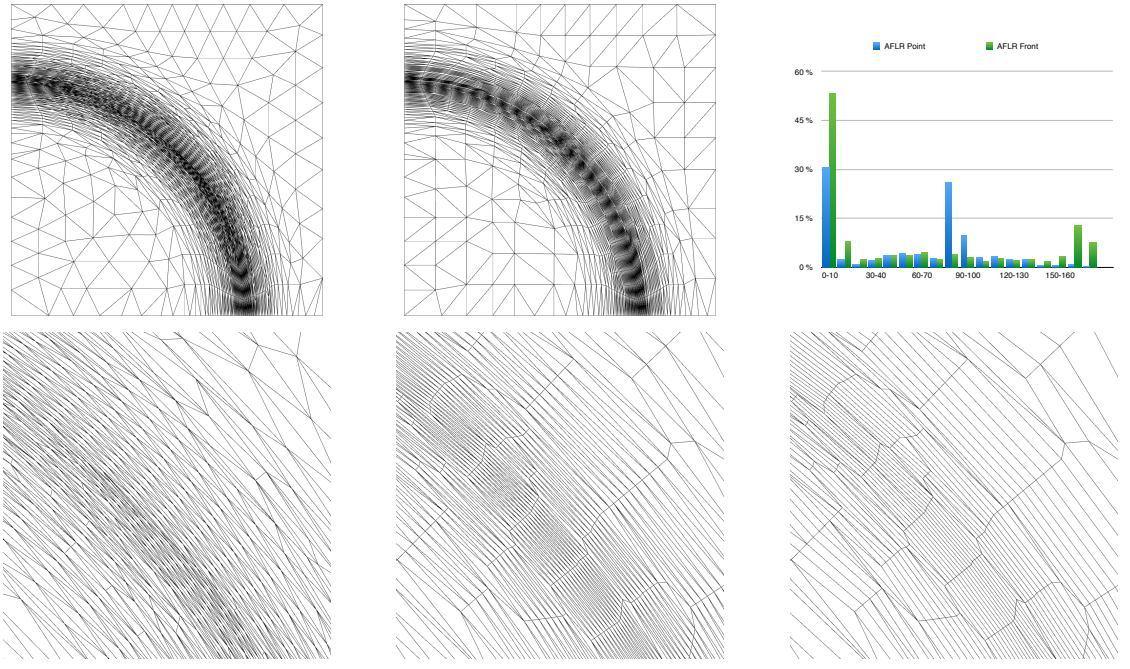


Fig. 9. *Circle analytical metric. Adapted meshes generated with advancing-front placement (left), advancing-point placement (middle) and its resulting adapted quad dominant mesh (right). Top right, angle (in physical space) distribution histograms for the metric-aligned advancing-front placement (in green) and for the metric-aligned advancing-point placement (in blue) approaches.*

### 5.2. Analytic functions defining a metric on the background mesh

In this section, the metric is computed numerically from a given analytical function linearly interpolated on a background mesh. The metric is given by the absolute value of the Hessian of the function which corresponds to control the interpolation error of the function in $L^\infty$-norm [2]. As the metric is defined on the background mesh: each time a new point is inserted the metric is interpolated to get its value. For these two analytic examples, we now compare results obtained with the metric-aligned advancing-point placement and a well-establish remeshing method using `Yams` [9].

**Diagonal function.** The first function is a simple diagonal anisotropic feature. Unlike the purely analytical case, the direction of this feature is not aligned with the unit square boundary. The function is

$$f_{diag}(x, y) = \tanh\left(50 \times (y - 0.2\,x - 0.5)\right) .$$

Comparison between the resulting adapted meshes for this easy function clearly demonstrates the improvement in quality obtained using alignment coupled with an metric-aligned advancing-point method, see Figure 10. With local remeshing, the mesh is globally aligned with the metric field and globally respects the size field. Locally, elements are almost aligned and almost of correct size. With metric-aligned advancing-point placement case, the mesh is better aligned globally and locally. Two diagonal anisotropic bands are expected and observed. Moreover, all elements have the same size in constant size regions. The mesh is quasi-structured resulting in a very-high quality quad dominant adapted mesh. We observe no bias due to the metric interpolation on the background mesh. The angle (in physical space) distribution histogram (for triangle meshes only), shown in Figure 10 (top right), highlights these comments where only 1.5% of the angle are greater than 120 degrees for the advancing-point strategy while this percentage rises to 29% for the local remeshing method, 18% being above 160 degrees.

We also notice that the local remeshing method generates a mesh with 45% more vertices for the same metric field (517 vs 356). Two reasons explain this difference. First, when a right-angle triangle is create with the advancing-point

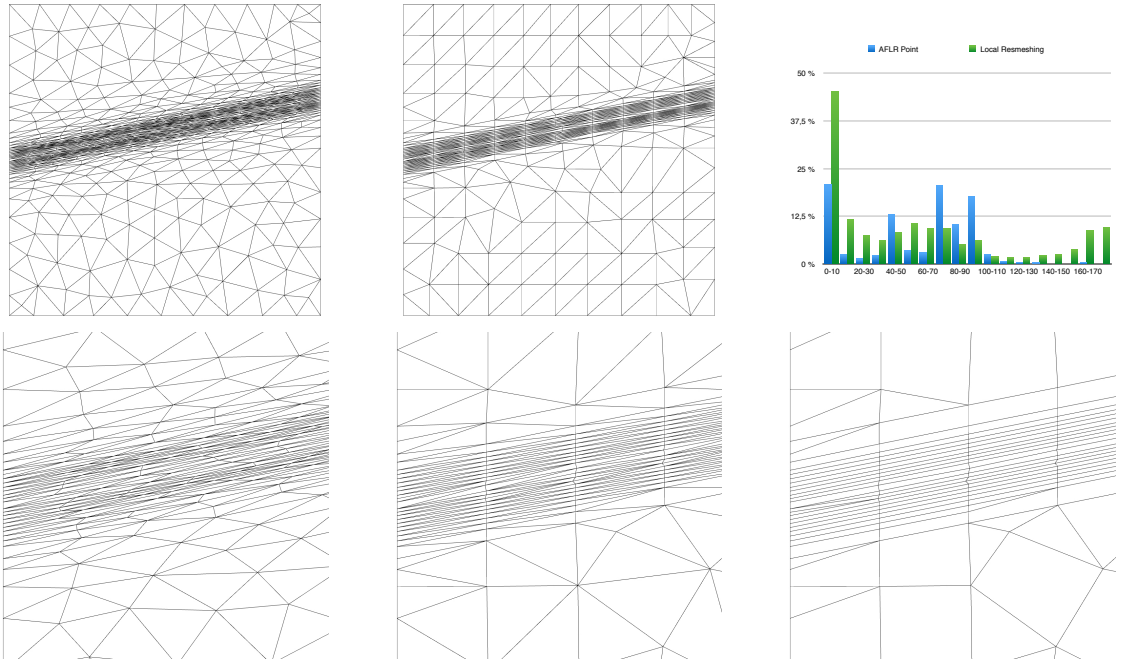


Fig. 10. *Diagonal analytical function. Adapted meshes generated with local remeshing (left), advancing-point placement (middle) and its resulting adapted quad dominant mesh (right). Top right, angle (in physical space) distribution histograms for the local remeshing (in green) and for the metric-aligned advancing-point placement (in blue) approaches.*

placement it has generally two edges of length 1 and one edge of length $\sqrt{2}$. Second, as a better aligned quasi-structured mesh is generated, it more optimally fills the space using less vertices. This leads to an average edge length of 1.13 for the new approach against an average edge length of 0.89 with the local remeshing.

**$\chi$-shape function.** The second function is more complicated and represent a $\chi$-shape feature. It has higher anisotropy than the previous examples and combines straight and curved features. The function reads:

$$f_\chi(x,y) = \tanh\left(-100 \times (y - 0.5 - 0.25\ \sin(2\pi\, x))\right) + \tanh\left(100 \times (y - x)\right)\ .$$

On this more complex function, the same conclusions as previously arise. A higher-quality mesh is clearly obtained with the metric-aligned advancing-point placement in comparison to the local remeshing approach, see Figure 11. With metric-aligned advancing-point placement a quasi-structured mesh is achieved in most regions which is emphasized by the quad-dominant mesh. The angle (in physical space) distribution is again improved (see Figure 11 (top right)): 32% and 10% of the angle are greater than 120 and 160 degrees for the local remeshing method, respectively, which has to be compared to 4.7% and 1.7% for the metric-aligned advancing-point strategy. The new method having 50% of the angles between 70 and 100 degrees.

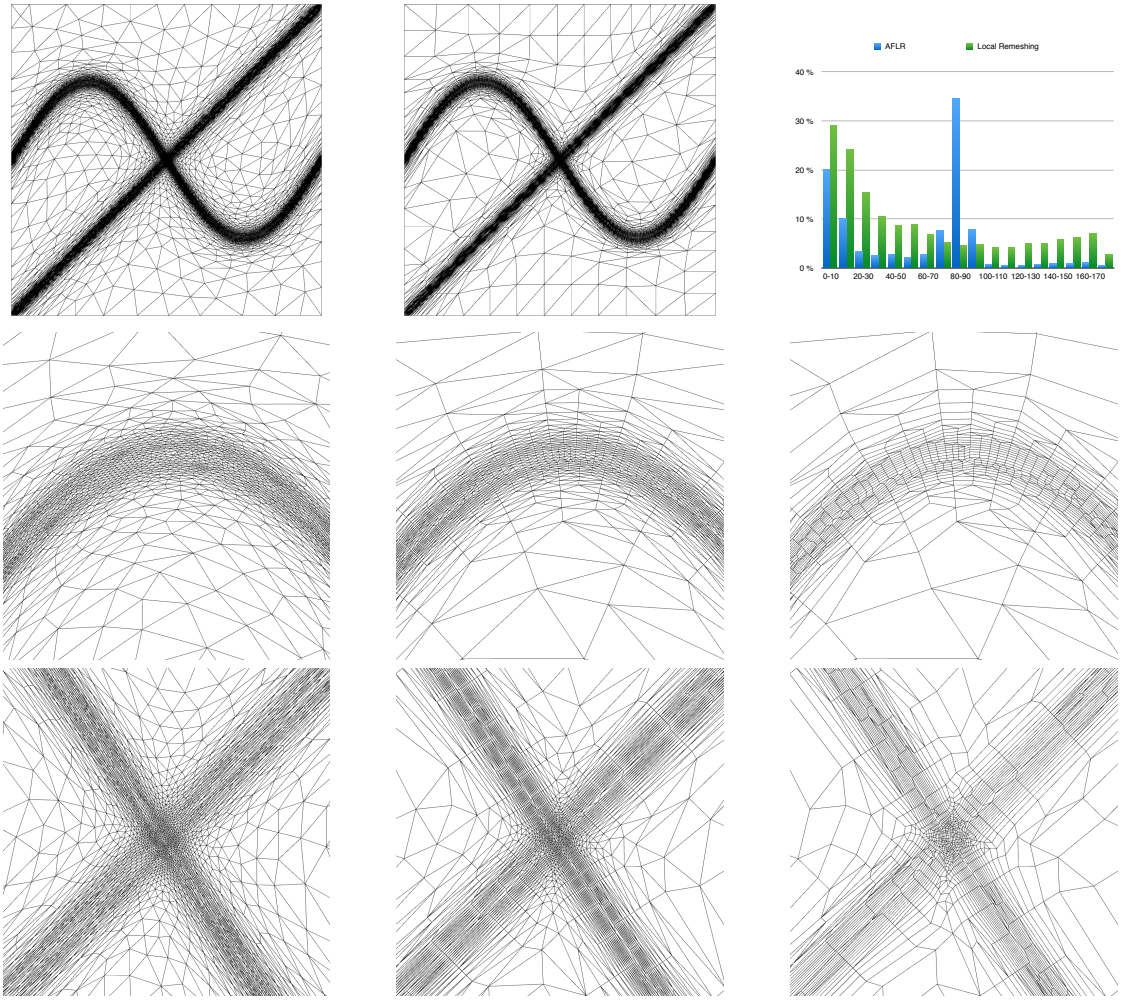


Fig. 11. *Chi-shape analytical function. Adapted meshes generated with local remeshing (left), metric-aligned advancing-point placement (middle) and its resulting adapted quad dominant mesh (right). Top right, angle (in physical space) distribution histograms for the local remeshing (in green) and for the metric-aligned advancing-point placement (in blue) approaches.*

Again in this case, the local remeshing method has generated a mesh containing 57% more vertices for the same metric field (6 793 vs 4 312) for the same reasons stated above. This leads to an average edge length of 1.21 for the new approach against an average edge length of 0.93 with the local remeshing.

### 5.3. CFD examples

Three CFD simulations have been run to compare a classic local remeshing approach using `Yams` [9] and the new metric-aligned advancing-point method proposed in this paper. In these cases the same non-adapted mesh is used as the starting point and the metric field is obtained from a corresponding CFD simulation in a classical solution-adapted mesh/solution manner.

**Supersonic scramjet.** The first CFD example is an internal steady flow representing a Scramjet engine configuration at Mach 3. This simulation is representative of compressible problems involving highly anisotropic phenomena with several strong shocks. Multi-scale anisotropic mesh adaptation is used during the simulation to control the interpolation error of the density flow variable in the $L^2$-norm [15]. Results are shown in Figures 13 and 14.

**Supersonic NACA0012.** The second simulation is a steady external flow around a NACA0012 at Mach 1.4 with an angle of attack equal to 0° where the bottom part of the domain represents the ground. In this simulation, a strong bow shock appears in front of the airfoil and is reflected by the ground. Goal-oriented anisotropic mesh adaptation is used during the simulation to control the error of the pressure signature on the ground [17]. Results are shown in Figures 15 and 16.

**Shock-bubble interaction.** The third simulation is from an unsteady supersonic straight shock wave impacting a bubble region of low density. When the shock wave interacts with the bubble, its shape is curved and vortices are created inside its wake. Time-dependent multi-scale anisotropic mesh adaptation is used during the simulation to control the interpolation error of the density flow variable in the $L^2$-norm [5]. A single point in time is shown in the results. Results are shown in Figures 17 and 18.

**Comments.** Similar to the analytical cases, we observe that the new metric-aligned advancing-point method ends up with higher quality anisotropic adapted meshes in comparison to classical remeshing. Good alignment of the elements is obtained even for these cases with a complex widely varying metric field generated from differing error estimates

Table 1. Adapted mesh number of vertices for the CFD examples.

| CFD cases | Metric-aligned advancing-point | Local remeshing |
|---|---|---|
| Supersonic scramjet | 6 354 | 9 216 |
| Supersonic NACA0012 | 5 274 | 7 257 |
| Shock-bubble interaction | 13 674 | 18 241 |

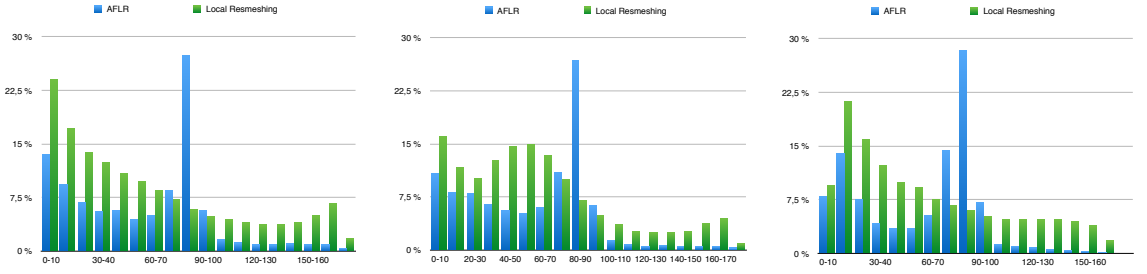


Fig. 12. *Angle distribution histograms for the three CFD simulations for the local remeshing (in green) and for the metric-aligned advancing-point placement (in blue) approaches.*

applied to a numerical CFD solution. Quasi-structured adapted meshes are readily obtained for these more complex features. This is again emphasized by the angle (in physical space) distribution histograms for each case shown in Figure 12. The number of angles greater than 120 degrees is considerably reduced by the new approach while a pick is obtained around 90 degrees as expected. The resulting meshes size, see Table 1, is again smaller with the new method for the same prescribed level of error for the reasons stated in Section 5.2. The local remeshing method generates approximately 50% more vertices in each case.

## 6. Conclusion

A new metric-aligned advancing-point placement strategy has been implemented in a metric-based anisotropic mesh generation procedure and results have been presented. The approach uses an advancing-point type point placement that aligns the elements with the underlying metric field. It is based on a well-proven methodology and adds novel point-placement strategies that produce solution-adapted anisotropic meshes that are optimal in size and alignment with the solution gradients upon which the metric field is based. The overall methodology was implemented in a 2D process that is be extendable to 3D. Results were presented for several examples, including analytical cases and actual CFD simulations. These results demonstrate that aligned high-quality anisotropic meshes that are more optimal for the solution process can be reliably generated.

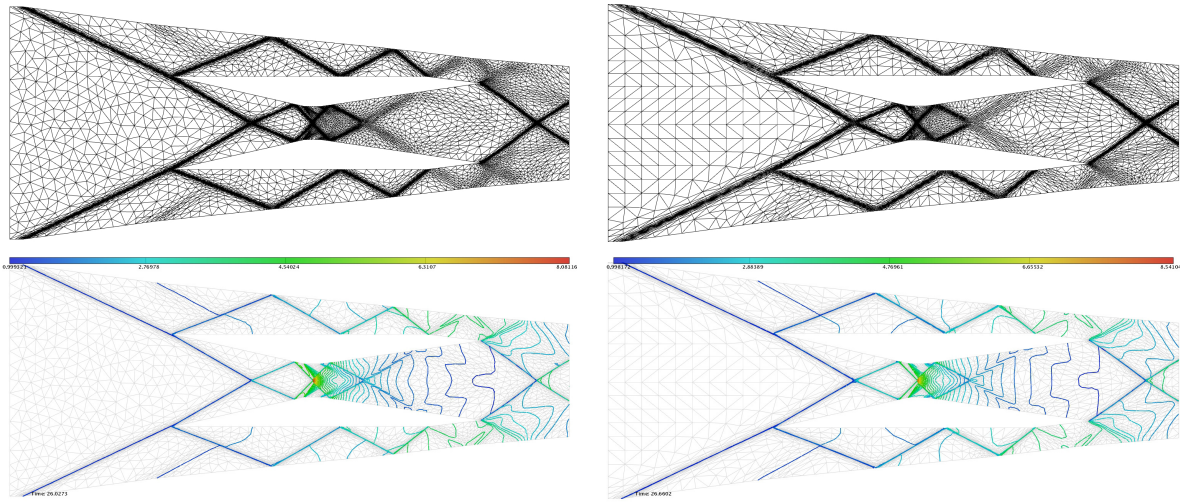

Fig. 13. *Supersonic scramjet simulation. Adapted meshes generated and solutions obtained with local remeshing (left) and metric-aligned advancing-point placement (right).*

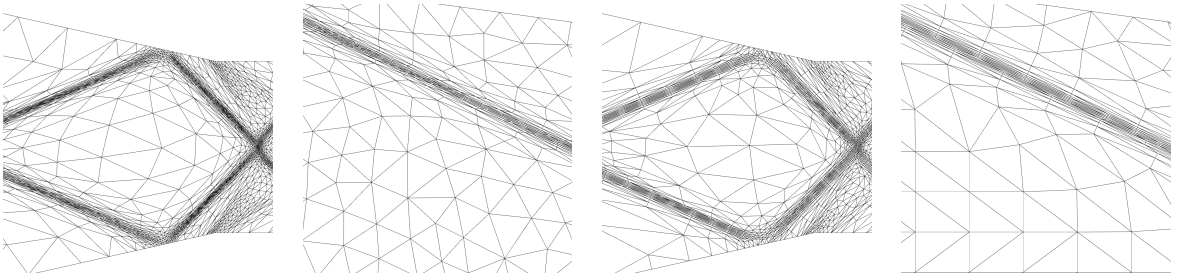

Fig. 14. *Supersonic scramjet simulation. Close-up view of the adapted meshes generated with local remeshing (far left and middle left) and metric-aligned advancing-point placement (middle right and far right).*

The 2D implementation presented is an initial version and subsequent work is needed to finalize the methodology for production usage. Additional work required includes improving mesh alignment and quality using smoothing during final mesh quality improvement. In addition, incorporating metric alignment into the quad element formation and smoothing algorithm is needed to improve quad mesh generation

The 2D work presented is fully extendable to 3D. An initial 3D implementation of metic-aligned point placement using a local remesher, Feflo.a, with a cavity-operator and a version of point placement that is similar in concept to advancing-point is presented in [14]. The easiest direct extension is to 3D surface meshing (required of any 3D process). The surface version of the current methodology is largely based on the 2D algorithm [19] and the primary changes needed are intersection of the 3D metric with the local surface geometry during point placement along with a variation of the advancing-point placement strategy to account for the 3D surface. For 3D volume mesh generation the metric-based approach has already been implemented without alignment using a non-aligned advancing-front

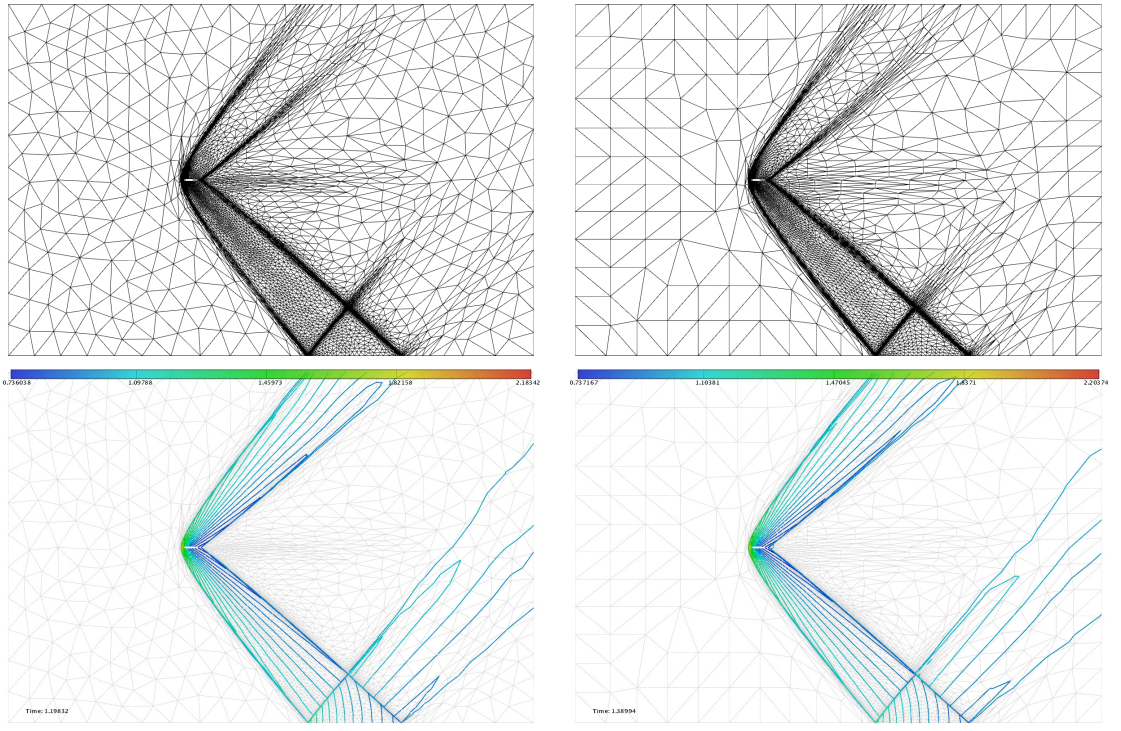

Fig. 15. *Supersonic NACA0012 simulation. Adapted meshes generated and solutions obtained with local remeshing (left) and metric-aligned advancing-point placement (right).*

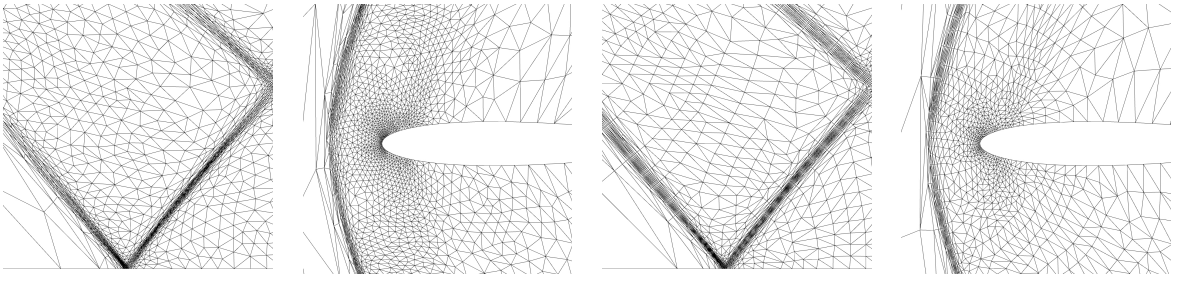

Fig. 16. *Supersonic NACA0012 simulation. Close-up view of the adapted meshes generated with local remeshing (far left and middle left) and metric-aligned advancing-point placement (middle right and far right).*

type point placement [20]. The aligned version of the advancing-point placement strategy as presented is directly extendable to 3D. Where considerable work will be required is any subsequent non-tetrahedral element formation as in hex-element generation. Non-trivial issues remain with generalized transition from hex to tetrahedral elements with transition via pyramids or split-quad-faces (solver dependent applicability). In addition, sliver elements can create bottlenecks to pseudo-structured elements in a generalized approach without considerable control. Established work and experience over several years with pseudo-structured boundary layer (BL) generation [19] provides confidence that these obstacles can be overcome. The potential benefits and payoff in 3D of an aligned metric-based anisotropic approach are significant and also would enable creation of a single unified approach to BL and field mesh generation.

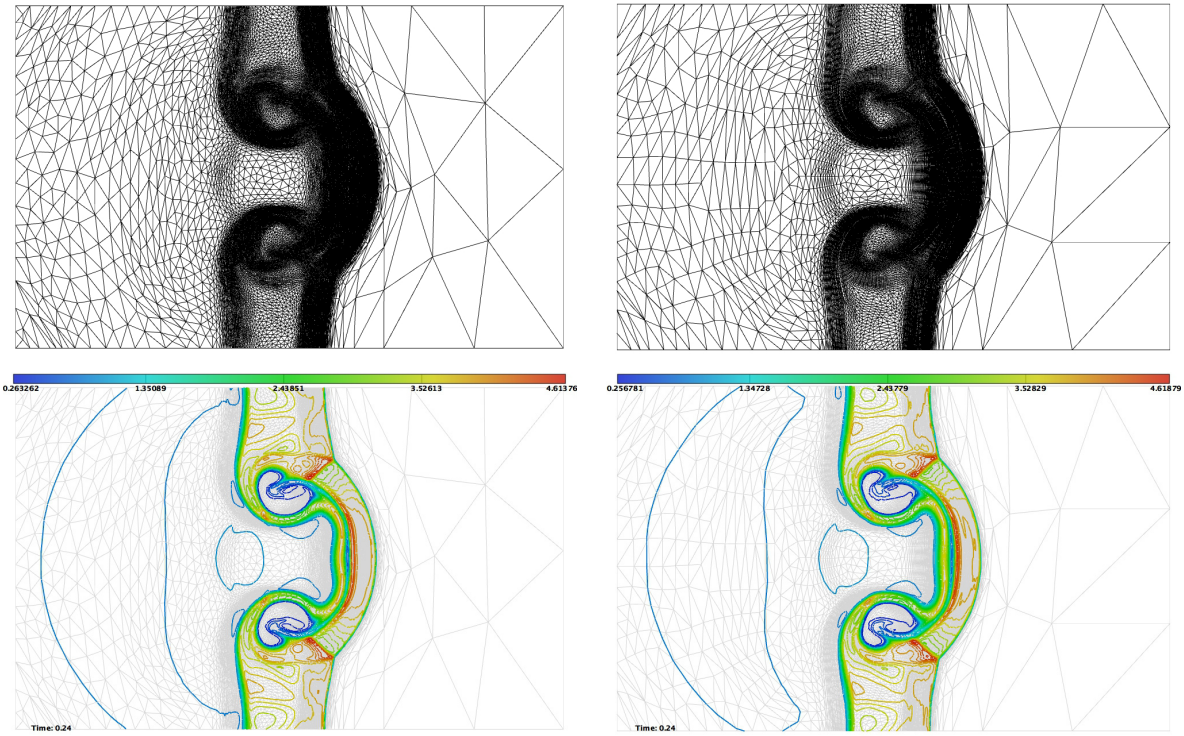


Fig. 17. *Shock-bubble interaction simulation. Adapted meshes generated and solutions obtained with local remeshing (left) and metric-aligned advancing-point placement (right).*

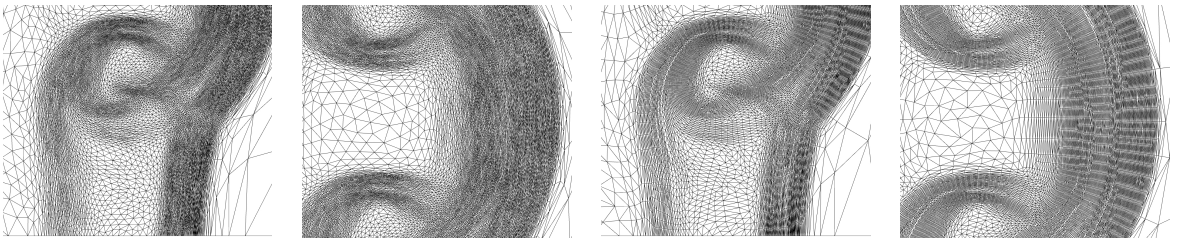

Fig. 18. *Shock-bubble interaction simulation. Close-up view of the adapted meshes generated with local remeshing (far left and middle left) and metric-aligned advancing-point placement (middle right and far right).*

## References


[1] F. Alauzet. Size gradation control of anisotropic meshes. *Finite Elem. Anal. Des.*, 46:181–202, 2010.

[2] F. Alauzet and P.J. Frey. Estimateur d'erreur géométrique et métrique anisotropes pour l'adaptation de maillage. Partie I : aspects théoriques. RR-4759, INRIA, March 2003. (in French).

[3] F. Alauzet, P.J. Frey, P.L. George, and B. Mohammadi. 3D transient fixed point mesh adaptation for time-dependent problems: Application to CFD simulations. *J. Comp. Phys.*, 222:592–623, 2007.

[4] F. Alauzet and A. Loseille. High order sonic boom modeling by adaptive methods. *J. Comp. Phys.*, 229:561–593, 2010.

[5] F. Alauzet and G. Olivier. Extension of metric-based anisotropic mesh adaptation to time-dependent problems involving moving geometries. In *49th AIAA Aerospace Sciences Meeting*, AIAA Paper 2011-0896, Orlando, FL, USA, Jan 2011.

[6] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Log-Euclidean metrics for fast and simple calculus on diffusion tensors. *Magn. Reson. Med.*, 56(2):411–421, 2006.

[7] C.L. Bottasso. Anisotropic mesh adaption by metric-driven optimization. *Int. J. Numer. Meth. Engng*, 60:597–639, 2004.

[8] C. Dobrzynski and P.J. Frey. Anisotropic Delaunay mesh adaptation for unsteady simulations. In *Proceedings of the 17th International Meshing Roundtable*, pages 177–194. Springer, 2008.

[9] P.J. Frey. About surface remeshing. In *Proceedings of the 9th International Meshing Roundtable*, pages 123–136, New Orleans, LO, USA, 2000.

[10] P.L. George, F. Hecht, and M.G. Vallet. Creation of internal points in Voronoi's type method. Control and adaptation. *Adv. Eng. Software*, 13(5-6):303–312, 1991.

[11] C. Gruau and T. Coupez. 3D tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric. *Comput. Methods Appl. Mech. Engrg.*, 194(48-49):4951–4976, 2005.

[12] W.T. Jones, E.J. Nielsen, and M.A. Park. Validation of 3D adjoint based error estimation and mesh adaptation for sonic boom reduction. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA-2006-1150, Reno, NV, USA, Jan 2006.

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

[14] A. Loseille. Metric-orthogonal anisotropic mesh generation. In *Proceedings of the 23th International Meshing Roundtable*. Elsevier, 2014.

[15] A. Loseille and F. Alauzet. Optimal 3D highly anisotropic mesh adaptation based on the continuous mesh framework. In *Proceedings of the 18th International Meshing Roundtable*, pages 575–594. Springer, 2009.

[16] A. Loseille and F. Alauzet. Continuous mesh framework. Part I: well-posed continuous interpolation error. *SIAM J. Numer. Anal.*, 49(1):38–60, 2011.

[17] A. Loseille, A. Dervieux, and F. Alauzet. Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations. *J. Comp. Phys.*, 229:2866–2897, 2010.

[18] A. Loseille and R. Löhner. Adaptive anisotropic simulations in aerodynamics. In *48th AIAA Aerospace Sciences Meeting*, AIAA Paper 2010-169, Orlando, FL, USA, Jan 2010.

[19] D.L. Marcum. Unstructured grid generation using automatic point insertion and local reconnection. In *The Handbook of Grid Generation*, Edited by J.F. Thompson, B. Soni, and N.P. Weatherill, chapter 18, pages 1–31. CRC Press, 1998.

[20] D.L. Marcum and F. Alauzet. Unstructured mesh generation using advancing layers and metric-based transition. In *21th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2013-2710, San Diego, CA, USA, Jun 2013.

[21] D.L. Marcum and N.P. Weatherill. Unstructured grid generation using iterative point insertion and local reconnection. *AIAA Journal*, 33(9):1619–1625, 1995.

[22] T. Michal and J. Krakos. Anisotropic mesh adaptation through edge primitive operations. *50th AIAA Aerospace Sciences Meeting*, Jan 2012.