A new 3D metric-aligned and metric-orthogonal method for anisotropic tetrahedral mesh generation is presented. The approach uses an advancing-point type point placement that aligns the elements with the underlying metric field and creates quasi-structured local topology. It is based on well-proven methodology and adds novel point-placement strategies that produce solution-adapted anisotropic meshes that are both sized and aligned with the solution gradients upon which the metric field is based. The overall methodology is extended to 3D from a previously developed 2D process. Results for analytical examples and a CFD simulation case are presented that demonstrate that aligned high-quality anisotropic tetrahedral meshes can be reliably generated.

1. Introduction

The benefits and fundamental concepts of metric-based mesh adaptation for dealing with anisotropic physical phenomena are well established [1]. Several successful examples [2–4] 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- 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 [5], Feflo.a [6], FUN3D [7], Gamic3d [8] and Omega_h [9].

It is worth mentioning that many of these codes include adaptive remeshers based on local 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 in three dimensions.

The approach used in this work is based on the existing AFLR advancing-front method with local-reconnection (edge/face-swapping) for mesh connectivity optimization [10–12], 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 is used in our approach, wherein the overall process is modified to satisfy a metric space definition. All geometric operations are performed in metric space with a consistent operational framework as defined in [1].

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 face that locally represents the front. In metric space the ideal point placement is chosen to form 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 in addition to the physical space geometry of the frontal face. We will refer to this approach as metric-aligned anisotropic mesh adaptation. Further improvements can be obtained if we consider local topology. Using right-angle type advancing-point point placement [10], quasi-structured element topology is produced locally that can be fully aligned with the solution gradients by the metric field. With right-angle type placement three-points are created from the face that locally represents the front. Alignment with the metric produces elements that are naturally ideal with respect to the metric field and also have quasi-structured topology. We will refer to this approach as metric-orthogonal anisotropic mesh adaptation.

These methodologies have been successfully implemented in the 2D version of AFLR [13]. The results obtained are of surprisingly good quality with respect to element shape as well as alignment with the metric field. Figs. 1 and 2 show the metric-orthogonal advancing-point point placement approach compared to classical local remeshing for a solution-adaptive simulation of a 2D scramjet flow field.

A first extension to three-dimensions has been done in Feflo.a within a different context and using a modified approach with a cavity-based local remeshing algorithm [14]. The extension of the metric-aligned and metric-orthogonal methods in three-dimensions for the advancing front method is natural as the point placement strategies follow the same algorithm as in two-dimensions. In three-dimensions, AFLR uses local-reconnection for connectivity optimization with a combined Delaunay and max angle related minimization type criterion, to minimize slivers (nearly degenerate elements) that are encountered in pure Delaunay-type methods. This optimization is especially important in the three-dimensional metric-orthogonal approach due to quasi-structured element formations. Moreover, based on the experience of the authors in boundary layer meshing [15–17] it is possible to extract mixed-element type meshes from a purely unstructured strategy when alignment is considered. This hopefully will eventually lead to a fully automatic mesh generation method capable of generating quasi-structured adapted meshes. The presented work is relative to the International Meshing Roundtable proceeding proposed by the authors [18].

2. Metric-based anisotropic mesh generation

We utilize the concept of metric-based mesh generation, initially introduced in [19], to generate a solution adaptive anisotropic mesh. The details, in the context of the present work, are presented in [13,14,17]. All geometrical operations are performed in either Euclidean metric-space (volume, angles, etc.) or a Riemannian metric-space (distance).

2.1. Euclidean metric space

For the sake of clarity, we recall the differential geometry notions that are used in the sequel. We use the following notations: bold face symbols, as \( \mathbf{a}, \mathbf{b}, \mathbf{u}, \mathbf{v}, \mathbf{x}, \ldots \), denote vectors or points of \( \mathbb{R}^3 \). The natural dot and cross product between two vectors \( \mathbf{u} \) and \( \mathbf{v} \) of \( \mathbb{R}^3 \) are denoted by: \( \mathbf{u} \cdot \mathbf{v} \) (dot product) and \( \mathbf{u} \times \mathbf{v} \) (cross product).

In these spaces, the length \( \ell_{\mathcal{M}} \) of vector \( \mathbf{ab} \) equal to \( b - a \) is given by the distance between its extremities:

\[
\ell_{\mathcal{M}}(\mathbf{a}, \mathbf{b}) = \sqrt{\langle \mathbf{a}, \mathcal{M} \mathbf{b} \rangle}.
\]

A Euclidean metric space \((\mathbb{R}^3, \mathcal{M})\) is a vector space of finite dimension where the dot product is defined by means of a symmetric definite positive CONSTANT tensor \( \mathcal{M} \):

\[
\mathbf{u} \cdot_{\mathcal{M}} \mathbf{v} = \langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{M}} = \langle \mathbf{u}, \mathcal{M} \mathbf{v} \rangle = \mathbf{u}^t \mathcal{M} \mathbf{v}, \text{ for } (\mathbf{u}, \mathbf{v}) \in \mathbb{R}^3 \times \mathbb{R}^3.
\]

In the following, the CONSTANT matrix \( \mathcal{M} \) is simply called a metric tensor or a metric. The dot product defined by \( \mathcal{M} \) makes \( \mathbb{R}^3 \) become a normed vector space \((\mathbb{R}^3, \| \cdot \|_{\mathcal{M}})\) and a metric vector space \((\mathbb{R}^3, d_{\mathcal{M}}(\cdot, \cdot))\) supplied by the following norm and distance definitions:

\[
\forall \mathbf{u} \in \mathbb{R}^3, \| \mathbf{u} \|_{\mathcal{M}} = \sqrt{\langle \mathbf{u}, \mathcal{M} \mathbf{u} \rangle};
\]

\[
\forall (\mathbf{a}, \mathbf{b}) \in \mathbb{R}^3 \times \mathbb{R}^3, d_{\mathcal{M}}(\mathbf{a}, \mathbf{b}) = \| \mathbf{ab} \|_{\mathcal{M}}.
\]
In these spaces, the length $\ell_M$ of vector $\mathbf{ab}$ equal to $b - a$ is given by the distance between its extremities:

$$\ell_M(\mathbf{ab}) = d_M(a, b) = \|\mathbf{ab}\|_M. \quad (2)$$

Note that this property is generally wrong for a general Riemannian metric space defined hereafter. As metric tensor $M$ is well-defined. These quantities are of main interest when dealing

with meshing. For instance using Relations (1) and (3), given a parallelepiped $K$ of $\mathbb{R}^3$, the volume of $K$ computed with respect to metric tensor $M$ is:

$$|K|_M = \mathbf{u}_M \cdot (\mathbf{v} \times M \mathbf{w}) = \sqrt{\det M} |K|_{I_3}, \quad (4)$$

where $|K|_{I_3}$ is the Euclidean volume of $K$. The angle between two non-zero vectors $\mathbf{u}$ and $\mathbf{v}$ is defined by the unique real-value $\theta_M \in [0, \pi]$ verifying:

$$\cos(\theta_M) = \frac{\langle \mathbf{u}, \mathbf{v} \rangle_M}{\|\mathbf{u}\|_M\|\mathbf{v}\|_M}. \quad (5)$$

In three dimensions, dihedral angles of a tetrahedron can be computed in a given Euclidean space from this definition as it is the angle between the faces normal. These relations are very useful in 3D for computing the area of faces with respect to $M$, let $F$ be a triangle in $\mathbb{R}^3$:

$$2 |F|_M = \|((A^\frac{1}{2} R) \mathbf{u} \times (A^\frac{1}{2} R) \mathbf{v})\| = \|\mathbf{u}\|_M\|\mathbf{v}\|_M \sin(\theta_M),$$

or

$$4 |F|_M^2 = \|((A^\frac{1}{2} R) \mathbf{u} \times (A^\frac{1}{2} R) \mathbf{v})\|^2 = \|\mathbf{u}\|_M^2\|\mathbf{v}\|_M^2 - (\mathbf{u} \cdot M \mathbf{v})^2.$$ 

**Geometric interpretation.** We will often refer to the geometric interpretation of a metric tensor. The set of points that are at distance 1 of a point $a$ is given by

$$B_M = \left\{ \mathbf{x} \mid \sqrt{\langle \mathbf{x} - a, M(\mathbf{x} - a) \rangle} = 1 \right\}.$$ 

It defines an ellipse in 2D or an ellipsoid in 3D centered at $a$ with its axes aligned with the eigen directions of $M$. Sizes along these directions are given by $h_i = \lambda_i^{-\frac{1}{2}}$. This ellipse is depicted in
Broadly speaking, a Riemannian metric space curves the parametrization space, i.e., the computational domain. We can extend the notions of a metric-based length and volume for Euclidean metric spaces to Riemannian metric spaces which will be the main operations performed by the mesh generator in such spaces. Fortunately, these notions can be easily derived in the context of meshing because we are not interested in evaluating these quantities on the Riemannian manifold. Indeed, as regards edge length computation, we do not want to compute the distance between two points which requires to find the shortest path on the curved manifold between these two points and to compute the length of the geodesic. We want to compute the length of the path between these two points defined by the straight line parametrization. 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 $M = (M(x))_{x \in \Omega}$, the length of edge $ab$ is computed using the line parametrization $y(t) = a + t ab$, where $t \in [0, 1]$:

$$\ell_M(ab) = \int_0^1 \|y'(t)\|_M dt = \int_0^1 \sqrt{ab M(a + t ab) ab} dt,$$

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

$$|K|_M = \int_K \sqrt{det M(x)} dx.$$  

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. These operations have a straightforward geometric interpretation when considering the ellipse associated with a metric. In this section, the practical uses of metrics inside the mesh generator is described.

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. For instance, the computation of the volume of an element in a Riemannian metric space given by Relation (7) can be done using quadrature formula which require the computation of some interpolated metrics inside the considered element. Fig. 5 illustrates metric interpolation along a segment, for which the initial data are the endpoints metrics.

Several interpolation schemes have been proposed in [20] which are based on the simultaneous reduction. The main drawback of these approaches is that the interpolation operation is not commutative. Hence, the result depends on the order in which the operations are performed when more than two metrics are involved. Moreover, such interpolation schemes do not satisfy useful properties such as the maximum principle. Consequently, to design an interpolation scheme on these objects, one needs a consistent operational framework. We consider the log-Euclidean framework introduced in [21].

We first define the notion of metric logarithm and exponential:

$$\text{Log}(M) := \text{log}(M(x))_{x \in \Omega} \quad \text{and} \quad \exp(M) := \text{exp}(M(x))_{x \in \Omega},$$

where $\text{log}(M) = \text{diag}(\text{log}(\lambda_i))$ and $\text{exp}(M) = \text{diag}(\exp(\lambda_i))$. We then define the logarithmic addition $\oplus$ and the logarithmic scalar multiplication $\odot$:

$$M_1 \oplus M_2 := \exp(\text{log}(M_1) + \text{log}(M_2))$$

and

$$\alpha \odot M := \exp(\alpha \cdot \text{log}(M)) = M^\alpha.$$
The logarithmic addition is commutative and coincides with matrix multiplication whenever the two tensors $\mathcal{M}_1$ and $\mathcal{M}_2$ commute in the matrix sense. 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 $(x_k)_{i=1..k}$ be a set of vertices and $(\mathcal{M}(x_k))_{i=1..k}$ their associated metrics. Then, for a point $x$ of the domain such that: $x = \sum_{i=1}^k \alpha_i x_i$ with $\sum_{i=1}^k \alpha_i = 1$, the interpolated metric is defined by:

$$\mathcal{M}(x) = \bigoplus_{i=1}^k \alpha_i \circ \mathcal{M}(x_i) = \exp \left( \sum_{i=1}^k \alpha_i \ln(\mathcal{M}(x_i)) \right)$$  \tag{8}$$

This interpolation is commutative. Moreover, it has been demonstrated in [21] that this interpolation preserves the maximum principle, i.e., for an edge $pq$ with endpoints metrics $\mathcal{M}(p)$ and $\mathcal{M}(q)$ such that $\det(\mathcal{M}(p)) < \det(\mathcal{M}(q))$ then we have $\det(\mathcal{M}(p + t pq)) < \det(\mathcal{M}(q))$ for all $t \in [0,1]$.

**Remark 2.1.** The interpolation formulation (8) reduces to $\mathcal{M}(x) = \prod_{i=1}^k \mathcal{M}(x_i)^{\alpha_i}$, if all the metrics commute. Therefore, an arithmetic mean in the log-Euclidean framework could be interpreted as a geometric mean in the space of metric tensors.

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

In this section, we describe how geometric quantities are computed numerically in Riemannian metric space as such quantities involve integrals. Let $\mathcal{M}$ be a discrete metric field defined at the vertices of a mesh $\mathcal{H}$ of a domain $\Omega$. Thanks to the interpolation operation, we have a continuous metric field in the whole domain, i.e., a Riemannian metric space $(\mathcal{M}(x))_{x \in \Omega}$. This representation of the metric field depends on $\mathcal{H}$ as the interpolation law is applied at the element level.

**Computation of edge length.** Approximation can be used to evaluate edge length in Riemannian metric space given by Relation (6). However, an analytical expression can be obtained if we consider that the metric field conforms to a geometric variation law as described in Section 2.3.1. Let $\mathbf{e} = \mathbf{p}_i \mathbf{p}_2$ be an edge of the mesh of Euclidian length $|\mathbf{e}|$, and $\mathcal{M}(\mathbf{p}_i)$ and $\mathcal{M}(\mathbf{p}_2)$ be the metrics at the edge extremities $\mathbf{p}_1$ and $\mathbf{p}_2$. We denote by $\ell_{\mathcal{M}}(\mathbf{e}) = \sqrt{\mathbf{e}^T \mathcal{M}(\mathbf{p}) \mathbf{e}}$ the length of the edge in metric $\mathcal{M}(\mathbf{p})$. We assume $\ell_{\mathcal{M}_1}(\mathbf{e}) > \ell_{\mathcal{M}_2}(\mathbf{e})$ and we set a $a = \ell_{\mathcal{M}_1}(\mathbf{e})$. The restriction of the (multi-dimensional) metric interpolator given by Relation (8) to an edge $\mathbf{e} = \mathbf{p}_1 \mathbf{p}_2$ leads to a geometric interpolation law on eigen values $\lambda$ and size $h$:

$$\lambda(t) = \exp \left( (1-t) \ln(\lambda_1) + t \ln(\lambda_2) \right) = \lambda_1^{1-t} \lambda_2^t$$

$$h(t) = h_1^{1-t} h_2^t.$$  

Under these assumptions, we deduce [22]:

$$\ell_{\mathcal{M}}(\mathbf{e}) = \ell_{\mathcal{M}_1}(\mathbf{e}) \frac{a - 1}{a \ln[a]}.$$  \tag{9}$$

We also recall that the prescribed size in direction $\mathbf{e}$ is

$$h_{\mathcal{M}}(\mathbf{e}) = \frac{\|\mathbf{e}\|}{\ell_{\mathcal{M}}(\mathbf{e})}.$$  

**Computation of element volume.** The evaluation of a tetrahedron volume in a Riemannian metric space consists in computing numerically Integral (7). Higher order approximation can be obtained by using Gaussian quadrature and metric interpolation based on the Log-Euclidean framework. For instance, if one considers a $k$-point Gaussian quadrature with weights $(\omega_j)_{j=1..k}$ and barycentric coefficients $(\beta_1^j, \beta_2^j, \beta_3^j, \beta_4^j)_{j=1..k}$, it yields:

$$|K|_{\mathcal{M}} \approx |K|_{\mathcal{M}_1} \sum_{j=1}^k \omega_j \det \left( \exp \left( \sum_{i=1}^4 \beta_i^j \log(\mathcal{M}_i) \right) \right).$$

However, for faster volume evaluation, only a first order approximation is considered in this work:

$$|K|_{\mathcal{M}} = \int_K \sqrt{\det \mathcal{M}} \, dx \approx \sqrt{\det \mathcal{M}_K} |K|_{\mathcal{M}_1}$$  \tag{10}$$

where $\mathcal{M}_K = \exp \left( \frac{1}{4} \sum_{i=1}^4 \log(\mathcal{M}_i) \right)$ and $p_i$ are the four vertices of tetrahedron $K$.

**Quality function: minimum edge weight function.** This quality function corresponds to the weight [23] for an element edge shared by two faces:

$$Q(K) = \min_{\mathbf{e}_i} \|\mathbf{e}_i\| = \max_{\mathbf{e}_i} n_{i1} \cdot n_{i2} / 6|K|,$$

where $\theta_i$ is the dihedral angle between two faces $F_{i1}$ and $F_{i2}$ sharing edge $\mathbf{e}_i$. It represents the geometric contribution to the value of the diagonal terms in a solution matrix for an elliptic equation. Consequently it represents a quantity that can degrade solver performance. It is directly analogous to minimizing the maximum angle in 2D and similar to that in 3D. In the present work it is used as a criterion for connectivity optimization with local-reconnection. This quality function can be written solely as a function of dot products, edge lengths and volume seeing that:

$$n_{i1} \cdot n_{i2} = (\mathbf{e}_i \times \mathbf{e}_i) \cdot (\mathbf{e}_k \times \mathbf{e}_k) = \mathbf{e}_i \cdot (\mathbf{e}_k \times \mathbf{e}_k) = (\mathbf{e}_i \cdot \mathbf{e}_k) \mathbf{e}_k - (\mathbf{e}_k \cdot \mathbf{e}_i) \mathbf{e}_i.$$  

To compute this quality function in Riemannian metric space, we choose a first order approximation assuming the metric is constant over element $K$ and given by $\mathcal{M}_K$:

$$Q_{\mathcal{M}}(K) = \max_{\mathbf{e}_i, K} \left( (\mathbf{e}_i \cdot \mathcal{M}_K \mathbf{e}_i)(\mathbf{e}_i \cdot \mathcal{M}_K \mathbf{e}_i) - \ell^2_{\mathcal{M}_K}(\mathbf{e}_i) \mathbf{e}_i \cdot \mathcal{M}_K \mathbf{e}_i \right).$$

The generalization to metric space for several similar quality functions is presented in [12].

### 2.4. Unit mesh

To generate anisotropic meshes, one must be able to prescribe at each point of the domain privileged sizes and orientations for the elements. This information will be transmitted to the mesher which will try to best fit these demands. The use of Riemannian metric space is an elegant and efficient way to achieve this goal. Indeed, it is possible for a mesher to work in such spaces as geometric quantities are well-posed. The main idea of metric-based mesh generation, initially introduced in [19], is to generate a unit mesh in a prescribed Riemannian metric space.

**Unit element.** A tetrahedron $K$, defined by its list of edges $(\mathbf{e}_i)_{i=1..6}$, is a unit element with respect to a metric tensor $\mathcal{M}$ if the length of all its edges is unit in metric $\mathcal{M}$:

$$\forall i = 1, \ldots, 6, \quad \ell_{\mathcal{M}}(\mathbf{e}_i) = 1$$

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

$$|K|_{\mathcal{M}} = \sqrt{\frac{7}{12}} \text{ and } |K|_{\mathcal{M}_1} = \sqrt{\frac{7}{12} (\det(\mathcal{M}))^{-\frac{1}{2}}}.$$  \tag{11}$$
Unit mesh. The notion of unit element 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.

For instance, size or shape of the computational domain can be incompatible with such a Riemannian metric space, and prevent an existence of a corresponding discrete mesh. Another simple counter example is the following. Let us consider the canonical Euclidean space $(\mathbb{E}^3(x))_{x \in K^3}$. Then, it is well known that $\mathbb{E}^3$ cannot be filled only with the regular tetrahedron. Consequently, even for the simplest Riemannian metric space $(\mathbb{E}^3(x))_{x \in K^3}$, there is no discrete mesh composed only of the unit regular tetrahedron. Therefore, the notion of unit mesh has to be relaxed.

A discrete mesh $\mathcal{H}$ of a domain $\Omega \subset \mathbb{E}^3$ is a unit mesh with respect to Riemannian metric space $(\lambda(x))_{x \in \Omega}$ 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. To avoid cycling while analyzing edges lengths, a tetrahedron $K$ defined by its list of edges $(e_i)_{i=1\ldots 6}$ is said to be quasi-unit if, $\forall i$, $\lambda(e_i) \in [\sqrt{2}, \sqrt{2}]$. The study in [1] shows that several non-regular space filling tetrahedra verify this constraint, which guarantees the existence for constant Riemannian metric space. Unfortunately, this weaker constraint on edges lengths can lead to the generation of quasi-unit elements with a null volume. Consequently, controlling only the edge lengths is not sufficient, the volume must also be controlled to relax the notion of unit element which is practically achieved by managing a quality function.

Generating adapted anisotropic meshes. Using the previous notions, the problem of mesh generation can be considerably simplified. Indeed, whatever the kind of desired mesh (uniform, adapted isotropic, adapted anisotropic), the mesh generator will always generate a unit mesh in the prescribed Riemannian metric space [19]. Consequently, the generated mesh is uniform and isotropic in the Riemannian metric space while it is adapted and anisotropic in the Euclidean space. This is illustrated in Fig. 6. This idea has turned out to be a huge breakthrough in the generation of anisotropic adapted meshes.

3. Metric-aligned and metric-orthogonal anisotropic mesh generation

3.1. Local remeshing quality issues

As stated in the introduction, local remeshing strategies [3,5-7,24] have been very successful in generating highly anisotropic adapted meshes mainly because they are 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 are simply rejected. Local remeshing algorithms typically use a process that analyzes the length of all the mesh’s edges: edges that are too long are split into two and edges that are too short are collapsed. Then, the mesh quality is optimized using connectivity optimization and vertex relocation. One can immediately see that the usual algorithm cannot reach exactly a size of 1 for all edges and it cannot 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.

Fig. 7 shows a fully unstructured anisotropic solution-adapted mesh obtained with a well-establish remeshing method. The

---

1 To give an example, if one takes for geometry a plate of height 0.1 and the size given by the metric field is 1 in that direction, we clearly see that there is an inconsistency between the size of the geometry and the size prescribed by the metric field.

3.2. Metric-based 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 as shown in Fig. 8. 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 criterion (minimize the maximum angle, maximize the edge weight, etc.). The overall procedure is applied repetitively until a complete field mesh is obtained. The basic steps in the procedure are outlined in Algorithm 1 and illustrated for advancing-point type placement in Fig. 9 (2D process shown for clarity). The overall steps and illustration of the 2D and 3D processes are equivalent. More complete details and results are presented in [10,11].

The AFRL algorithm can utilize a sizing function that can be defined by a simple isotropic length scale based on local edge lengths or an 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 (as presented in Section 2) such as dot products, cross products, reconnection criterion, etc. 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 $\ln(\lambda)$ 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 (as presented in Section 2). The background mesh and metric field can also be represented by an analytical function for testing purposes. However, without further modifications, this leads to a mesh generation method equivalent to a local remeshing approach. The resulting physical space quality is not improved and the method does not 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. 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.
3.3. Study of point placement strategy in metric spaces

The basics of the two different point insertion strategies are described in [10,11] and are illustrated in Fig. 8. For both placement types, the local front consists of one active face \( f \) (edge in 2D). The advancing-front type placement leads to generation of equilateral type elements. Each active face \( f \) (edge in 2D) proposes one point from the centroid of the face and in the orthogonal direction at a distance \( \ell = \sqrt{2} \ell(f) \). The advancing-point type placement leads to generation of right angle type elements. In this case, each active face \( f \) (edge in 2D) proposes three points (two points in 2D), one for each vertex. These points are issued from each face vertex and in the orthogonal direction at a distance \( \ell = \ell(f) \).

Now, we analyze the behavior of each strategy in the context of metric-based mesh generation depending whether the orientation of the front (which is the active face) is or is not aligned with the metric direction, i.e., the metric eigenvector. In the case of active faces not aligned with the metric direction, we notice that for either method if the point placement is not aligned with the metric eigenvectors, i.e., it is still done orthogonally to the active face, then there is a loss of anisotropy and possible creation.
of isotropic elements in physical space while the element is unit in metric space, see Fig. 10(a) and (b). This adverse effect is also illustrated in Fig. 11(a) and (b) where adapted meshes have been generated without metric-alignment. We clearly see the creation of isotropic elements. However, if the metric eigenvector direction is used to propose the point then anisotropy is preserved, see Fig. 10(c) and (d). This is clearly illustrated on the example of Fig. 11(c) and (d).

In the case of active faces 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 Figs. 12(b) and 11(c). However, more optimal angles are obtained if the smallest triangle side proposes a point, see Fig. 12(a). With the advancing-point placement and metric alignment, optimal angles are generally produced and right angle shape elements are still preserved and quads can be produced, as shown in Figs. 12(c) and (d), and 11(d).

From the previous considerations, the metric-orthogonal method uses the advancing-point type placement to propose new points, i.e., for each active face, each vertex of the face proposes a new point. For alignment, the point placement uses both the metric orientation and the face normal direction. If the anisotropic ratio for the metric is sufficient then the metric is used solely to align the point placement and if it is close to isotropic then the geometric normal is used. In between the two directions are averaged. This is crucial, because it provides an algorithm which is consistent with the classical (non-adapted) AFLR method when an isotropic metric field is provided. To find the optimal position of the new point in the obtained direction, an iterative strategy is used wherein the proposed point metric is re-evaluated iteratively until convergence. This is mandatory to generate edge of length one in the Riemannian metric space because we face a non-linear problem. Each time, metric at new point is evaluated by interpolation on the background mesh. The metric-orthogonal point placement strategy is described in Algorithm 2 where \( r_{\text{min}} \) is the anisotropic ratio below which the metric is considered isotropic (then, the face normal is used for alignment) and \( r_{\text{max}} \) is the anisotropic ratio above which the metric is considered anisotropic (then, the metric eigenvector is used for alignment).
The **metric-aligned method** uses exactly the same approach except that the face proposes only one new point from its barycentre and the considered metric is the face average metric (the average of its three vertices metric).

**Metric-orthogonal method and unit mesh.** The metric-orthogonal approach tends to generate right-angle tetrahedron and not equilateral tetrahedron in metric space. Therefore, the optimal tetrahedron has three edges of length 1 and three edges of length $\sqrt{2}$. Therefore, to obtain a converging process we have to relax the unit mesh length bound that has been given in Section 2.4. When, the metric-orthogonal method is considered the unit edge length bound is set to $[\sqrt{2}/2, 2]$.

### 4. Numerical examples

3D examples are presented to illustrate the effectiveness of the proposed metric-orthogonal point placement strategy for generating high-quality aligned anisotropic adapted meshes. For the presented results, the adapted meshes shown are obtained directly from the AFLR metric-orthogonal method without final quality improvement smoothing. This better illustrates the effectiveness of the point placement. Further improvement is expected with a smoothing process that is appropriately modified for the present approach. In addition, no process for preserving local pseudo-structured topology is implemented. Results in 2D have shown that including preservation of local pseudo-structured topology, as the triangles are generated, enhances the local structure and increases the number of triangles that can be combined into high-quality quadrilaterals. This process is directly extendable to 3D for mixed element type meshes. Also, the pseudo-structured topology with high-aspect-ratio elements will typically produce some difficult to remove sliver elements. There are a few approaches immediately available to address this issue, including direct sliver removal with element combination and improved local reconnection. When element combination is used to form prism and hex elements, the sliver elements can be directly removed. Those element types can then be re-formed into tetrahedral elements with proper ordering, if a tetrahedral only mesh is desired. While, these quality improvements are essential for production type usage, they are not for the present effort, which is focused on metric-orthogonal point placement. For the results presented in this section, additional CPU usage for mesh generation with metric alignment is minimal and of the same order as that for the metric-based generator without alignment. In general the base AFLR approach generates hundreds of thousands of tetrahedral elements in order seconds and up to tens of millions of elements in order minutes on typical laptop/desktop computers. With the metric-based approach added with alignment the mesh generation time increases by a factor of two to three. For each of the cases shown in this section the CPU time required mesh generation is under a minute.

#### 4.1. Non-adapted mesh

A non-adapted mesh with advancing-point type point placement was generated on a notional two booster launch vehicle configuration. Results are shown in Fig. 13. As shown the advancing point type point placement produces a local element structure that is predominantly quasi-structured. Dihedral angle distribution is shown in Fig. 14. As shown, the majority of elements are in the 40–100 degree range for the isotropic quasi-structured elements (an ideal isotropic and uniform quasi-structured mesh would have peaks at 45 and 90 degrees). A significant number of hexahedral elements could be obtained from the advancing-point mesh for this case given the local structure and that the surface mesh contains primarily quadrilateral faces.

#### 4.2. Adaptation to analytical metric functions

In these examples, the metric is defined analytically, meaning that each time a new point is inserted its exact metric value is evaluated. All analytical metric functions are applied to a cubic domain $[-0.5, 0.5] \times [-0.5, 0.5] \times [-0.5, 0.5]$. The metric-orthogonal method is considered.
Fig. 13. Two-booster launch vehicle. Multiple views of the mesh generated with advancing-point placement showing significant quasi-structured alignment.

Fig. 14. Two-booster launch vehicle. Dihedral angle distributions.
Cross metric function. The first metric analytical function considered represents straight anisotropic features that intersect and form a 3D cross. The direction of these features is aligned with the unit square boundary geometry. The function is

\[ M(x, y, z) = \begin{pmatrix} h_x^{-2} & 0 & 0 \\ 0 & h_y^{-2} & 0 \\ 0 & 0 & h_z^{-2} \end{pmatrix} \]

where

\[
\begin{aligned}
& h_x = \min(2^{\alpha_x} \times h_{\min}, h_{\max}) \\
& h_y = \min(2^{\alpha_y} \times h_{\min}, h_{\max}) \\
& h_z = \min(2^{\alpha_z} \times h_{\min}, h_{\max})
\end{aligned}
\]

with \( h_{\min} = 0.003125 \), \( h_{\max} = 0.0625 \), \( \alpha_x = 40 \times |x| \), \( \alpha_y = 40 \times |y| \), and \( \alpha_z = 40 \times |z| \). In this case the challenge is to maintain significant alignment and quasi-structured topology in all directions. The adapted surface mesh used contains unstructured triangular faces. As shown in Fig. 15 the resulting mesh does contain predominantly highly aligned quasi-structured elements throughout the domain.

Quarter cylinder metric function. The metric function represents a curved anisotropic feature having the shape of a cylinder. The analytical function reads:

\[
M(x, y, z) = \begin{pmatrix} h_1^{-2} \cos^2 \theta + h_2^{-2} \sin^2 \theta & (h_1^{-2} - h_2^{-2}) \cos \theta \sin \theta & 0 \\ (h_1^{-2} - h_2^{-2}) \cos \theta \sin \theta & h_1^{-2} \sin^2 \theta + h_2^{-2} \cos^2 \theta & 0 \\ 0 & 0 & h_2^{-2} \end{pmatrix}
\]

where \( h_1 = \min(0.002 \times 5^x, h_{\max}) \), \( h_2 = \min(0.05 \times 2^x, h_{\max}) \), \( \theta = \arctan(x, y) \), \( h_{\max} = 0.1 \), \( \alpha = 10 \times |0.75 - \sqrt{x^2 + y^2}| \).

The main difficulty of this analytical 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 Figs. 16 and 17. We notice, as expected, that the metric-orthogonal method preserves the right-angle tetrahedra shape for the elements. Dihedral angle distributions are shown in Fig. 18 (blue bars) for the metric-orthogonal results. There is a significant peak in the 80–90 degree range from quasi-structured elements and smaller peaks at the lower angles from anisotropic elements. Also shown for comparison in Fig. 18 (orange bars) are results for the metric-aligned method. We see a larger number of elements at the minimum and maximum extremes (this is also typical of local remeshing). These are reduced using the metric-orthogonal method.

4.3. Supersonic missile

CFD simulations were run for a notional missile configuration to evaluate results for the new metric-orthogonal method in an actual solution adaptation process. The missile configuration is shown in Fig. 19. In this case we used a non-adapted surface and volume mesh as the starting point composed of 51262 vertices.

Algorithm 1 AFRL Advancing Front Overall Process

1. Generate a valid initial triangulation of the boundary points only and recover all boundary faces.
2. Assign a metric to each boundary point from interpolation on the background metric field using Equation (8).
3. Set each element as active (satisfied) or inactive (unsatisfied) based on whether the element sizing satisfies the sizing specified by the metric field.
4. Determine element faces for advancement by inspection of element face neighbors. If a given element is active and has one inactive neighbor and three other active neighbors then the face neighboring the inactive neighbor is considered for advancement.
5. Create the list of points to be inserted depending on the chosen method:
   - for traditional equilateral type elements method, generate points using advancing-front type point placement as shown in Fig. 8 (left). Each active element face proposes to advance one point from the face barycentre.
   - for right angle type elements method, generate points using advancing-point type point placement as shown in Fig. 8 (right). Each active element face proposes to advance multiple points from the face vertices.
6. Set a metric to new points from interpolation on the background metric field using Equation (8).
7. Reject or average new points that are too close to other new points. Averaging only applies in the case of advancing-point type placement and then only for those new points that come from the same existing mesh point.
8. 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.
9. Optimize the connectivity using local-reconnection. For each element pair (or set of elements around an edge), 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.
10. Repeat the point generation and local-reconnection process, described above, until no new points are generated.
11. Smooth the coordinates.
12. Optimize the connectivity using the local-reconnection process.

Algorithm 2 Metric-Orthogonal Process

1. Find the eigenvector \( v_\alpha \) of \( M_\alpha \) that is most aligned to the face normal \( n \)
2. Compute point \( q \) that is at length 1 from point \( p \) in metric \( M_\alpha \) direction \( v_\alpha \):
   \[
   q = p + \alpha v_\alpha \quad \text{such that } \ell_{M_\alpha(pq)} = 1.
   \]
3. Find the eigenvector \( v_\beta \) of \( M_\beta \) that is most aligned to the face normal \( n \)
4. Average both eigenvectors: \( v_{pq} = \frac{1}{2} (v_\alpha + v_\beta) \)
5. Compute metric \( M_\beta \) anisotropic ratio: \( r = \sqrt{\frac{\max(\beta)}{\min(\beta)}} \)
6. Get optimal direction: \( v_{opt} = \beta(r) n + (1 - \beta(r)) v_{pq} \) with \( \beta(r) = \max(0, \min(\frac{r-r_{\min}}{r_{\max}-r_{\min}}, 1)) \)
7. Compute the optimal point \( \tilde{q} \) that is at length 1 from point \( p \) in Riemannian metric space \( M \) in direction \( v_{opt} \) using a Newton’s algorithm:
   \[
   \tilde{q} = p + \alpha v_{opt} \quad \text{such that } \ell_{M(pq)} = 1.
   \]
and 277 184. A metric field was obtained from a corresponding CFD simulation in a typical coupled adapted mesh/solution manner [25]. The flow field represents supersonic flow over a notional missile at Mach 1.8. This simulation is representative of compressible problems involving highly anisotropic phenomena with strong shocks.

**Flow solver.** The flow is modeled by the compressible Euler equations. We used the flow solver Wolf [3, 26], which is a finite element/finite volume flow solver for unstructured meshes composed of tetrahedra. Wolf is vertex-centered and achieves second order accuracy in space using a low dissipation MUSCL technique. Temporal integration is implicit and uses an SGS Newton’s method coupled with local time stepping.

Feature-based anisotropic mesh adaptation. Multi-scale anisotropic mesh adaptation is used during the simulation to control the interpolation error of the local Mach number flow variable in the $L^2$-norm [25, 27]. In this case, the optimal metric field at each point $\mathbf{x}$ of domain $\Omega$ is given by relation:

$$M_{L^2}(\mathbf{x}) = N^2 \left( \int_\Omega \det(|H_u(\bar{x})|) \frac{1}{2} \right)^{-\frac{2}{3}} \det(|H_u(\mathbf{x})|)^{-\frac{1}{3}} |H_u(\mathbf{x})|,$$

where $u$ is the chosen sensor (here the local Mach number), $|H_u|$ is the absolute value of the Hessian of the sensor, and $N = c(M_{L^2})$ is the metric complexity prescribed by the user representing the size of the mesh.

There is a direct relationship between the prescribed metric complexity and the number of elements of the generated

---

**Fig. 15.** Cross analytical metric function. Adapted mesh obtained with the metric-orthogonal method: overall view (left) and zoom view (right).

**Fig. 16.** Quarter cylinder analytical metric function. Adapted mesh obtained with the metric-orthogonal method: overall view (left) and zoom view (right) for a constant y-plane cut (slightly rotated for clarity).
mesh. For a given metric field $(M(x))_{x \in \Omega}$, the metric complexity reads $[25,27]$:

$$C(M) = \int_\Omega \sqrt{\det M(x)} \, dx.$$ 

If a unit mesh $\mathcal{H} = \bigcup K_i$ (where the $K_i$ are the tetrahedra of mesh $\mathcal{H}$) is generated with respect to $(M(x))_{x \in \Omega}$, thanks to Relations (10) and (11), the metric complexity can be approximated by:

$$C(M) \approx \sum_K \sqrt{\det M_K |K|} \approx \sum_K \frac{\sqrt{2}}{12} = \frac{\sqrt{2}}{12} \times nt,$$

where $|K|$ is the volume of $K$, $M_K$ is the average metric at element $K$, and $nt$ is the number of elements of the mesh. This relationship is true in the metric-aligned context where unit equilateral tetrahedra are generated in metric space.

In the case of the metric-orthogonal approach, the volume of the unit element – which is now the unit orthogonal tetrahedron – is no more $\frac{\sqrt{2}}{12}$ but $\frac{1}{6}$. Thus, when the metric-orthogonal approach is used we have the following relation:

$$C(M) \approx \frac{1}{6} \times nt,$$
Therefore, if a given metric complexity $N$ is prescribed by the user, we will then get a mesh with approximately $\sqrt{2}$ less elements using the metric-orthogonal method than using the metric-aligned one. The above relations and the ratio may vary a bit depending of the considered problem and geometry.

**Results analysis.** For each case, adapted meshes are shown after 5 adaptation iterations. Results obtained with an established local remeshing approach without alignment are shown in Fig. 20 (left). In that case, a metric complexity of 60 000 has been prescribed leading to a mesh composed of 93 147 vertices and 532 276 tetrahedra.

Result for the present AFLR method with the metric-orthogonal approach is shown for comparison in Fig. 20 (right). The prescribed metric complexity was 85 000 and the resulting mesh contained 103 170 vertices and 585 988 tetrahedra. Detail views on the mesh are shown in Fig. 21 and the obtained flow solution (density field) is given in Fig. 22. We observe that the new metric-orthogonal method ends up with a higher quality (in terms of alignment and maximum angle) anisotropic adapted mesh in comparison to the same method with advancing-front point placement or to classical remeshing. A local quasi-structured mesh is also obtained. Very good alignment of the elements is obtained even for this case with a complex widely varying metric field obtained from a numerical CFD solution. The elements are perfectly aligned with the Mach cone of the supersonic flows (Fig. 21), and we observe in the back plane view that the pseudo-structured elements are able to follow the curvature of the solution. To further quantify the mesh quality, dihedral angle distributions are shown in Fig. 23 for the adapted missile case using both metric-aligned (advancing-front) and metric-orthogonal (advancing-point) methods. The distributions are similar in characteristics to that of the analytical quarter cylinder example with additional spreading due to the geometry and the flow field complexity.

The flow field solutions obtained with the metric-orthogonal method, shown in Fig. 22, are visually identical and quantitatively equivalent with the same estimated accuracy to the results obtained with the local remeshing approach. Previous results [13] for multiple 2D solution-adapted cases also demonstrated that the metric-orthogonal method required fewer elements than the local-remeshing approach to reach the same level of accuracy. While there is no reason to expect differing results in 3D, additional comparisons are needed to truly demonstrate this.

We conjecture that anisotropic solution-adapted meshes with optimal quality in terms of alignment and maximum angle (in both metric and physical space) offer improvements for the overall CFD simulation process. We base this on observations that include: large element angles in physical space can produce stiffness in the solution matrix and degrade solver performance, fully- or pseudo-structured mesh alignment with strong bow-shock waves can improve solution accuracy, and fully- or pseudo-structured mesh alignment in boundary-layer regions is typically required for turbulent flow simulations. Multiple CFD simulation cases must be considered in detail with differing solvers and alternative solution-adaptive methods in order to quantify the improvements enabled by using an aligned approach.

### 5. Conclusion

A new metric-orthogonal (resp. metric-aligned) based on the advancing-point (resp. advancing-front) type point placement strategy has been implemented in a metric-based anisotropic
The new approaches use an advancing-point or an advancing-front type point placement that aligns the elements with the underlying metric field. It is based on well-proven methodology and adds novel point-placement strategies that produce solution-adapted anisotropic meshes that are both sized and aligned with the solution gradients upon which the metric field is based. The overall methodology was implemented in a 3D process that was extended from 2D. Results were presented for multiple examples, including analytical cases and a CFD simulation. These results demonstrate that aligned high-quality anisotropic meshes can be reliably generated. Additional work is required to quantify the level of improvement that alignment offers for the overall CFD simulation process.

The strategy presented has promise for generation of non-tetrahedral elements as in prism or hex-element generation. Considerable work is required to achieve this, and non-trivial issues remain with generalized transition from hex to tetrahedral elements with transition via pyramids or split-quad-faces (solver dependent applicability). Established work and experience over several years with quasi-structured boundary layer (BL) generation [10] provides confidence that these obstacles can be overcome in time. The potential benefits and payoff of an aligned metric-based anisotropic multi-element type approach are significant and also would also provide a viable single unified approach to BL and field mesh generation.

Acknowledgments

The first author would like to acknowledge that this work is supported, in part, by the Engineer Research & Development Center, USA under Cooperative Agreement number W912HZ-15-2-0004 and NASA, USA grant no. NNX15AU39A. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Engineer Research & Development Center, NASA, or the U.S. Government.
Fig. 23. Dihedral angle distributions for the supersonic missile simulation using advancing-point and advancing-front.

References