

# VISUALIZING MESH ADAPTATION METRIC TENSORS

Ko-Foa Tchou   Julien Dompierre   Marie-Gabrielle Vallet   Ricardo Camarero

*École Polytechnique de Montréal*

*C.P. 6079, Succ. Centre-ville, Montréal (QC) H3C 3A7, Canada.*

*[ko-foa.tchon|julien.dompierre|marie-gabrielle.vallet|ricardo.camarero]@polymtl.ca*

## ABSTRACT

Riemannian metric tensors are used to control the adaptation of meshes for finite element and finite volume computations. To study the numerous metric construction and manipulation techniques, a new method has been developed to visualize two-dimensional metrics without interference from any adaptation algorithm. This method traces a network of orthogonal tensor lines to form a pseudo-mesh visually close to a perfectly adapted mesh but without many of its constraints. Although the treatment of isotropic metrics could be improved, both analytical and solution-based metrics show the effectiveness and usefulness of the present method. Possible applications to adaptive quadrilateral and hexahedral mesh generation are also discussed.

**Keywords:** tensor visualization, mesh adaptation, Riemannian metric, tensor line, hyperstreamline.

## 1. INTRODUCTION

Symmetric second-order tensor data frequently arises from medical and engineering applications. Classical examples are diffusion tensors from Magnetic Resonance Imaging (MRI) and stress tensors from solid mechanics. Lately, Riemannian metric tensors have also been used to control mesh adaptation for finite element and finite volume computations [1–3]. Numerous methods have been developed to construct and manipulate those metrics. For example, the Hessian of a computed field can be used to construct a metric tensor for solution-adaptive remeshing. When no solution is yet available, metrics based on the computational domain geometry can be used instead [4]. User specifications can also be formulated as metric tensors and combined with solution-based and geometric metrics. The resulting tensors may, however, prescribe abrupt size variations that a proper conformal mesh cannot possibly reproduce. Post-processing methods have thus been proposed to smooth such metrics and improve mesh gradation [5, 6]. Many variations exist on these metric construction and manipulation methods. There are indeed several alternatives to compute the solution derivatives, particularly at domain boundaries, and form the Hessian matrix. Similarly, the geometric features of the domain, such as its local thickness and curvature, may be combined differently than in [4] to obtain a geometric metric. The interpolation itself of a discrete metric can also vary to favor bigger or smaller elements for example [7]. Metric visualization would be an invaluable tool to study the impact of these different alternatives. Although a perfectly adapted mesh is indeed an indirect visualization of the target metric, it is biased by the adaptation algorithm. Furthermore, such an *a posteriori* approach cannot be used to evaluate beforehand the feasibility of a given metric, i.e., whether it is theoretically possible or not to generate a proper mesh perfectly

adapted to this metric. A more direct visualization method is thus needed.

Compared to scalars and vectors, tensor fields are still challenging to visualize. Tensors are matrix valued functions and their individual components can be visualized separately as scalars. However, it is difficult to gain insight on the structure of the field from multiple scalar plots. Furthermore, the matrix components are strongly dependent on the orientation of the reference coordinate system. A better decomposition is based on the tensor's eigensystem. For example, iconic methods plot, at discrete locations, elliptical or ellipsoidal glyphs reflecting the local magnitude of the eigenvalues as well as the orientation of the corresponding eigenvectors. Such a discontinuous information is, however, difficult to interpolate visually in order to assess the global structure of the tensor field. An alternative is to use tensor lines [8] or hyperstreamlines [9], i.e., streamline equivalents but tangent to the tensor's eigenvector fields. To avoid cluttering the domain, a small and carefully chosen set of tensor lines originating from special degenerate points can be used to extract a topological skeleton of the tensor field [10]. No single method, however, has yet covered all the aspects of the complex nature of tensor fields and new methods appear regularly. Some simulate the deformation of a continuous medium under stress [11], others use direct volume rendering techniques [12] for example. Choosing the best one is context dependent.

For metrics, a mesh-like approach is probably the most intuitive. That is why the proposed method saturates the domain with tensor lines to mimic a perfectly adapted mesh but without many of its constraints like continuity and conformity. Such a *pseudo-mesh* is not biased by any adaptation algorithm and can be constructed even if a proper mesh cannot. The tensor line placement technique is very close to the one

used by Alliez et al. [13] for their polygonal surface remeshing algorithm. However, instead of a surface curvature tensor, an adaptation metric tensor is considered. Furthermore, tensor lines are spaced a unit metric distance apart like the vertices in an adapted mesh.

After explaining how a metric tensor is used to adapt meshes, the present paper describes the construction of a pseudo-mesh to visualize such a metric. Analytical and solution-based metrics illustrate the effectiveness and usefulness of the method. Only two-dimensional metrics are considered in the present study but future developments could include a three-dimensional extension as well as the generation of proper meshes from pseudo-meshes.

## 2. MESH ADAPTATION CONTROL METRICS

Accuracy of finite element and finite volume methods is strongly dependent on the quality of the domain discretization and, more precisely, its mesh. Control of the size, stretching and orientation of the mesh elements is thus crucial and can be done through mesh adaptation [1–3]. To decouple the actual adaptation algorithm from the target mesh specifications, the process can be controlled using the metric of the transformation that maps a perfect mesh element into a unit square for quadrilateral meshes or a unit equilateral triangle for simplicial ones.

### 2.1. Definition

In two dimensions, such a Riemannian metric tensor is defined at every point of the domain by a  $2 \times 2$  symmetric positive-definite matrix  $\mathcal{M}$ . This matrix can be factored as the product of a rotation matrix  $\mathcal{R}$  and a diagonal scaling matrix  $\Lambda$ :

$$\begin{aligned} \mathcal{M} &= \mathcal{R}\Lambda\mathcal{R}^{-1} \\ &= \begin{pmatrix} \vec{e}_1 & \vec{e}_2 \end{pmatrix} \begin{pmatrix} h_1^{-2} & 0 \\ 0 & h_2^{-2} \end{pmatrix} \begin{pmatrix} \vec{e}_1^T \\ \vec{e}_2^T \end{pmatrix} \end{aligned} \quad (1)$$

The columns of  $\mathcal{R}$  are the eigenvectors of  $\mathcal{M}$  and correspond to two prescribed directions  $\vec{e}_1$  and  $\vec{e}_2$ . Since  $\mathcal{R}$  is orthogonal, its inverse  $\mathcal{R}^{-1}$  is equal to the transposed matrix  $\mathcal{R}^T$ . The diagonal terms of  $\Lambda$  are the eigenvalues of  $\mathcal{M}$  and correspond to the inverse of the squared target sizes  $h_1$  and  $h_2$  along the prescribed directions  $\vec{e}_1$  and  $\vec{e}_2$ . This metric can be interpreted as the transformation that maps an ellipse to a unit radius circle (Fig. 1). The axes of this ellipse are given by the eigenvectors of the matrix  $\mathcal{M}$  and its eigenvalues are reflected in the width and height of the ellipse.

Mesh adaptation algorithms perform local or global opera-

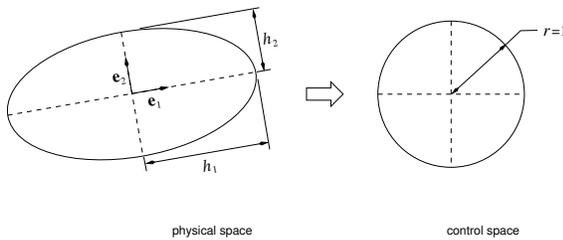


Figure 1: Geometric interpretation of a Riemannian metric.

tions to enforce the target size, stretching and orientation prescribed by the control metric. An important parameter used by those algorithms is the metric length between point  $A$  and point  $B$

$$l_{AB}^{\mathcal{M}} = \int_0^1 \sqrt{(\vec{p}_B - \vec{p}_A)^T \mathcal{M}(\vec{p}_t) (\vec{p}_B - \vec{p}_A)} dt \quad (2)$$

where  $\vec{p}$  denotes a position vector and  $\vec{p}_t = \vec{p}_A + t(\vec{p}_B - \vec{p}_A)$ . It has been shown that the adaptation process is equivalent to requiring all the mesh edges to have a unit metric length [2]. That is why perfectly adapted meshes are said to be unit meshes.

### 2.2. Construction

To concentrate elements in critical regions, such control metrics can come from many sources. They can be given analytically or deduced from the geometric properties of the domain to mesh [4] for example, but are usually constructed from *a posteriori* error analysis. The approximation error between an exact solution  $u$  and a computed finite element solution  $u_h$  is difficult to estimate in general but, according to Céa's lemma, it is bounded by the interpolation error for elliptic problems [14]. Practically, this relation holds for a large class of problems and the interpolation error is commonly used as an error estimator for adaptive mesh generation. If the solution in an  $n$ -dimensional space is considered as an hypersurface of dimension  $n + 1$ , such an error can be geometrically interpreted as the gap between the surface and its piecewise linear interpolation [7]. The local mesh density necessary to achieve a prescribed error level is thus related to the curvature of this surface and, therefore, the Hessian of the solution, i.e., its second order derivatives

$$\mathcal{H} = \begin{pmatrix} \partial^2 u_h / \partial x^2 & \partial^2 u_h / \partial x \partial y \\ \partial^2 u_h / \partial y \partial x & \partial^2 u_h / \partial y^2 \end{pmatrix} \quad (3)$$

Since  $\partial^2 u_h / \partial x \partial y = \partial^2 u_h / \partial y \partial x$ , this matrix is symmetric and can be decomposed as

$$\begin{aligned} \mathcal{H} &= \mathcal{R}\Lambda\mathcal{R}^{-1} \\ &= \begin{pmatrix} \vec{e}_1 & \vec{e}_2 \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} \vec{e}_1^T \\ \vec{e}_2^T \end{pmatrix} \end{aligned} \quad (4)$$

where  $\mathcal{R}$  is the Hessian's eigenvector matrix and  $\Lambda$  is its diagonal eigenvalue matrix. The corresponding adaptation metric is

$$\begin{aligned} \mathcal{M} &= \mathcal{R}\tilde{\Lambda}\mathcal{R}^{-1} \\ &= \begin{pmatrix} \vec{e}_1 & \vec{e}_2 \end{pmatrix} \begin{pmatrix} \tilde{\lambda}_1 & 0 \\ 0 & \tilde{\lambda}_2 \end{pmatrix} \begin{pmatrix} \vec{e}_1^T \\ \vec{e}_2^T \end{pmatrix} \end{aligned} \quad (5)$$

where  $\tilde{\lambda}_i = \min(\max(C|\lambda_i|, h_{\max}^{-2}), h_{\min}^{-2})$  and the target size along  $\vec{e}_i$  is  $h_i = \tilde{\lambda}_i^{-1/2}$ . Note also that  $h_{\max}$  and  $h_{\min}$  are the maximum and minimum allowable target sizes while the constant  $C$  controls the level of error and, consequently, the final number of mesh elements.

## 3. VISUALIZATION METHOD

Streamlines are well known tools for visualizing the structure of a vector field. They are generalized to second order

tensor fields by tensor lines [8] or hyperstreamlines [9] tangent to the tensor's eigenvector fields. The present visualization method forms a pseudo-mesh by tracing a set of tensor lines for each eigenvector field and is very similar to the polygonal surface remeshing technique proposed by Alliez et al. [13]. However, instead of a curvature tensor, the metric tensor is used to generate the two orthogonal sets of lines of the pseudo-mesh. Nevertheless, if the solution to which the mesh has to be adapted is considered as a hypersurface then its Hessian, and thus the metric, is related to the curvature of this surface. The present method is therefore a natural extension of the algorithm presented by Alliez et al.

### 3.1. Tensor Field Decomposition

A two-dimensional metric tensor field can be decomposed into a major and a minor eigenvector field. The major field corresponds to the eigenvectors with the biggest eigenvalues and the minor to the smallest ones. To compute those fields from the metric  $\mathcal{M}$  at every point of the domain, the deviator  $\mathcal{D}$  can be defined [15]

$$\mathcal{D} = \mathcal{M} - \frac{1}{2} \text{tr}(\mathcal{M}) \mathcal{I} = \begin{pmatrix} \alpha & \beta \\ \beta & -\alpha \end{pmatrix} \quad (6)$$

where  $\text{tr}(\mathcal{M})$  denotes the trace of  $\mathcal{M}$  and  $\mathcal{I}$  is the identity matrix. The eigenvalues are then computed as

$$\lambda_{1,2} = \frac{1}{2} \text{tr}(\mathcal{M}) \pm \sqrt{\alpha^2 + \beta^2} \quad (7)$$

while the eigenvectors are given by

$$\vec{e}_i = \frac{\vec{e}'_i}{\|\vec{e}'_i\|} \quad (8)$$

where

$$\vec{e}'_{1,2} = \begin{pmatrix} \beta \\ -\alpha \pm \sqrt{\alpha^2 + \beta^2} \end{pmatrix}$$

and the subscripts 1 and 2 correspond to the major and minor fields respectively, i.e.,  $\lambda_1 \geq \lambda_2$ .

Note that metrics constructed from *a posteriori* error analysis are usually discrete and defined only at the vertices of the computational mesh. Term-by-term linear interpolation is used to compute  $\mathcal{M}$  within each mesh element and  $\mathcal{D}$  is then computed from the interpolated  $\mathcal{M}$ .

Furthermore, the tensor  $\mathcal{D}$  represents the deviation of the metric from isotropy, i.e.,  $\lambda_1 = \lambda_2$  which implies  $\alpha = \beta = 0$ . Isotropic tensors are degenerate cases where no major or minor eigenvector can be distinguished. They correspond to umbilic points for the curvature tensor on a three-dimensional surface as noted by Alliez et al. [13]. Whether for curvature or metric tensors, isotropic regions are important topological features of the tensor field. Special tensor lines called separatrices originate from isolated isotropic points and effectively divide the domain into non-degenerate regions. The set of separatrices constitutes a topological skeleton of the tensor field [10]. Locating such isolated isotropic points for metrics linearly interpolated on triangular meshes can be done by looping through all the mesh elements and solving a  $2 \times 2$  linear system. However,

metrics can also be isotropic along lines and in whole regions. Although not generally prevalent, such regions are problematic and must be detected because the present visualization method cannot be applied directly there. Several techniques are proposed in Section 4 to deal with those isotropic regions.

### 3.2. Tensor Line Integration

To plot lines tangent to the metric eigenvector fields, a technique analogous to streamline integration is used. Starting with a seed point, the metric field is interrogated, the local tensor is decomposed and the target eigenvector is used to advance to a new point. A fourth order adaptive Runge-Kutta integration scheme is used [16]. However, since eigenvectors are actually determined modulo a non-zero scalar coefficient, they only have direction but neither norm or orientation. Those quantities are needed for the integration process and have to be somehow artificially specified. The norm  $v$  used by Tricoche [15] is given by

$$v = \alpha^2 + \beta^2 = \frac{1}{4} (\lambda_1 - \lambda_2)^2 \quad (9)$$

However, the eigenvalues of the metric tensors used for mesh adaptation vary widely as the squared inverse of the prescribed target sizes and can cause numerical problems. That is why the following normalized  $v$  was used instead

$$v = \left( \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} \right)^2 \quad (10)$$

Furthermore an artificial orientation is chosen by assuming a locally smooth variation of the eigenvector fields. Of the two possible orientations at each new tensor line point, the one forming the minimum angle  $\theta$  with the orientation at the previous point is taken (Fig. 2). This smooth variation hypothesis breaks down near degenerate points. Those isotropic points constitute bifurcations where the eigenvectors are not defined, i.e., they can take any direction. The artificial velocity norm  $v$  is, however, equal to zero in those regions and the integration process has to be stopped anyway.

### 3.3. Pseudo-Mesh Generation

To gain insight on the structure of the metric field, the domain is saturated with tensor lines tangent to the two eigenvector fields. The distance between pairs of lines in the same

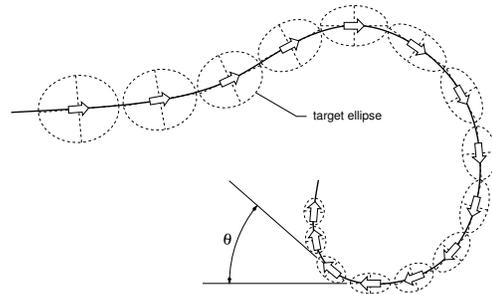


Figure 2: Tensor line integration.

field should be as close as possible to a unit metric length. The resulting network of lines constitute a pseudo-mesh that is visually close to a perfectly adapted mesh, but without any continuity or conformity constraints, and is thus easy to interpret in a mesh generation context.

To achieve such a saturation, tensor lines are integrated from seed points until they either are too close to existing lines in the same eigenvector field, leave the domain or reach a degenerate isotropic region. Note that, to make the final network of tensor lines as close as possible to an actual mesh, the proximity checks used to stop the integration are performed only within a small angular range  $\theta_p$  perpendicular to the integrated line (Fig. 3). A value of 20 degrees for  $\theta_p$  was used in practice. Furthermore, an Alternating Digital Tree (ADT) [17] is used to accelerate those proximity tests and each eigenvector field is treated independently.

The seed generation and selection process, inspired by streamline placement methods [18,19], is critical in ordering the integration of the tensor lines. The first lines to be plotted will indeed be the longest and thus should be the most important. Any isolated degenerate point is inserted in an initial set of seed points. By definition, an infinite number of tensor lines go through those points but the most important ones are the separatrices. Degenerate points can be classified by their number of separatrices: wedges have only one separatrix while trisectors have three. For linear tensor fields, the departing angle of those separatrices can be computed using a third-order polynomial equation [15]. A tensor line can theoretically be integrated for each pair of degenerate point and separatrix angle. Once all the degenerate points have been processed, potential seeds are placed alongside the separatrices. For each integrated tensor line point, two seed points are placed perpendicularly to the line at a distance  $d_s$

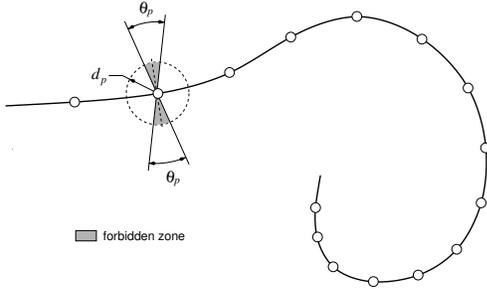


Figure 3: Proximity check.

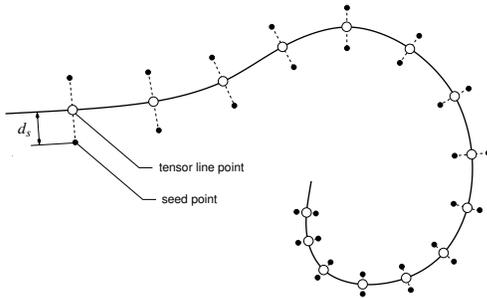


Figure 4: Seed point placement.

---

#### Algorithm 1 Tensor line saturation

---

**input:** set of potential seeds  
**repeat**  
    choose a seed  
    integrate tensor line from this seed  
    discard old seeds too close to the new tensor line  
    add new seeds along the new tensor line  
**until** no more potential seeds left

---

(Fig. 4). Using this initial set of non-degenerate potential seed points, the domain is saturated with tensor lines using Algorithm 1. Note that, for those non-degenerate seeds, two half-lines are actually integrated: one along each possible orientation of the local eigenvector. Once a new tensor line has been integrated, the next seed to be processed is the one that best fits the local requirements, i.e., unit metric distance to the closest line in the same eigenvector field. Before terminating the plotting process, the domain is interrogated at random points. If saturation is not adequate locally, i.e., the random point is farther away than a unit metric length to the closest line in the same eigenvector field, then this point is added to the set of seeds and Algorithm 1 is restarted. This last check usually results in only a handful of new lines.

Finally note that  $d_p$  (Fig. 3) and  $d_s$  (Fig. 4) should correspond to unit metric distances. Such a unit distance can be approximated by the locally prescribed target size  $h$  in the direction of the eigenvector field perpendicular to the considered one. However, to decrease fragmentation of the tensor lines,  $d_p$  was set to  $h/\sqrt{2}$  and  $d_s$  to  $\sqrt{2}h$ . Those values mirror the refinement and coarsening thresholds used on mesh edges in simplicial adaptation.

## 4. RESULTS AND DISCUSSION

The present visualization method has many advantages over traditional iconic tensor visualization but also has some limitations. To illustrate them, both analytical and solution-based metrics are visualized using pseudo-meshes.

### 4.1. Analytical Metrics

The first case is an isotropic metric commonly used to test mesh adaptation algorithms [2]

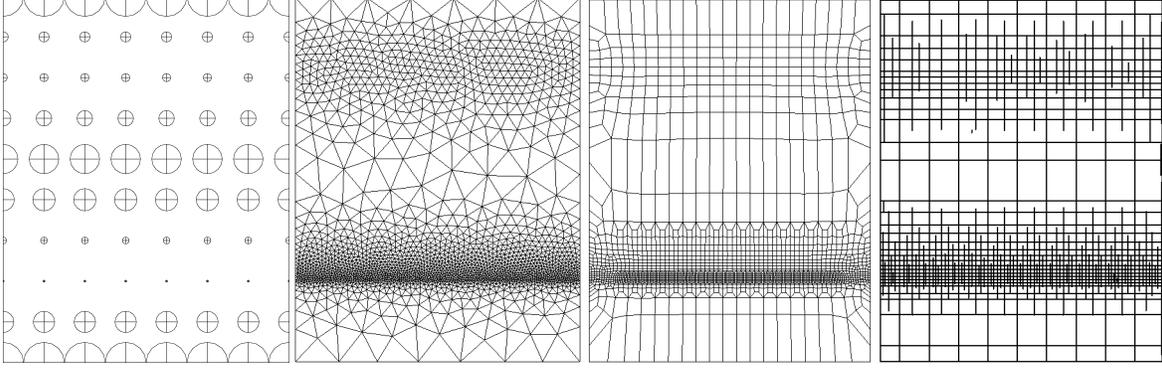
$$\mathcal{M} = h^{-2} \mathcal{I} \quad (11)$$

where  $h$  is given by

$$h = \begin{cases} 1 - 19y/40 & \text{if } y \in [0, 2], \\ 20^{(2y-9)/5} & \text{if } y \in [2, 4.5], \\ 5^{(9-2y)/5} & \text{if } y \in [4.5, 7], \\ 1/5 + (y-7)^4/20 & \text{if } y \in [7, 9]. \end{cases}$$

However, as mentioned in Section 3, isotropic metrics are considered degenerate and cannot be visualized by the present method. That is why the definition of this metric has been modified to make it slightly anisotropic as follows

$$\mathcal{M} = h^{-2} \begin{pmatrix} 1 & 0 \\ 0 & (1 + \epsilon)^{-2} \end{pmatrix} \quad (12)$$



**Figure 5: Visualization of the isotropic metric given by Eq. (12). From left to right: iconic visualization; adapted triangular mesh [20]; adapted quadrilateral mesh [21]; pseudo-mesh visualization.**

A small  $\epsilon$  does not disturb the structure of the field but enables the algorithm to distinguish two different eigenvalues and thus trace tensor lines. A value of 0.01, corresponding to a one percent difference between the horizontal and vertical target sizes, was used to generate the pseudo-mesh in Fig. 5. Note that, since, by construction, a non-zero  $\epsilon$  results in a slightly anisotropic metric everywhere, no degenerate isotropic point exists to initialize the tensor line saturation process and random seeds were used instead. For comparison, an iconic visualization as well as the final triangular and quadrilateral adapted meshes corresponding to the same metric, but with  $\epsilon$  set to zero, are also presented. The iconic visualization reflects the local target element size at discrete points of the domain with the radius of its circles, but gives little information on the structure of the metric field. The triangular adapted mesh, on the other hand, conveys a more continuous visual representation of the metric. The structure of the metric field reflected by this mesh agrees with the pseudo-mesh visualization and confirms that introducing a small  $\epsilon$  does not disturb too much this field. Such an approach cannot be used systematically to remove isotropic regions but, as shown in Section 4.2, those regions are rather exceptional in practical solution-based metrics and can be removed by appropriate smoothing.

Although an adapted mesh is a good way to visualize a metric *a posteriori*, the quality of such a visualization is strongly dependent on the performance of the adaptation algorithm. Furthermore, metric visualization should be possible before any adaptation to determine if a perfect unit mesh is even feasible. Take for example the adapted quadrilateral mesh presented in Fig. 5. This quadrilateral mesh does not comply as well as the triangular mesh to the prescribed metric because the particular cubical adaptation algorithm that was used can only refine but neither coarsen nor reconnect unlike the simplicial one [21]. A metric visualization through such a mesh is thus biased by the adaptation algorithm. An even more important problem is that a quadrilateral mesh perfectly adapted to the metric given by Eq. (12) is impossible as can be seen in its pseudo-mesh visualization. The prescribed size transitions can indeed only be achieved using hanging nodes or non-quadrilateral elements. This demonstrates the utility of the present visualization method to evaluate mesh adaptation control metrics.

Using pseudo-meshes has, however, some caveats. First of

all, although they are not biased by an adaptation algorithm, they do not exactly reflect a perfect unit mesh in the metric space. Approximations have indeed been introduced in the metric length computation and tensor lines are not placed exactly at unit metric distances. This results in some stray lines here and there. However, this compromise is necessary to minimize fragmentation of the lines and improve visual clues on the overall structure of the tensor field. On average, the spacing is close to unity and the pseudo-meshes are as close to a unit mesh as possible. Furthermore, since a pseudo-mesh does not have to comply to the usual constraints of a mesh, such as conformity and continuity, it can be generated even if a proper mesh cannot.

The second analytical case, presented in Fig. 6, is also a classic but an anisotropic one [2]. It will be used to illustrate how the present visualization algorithm treats non-isolated degenerate points and is given by

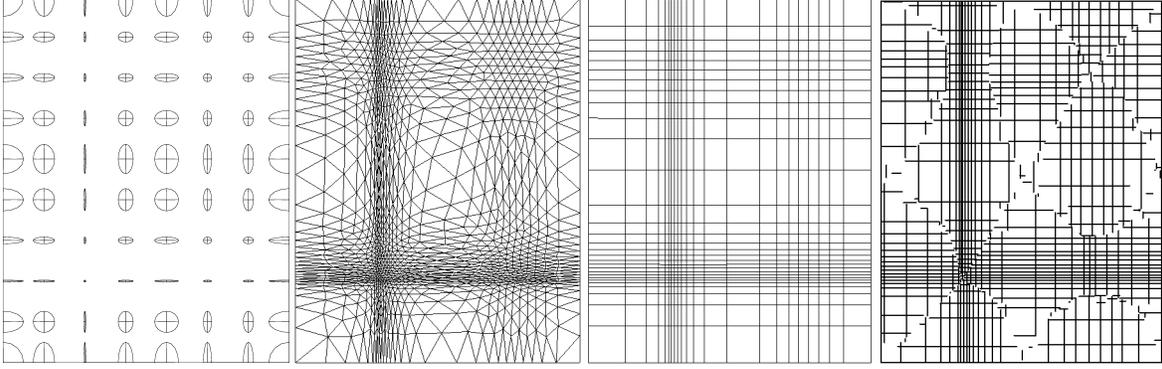
$$\mathcal{M} = \begin{pmatrix} h_1^{-2} & 0 \\ 0 & h_2^{-2} \end{pmatrix} \quad (13)$$

where  $h_1$  and  $h_2$  are computed as follows

$$h_1 = \begin{cases} 1 - 19x/40 & \text{if } x \in [0, 2], \\ 20^{(2x-7)/3} & \text{if } x \in ]2, 3.5], \\ 5^{(7-2x)/3} & \text{if } x \in ]3.5, 5], \\ 1/5 + (x-5)^4/20 & \text{if } x \in ]5, 7], \end{cases}$$

$$h_2 = \begin{cases} 1 - 19y/40 & \text{if } y \in [0, 2], \\ 20^{(2y-9)/5} & \text{if } y \in ]2, 4.5], \\ 5^{(9-2y)/5} & \text{if } y \in ]4.5, 7], \\ 1/5 + (y-7)^4/20 & \text{if } y \in ]7, 9]. \end{cases}$$

This metric presents a set of degenerate lines where  $h_1 = h_2$  (Fig. 7). Locating those degenerate lines automatically is not trivial. Furthermore, they are not tensor lines and thus cannot be visualized directly by the present method. However, since they actually stop tensor line integration, they abruptly disrupt the tensor line network giving thereby visual clues on their location as shown in Fig. 6. Again, an iconic visualization and the final triangular and quadrilateral adapted meshes are presented in addition to the pseudo-mesh visualization. The elliptical icons reflect the prescribed size, stretching and



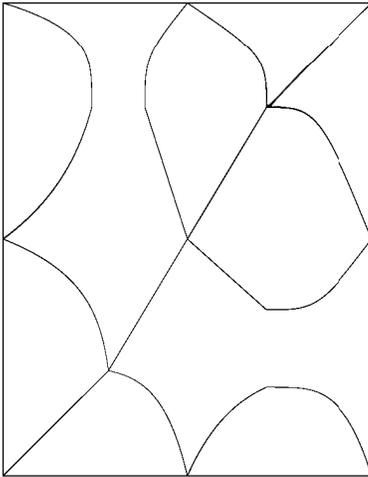
**Figure 6: Visualization of the anisotropic metric given by Eq. (13). From left to right: iconic visualization; adapted triangular mesh [20]; adapted quadrilateral mesh [21]; pseudo-mesh visualization.**

orientation of the target mesh elements. Using the pseudo-mesh as a reference, the quadrilateral mesh seems better adapted to this particular metric than the triangular one. This is due to the axis alignment of the prescribed metric topology. The adapted quadrilateral mesh does not, however, give any clue on the location of the degenerate isotropic lines.

Finally note that, again, no isolated degenerate point exists for this metric and random seeds were used to initialize the pseudo-mesh generation. Note also that, near the degenerate lines, the major and minor eigenvalues switch and neighboring perpendicular lines belong to the same eigenvector field. This defeats the perpendicular proximity checks and stops line integration prematurely explaining some fragmentation near those degenerate lines.

#### 4.2. Solution-Based Metrics

Analytical metrics are somewhat artificial but allow the illustration of the algorithm behavior in extreme conditions. The following metrics are more representative of real world cases and are constructed from the Hessian of a numerical solution. In those metrics, exactly isotropic regions are rare but



**Figure 7: Degenerate isotropic lines for the metric given by Eq. (13).**

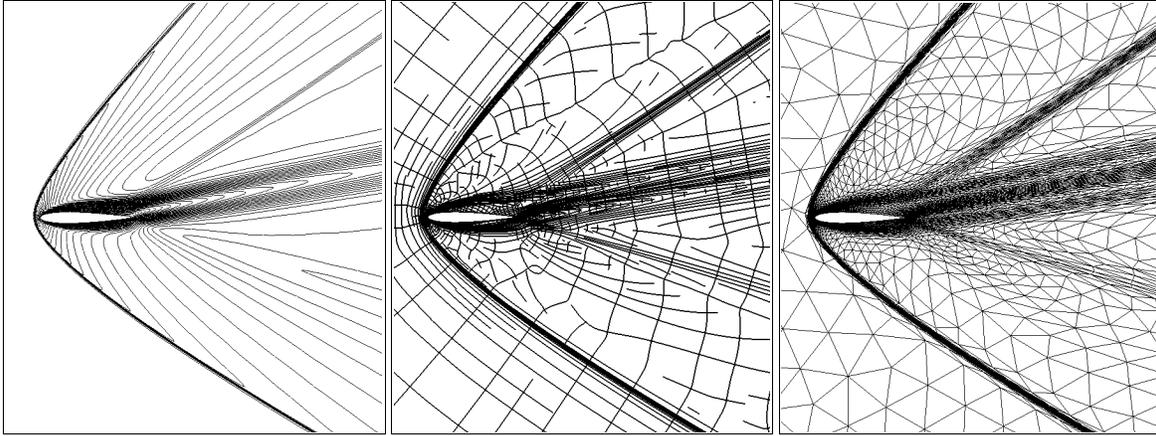
almost isotropic ones are not and isolated degenerate points are a plenty. Furthermore, these degenerate regions tend to be unstable and can be removed with a slight perturbation of the metric field such as a small amount of smoothing.

Figure 8 plots iso-Mach lines for the steady laminar supersonic flow around a NACA 0012 airfoil for an angle of attack of 10 degrees, a Reynolds number of 1000 and a Mach number of 2.0. This figure also presents the pseudo-mesh visualization of the target metric constructed from the Hessian of the solution Mach field as well as the resulting adapted triangular mesh. The pseudo-mesh was generated on a slightly smoothed metric to minimize degenerate regions. A simple term-by-term Laplacian like operator was used on the background triangular mesh employed as a support medium for the metric

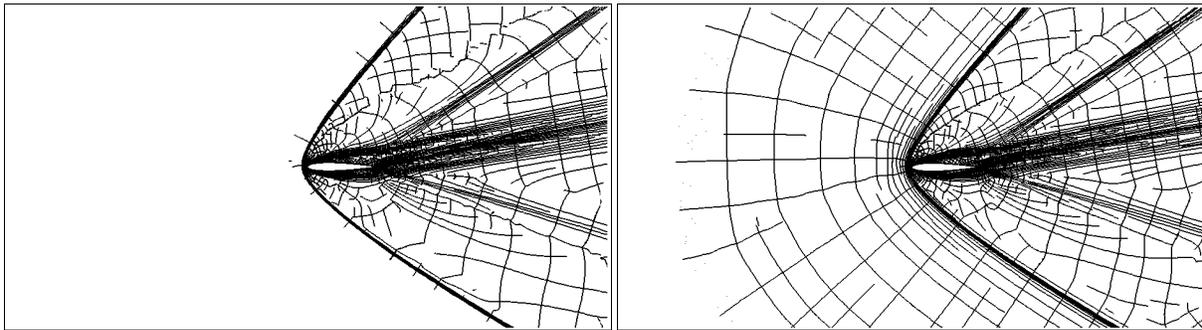
$$\mathcal{M}_i^{n+1} = \mathcal{M}_i^n + \omega \frac{\sum_j (\mathcal{M}_j^n - \mathcal{M}_i^n) / l_{ij}}{\sum_j 1/l_{ij}} \quad (14)$$

where  $n$  is an iteration counter,  $j$  denotes all the vertices sharing an edge with vertex  $i$ ,  $l_{ij}$  is the Euclidean distance between  $i$  and  $j$  while  $\omega$  is a relaxation factor. To try to avoid disturbing the metric as much as possible, only 10 iterations with a relaxation factor of 0.1 were performed. Figure 9 plots the pseudo-meshes generated for the original metric and the smoothed one. Note the blank region upwind of the detached bow shock. In a supersonic flow, there is little variation in this region and the metric prescribes uniform elements of size  $h_{\max}$  there. Such an isotropic region is impossible to visualize directly with the present method. However, this region is next to an anisotropic one and is very unstable. A small amount of smoothing makes it anisotropic enough for the algorithm to trace tensor lines.

However, care must be taken to avoid contaminating the metric with too much smoothing. Figure 10 presents another example of laminar compressible flow around a NACA 0012 airfoil but this time for unsteady transonic conditions, i.e., a zero angle of attack, a Reynolds number of 5000 and a Mach number of 0.85. The same amount of smoothing was used on the metric before generating its pseudo-mesh visualization. Although the overall structure of the metric was captured, some features in the smoothed metric as visualized by the pseudo-mesh have been slightly washed out compared to the corresponding adapted triangular mesh. Look in particular at the thickness of the shocks.



**Figure 8: Steady laminar compressible flow around a NACA 0012 airfoil for an angle of attack of 10 degrees, a Reynolds number of 1000 and a Mach number of 2.0. From left to right: iso-Mach lines; pseudo-mesh visualization of the metric; adapted triangular mesh [22].**

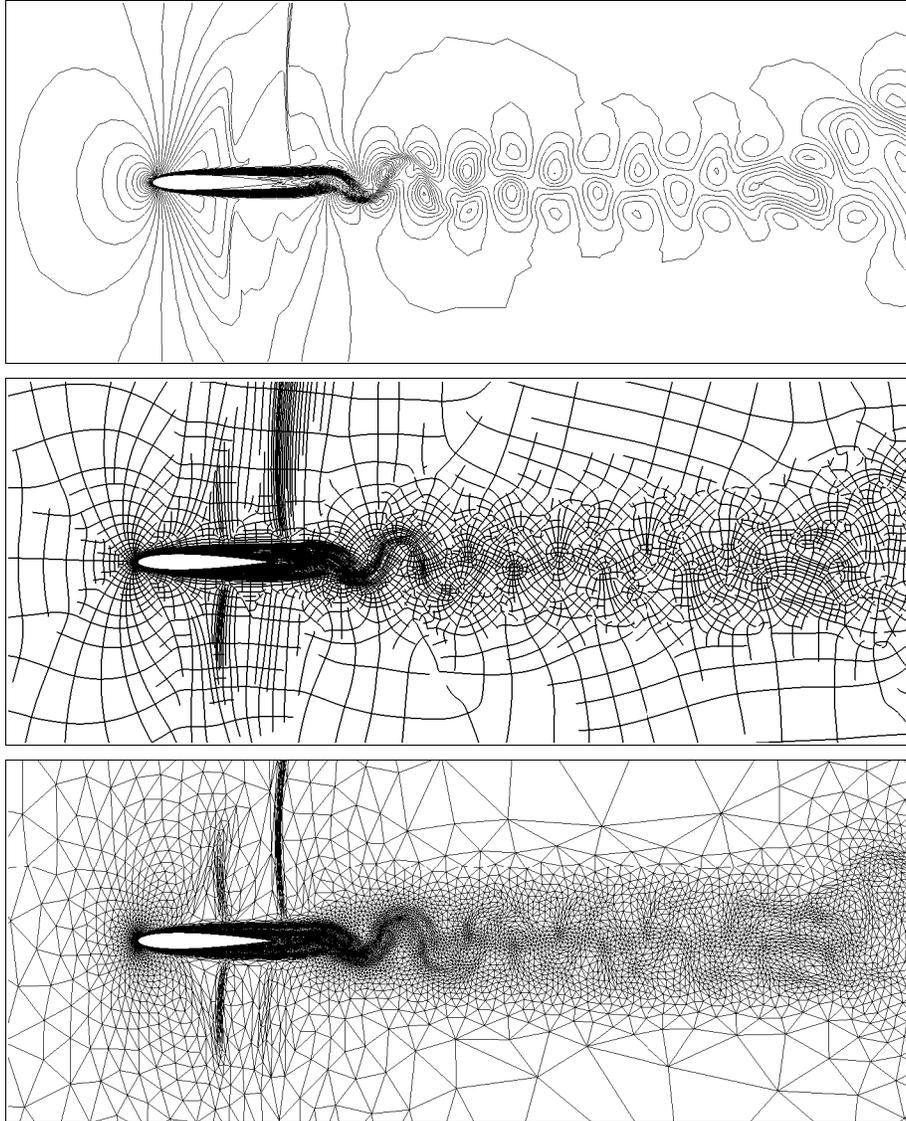


**Figure 9: Steady laminar compressible flow around a NACA 0012 airfoil for an angle of attack of 10 degrees, a Reynolds number of 1000 and a Mach number of 2.0. Pseudo-meshes for the original (left) and smoothed (right) metric.**

To illustrate the effect of various levels of smoothing, the next case is the portrait of German mathematician Bernhard Riemann (1826–1866). The gray levels of the bitmap photo in Fig. 11 are considered as the solution and their Hessian is used to construct the target metric. Although such a metric may appear to be nothing more than a toy application, it could eventually be used for image processing. Figure 11 shows the pseudo-mesh visualization of this metric without any smoothing as well as after 10 and 100 iterations with a relaxation factor of 0.1. Before any smoothing, the metric prescribes uniform elements of size  $h_{\max}$  in white regions without any significant variation of the gray levels. These regions are isotropic and appear as blank patches in the pseudo-mesh visualization because the tensor line integration algorithm cannot treat them. However, even outside those patches, the tensor lines seem to twist and turn and do not have any consistent directionality except along high-contrast contours. This is due to the noise in the bitmap gray levels that overwhelm the metric in the absence of strong gradients. Those almost degenerate regions contain a lot of isolated isotropic points, about 140 thousands for this particular case. When smoothing is applied, even only 10 iterations, these unstable regions tend to disappear and the almost random directionality becomes more coherent. How-

ever, Laplacian smoothing erodes sharp features and, after 100 iterations, the details of the photo are washed out. Paradoxically, smoothing reduces isotropy in almost degenerate regions but also reduces anisotropy in neighboring regions. In essence, it redistributes anisotropy and exposes a coherent underlying directionality. This observation is not so much interesting in the context of visualization as it is for adapted mesh generation from a pseudo-mesh as mentioned in Section 5. For visualization, the important thing to remember is that smoothing should be kept to a bare minimum, i.e., just enough to eliminate most degenerate regions but still retain the structure of the metric field. How much is case dependent but 10 iterations with a small relaxation factor around 0.1 seems adequate.

Note furthermore that adaptation algorithms also introduce at least some level of smoothing as shown in the adapted meshes of Fig. 11. Those algorithms indeed use refinement and coarsening criteria based on metric length and, since this length is integrated using Eq. (2), it indirectly smooths the effective metric field seen by those algorithms. Furthermore, a regularization step is usually applied at the end of the adaptation process and this step is little more than smoothing in the metric space. Therefore, even if a small amount of smooth-



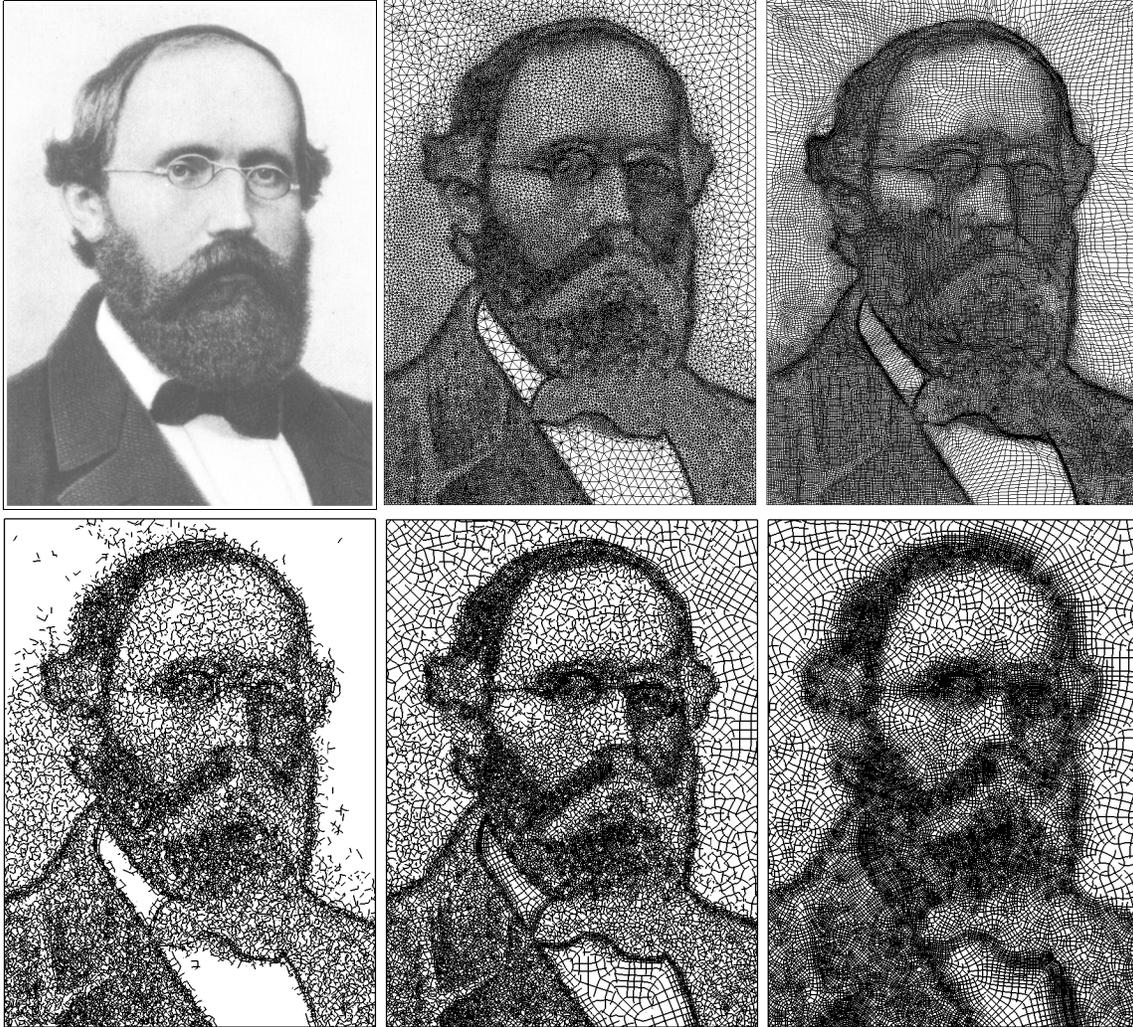
**Figure 10: Unsteady laminar compressible flow around a NACA 0012 airfoil for a zero angle of attack, a Reynolds number of 5000 and a Mach number of 0.85. From top to bottom: iso-Mach lines; pseudo-mesh visualization of the metric; adapted triangular mesh.**

ing is applied on the metric to generate the pseudo-mesh, the resulting visualization is likely to be more faithful than the corresponding adapted mesh, if one can be generated.

## 5. FUTURE DEVELOPMENTS

The main application of the present visualization method is the study of metric manipulations such as smoothing or interpolation for example. Metrics constructed with different Hessian computation methods could also be visualized and analyzed without any interference from adaptation algorithms. Similarly, different boundary conditions could be visually explored for metrics constructed from turbulent flows with special wall models.

There is, however, still room for improvement. For example, the metric length could be more precisely computed during the saturation process. To further decrease tensor line fragmentation, line integration could be stopped only if a new line stays close to an existing one more than a given portion of its length. The most important improvement, however, would be to find a more efficient way to deal with isotropic or almost isotropic regions. For example, the pseudo-mesh generation for the unsmoothed metric constructed from the portrait of Riemann (Fig. 11) required hours of CPU time on an AMD Athlon running at 1.4 GHz while the other test cases typically required only 5 to 10 minutes. This slow down was due to the almost isotropic regions and the sheer number of isolated isotropic points contained by those regions, i.e., about 140 thousands. For each of those degen-



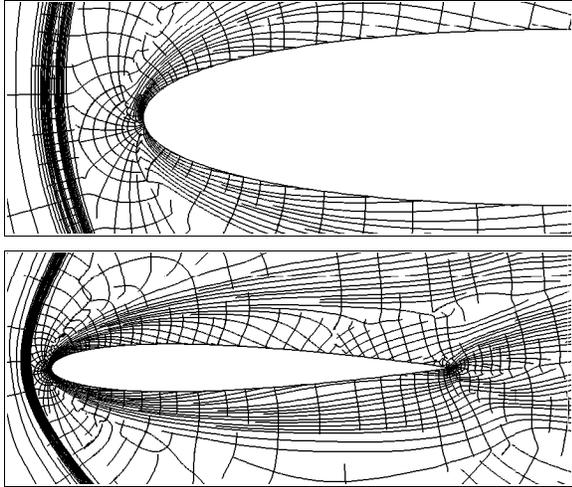
**Figure 11: Portrait of German mathematician Bernhard Riemann (1826–1866). First row, from left to right: photo; adapted triangular mesh [21]; adapted quadrilateral mesh [21]. Second row, from left to right: pseudo-mesh visualization of the metric without any smoothing; metric after 10 smoothing iterations; metric after 100 smoothing iterations.**

erate points, a number of separatrices had to be integrated in an almost degenerate neighborhood. Those tensor lines thus frequently changed direction and progressed at a very slow speeds. Presently, the only solution is to apply a small amount of smoothing on the metric. However, the limit to impose on the amount of smoothing to preserve the features of the metric is still case dependent and this issue should be addressed in future developments.

Furthermore, a pseudo-mesh is very close to a perfectly adapted unit mesh and it is thus tempting to try to generate a proper mesh from it, particularly an all-quadrilateral one. Look, for example, at the pseudo-mesh boundary layer in Fig. 12. As attractive as such a method may appear, the visualizations presented in the previous section show, however, that not all metrics are suitable for the generation of an all-quadrilateral mesh. Take for example the metric visualized in Fig. 5. A conformal all-quadrilateral mesh is clearly not feasible and either hanging nodes or non-quadrilateral

elements have to be introduced to perfectly match the prescribed metric. This is due to the decoupling of the prescribed mesh density from the topology of the target metric. There is, indeed, no link between the direction of the tensor lines and the prescribed target size. Based on the metric topology alone, the perfect mesh should be a uniform Cartesian grid. However, to match the prescribed target sizes, this grid should have varying density. If hanging nodes are to be avoided, then this grid cannot be Cartesian and the tensor lines should curve and bifurcate at degenerate points, acting as sources and sinks, to transition between high and low density regions. This is exactly what happens in the adapted quadrilateral mesh. This suggests that some type continuity constraint must be enforced on the metric field to ensure the feasibility of an all-quadrilateral mesh. Adequate preprocessing of target metrics should be explored in future developments.

Another obstacle for mesh generation from tensor lines is

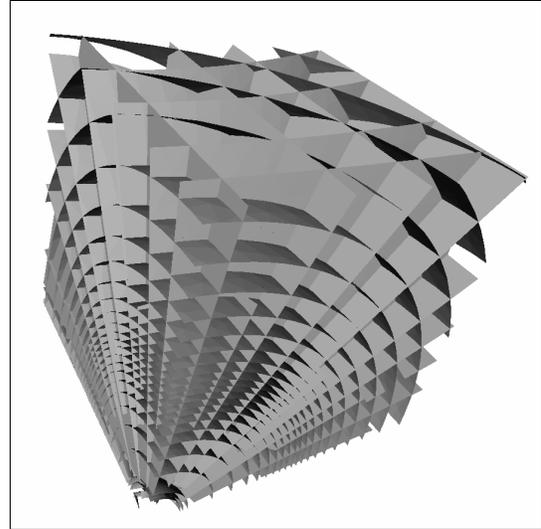


**Figure 12: Steady laminar compressible flow around a NACA 0012 airfoil for an angle of attack of 10 degrees, a Reynolds number of 1000 and a Mach number of 2.0. Details of the pseudo-mesh of Fig. 8: leading edge (above) and boundary layer (below).**

their absence in degenerate regions and their random directionality in almost degenerate ones. As mentioned previously, a little smoothing can solve this problem. This is the approach used by Alliez et al. [13] to generate a polygonal surface mesh from a network of curvature tensor lines. The resulting meshes are very attractive and are probably the closest an automated method can get to what a human expert, i.e., a computer graphics artist, would generate manually. However, in the context of adapted mesh generation, uncontrolled Laplacian smoothing erodes too much the main features of the metric. Although it gives more consistent directions to the tensor lines, it indeed results in more and more uniform target sizes. Future developments should thus improve the smoothing method to preserve the mesh clustering prescribed by the metric.

However, even if, with adequate smoothing and preprocessing of the metric, the generation of a proper adapted mesh is feasible, its cost efficiency compared to simplicial adaptation algorithms is uncertain. One way to improve this efficiency is to amortize the pseudo-mesh construction by generating a coarse mesh and then uniformly splitting the resulting elements. An even more efficient approach would be to use the pseudo-mesh to generate an adapted block decomposition of the domain combined with a fast structured mapping method. Adaptively refining block decompositions has shown that the quality of the results depends on the topology of the initial blocks [23]. An extension of the present work could eventually result in a method to generate such an initial block decomposition adapted to not only the domain geometry, as with a medial axis approach [24], but also to the solution.

Finally, a three-dimensional extension of the method could also be explored in future developments. In three dimensions, the metric is a  $3 \times 3$  symmetric positive-definite matrix. The metric field can thus be decomposed into three eigenvector fields and tensor surfaces are used instead of tensor lines to form a pseudo-mesh. A tensor surface is perpen-



**Figure 13: Pseudo-mesh visualization for a spherical metric.**

dicular to one of the eigenvector fields and tangent to the other two. Figure 13 shows an early result for a spherical analytical metric. As can be seen in this figure, occlusion problems may not be avoidable in three dimensions. However, the main purpose of such an extension would be adapted hex-dominant mesh generation and not visualization.

## 6. CONCLUSION

The proposed two-dimensional metric visualization method extends the polygonal surface remeshing algorithm developed by Alliez et al. [13] to generate a network of tensor lines visually close to a perfectly adapted mesh. Such a pseudo-mesh visualization is intuitive to understand in a mesh generation context and is not biased by any adaptation algorithm. Furthermore, it can be constructed even if a proper mesh cannot. Both analytical and solution-based metrics have illustrated its advantages as well as its limitations, particularly for isotropic metrics.

Pseudo-mesh visualization is an ideal tool to study metric manipulation methods but could also be used to generate proper all-quadrilateral meshes. However, not all metrics can be used for such a purpose and a preprocessing method should be developed to improve this potential. An adaptive hex-dominant mesh generation method from pseudo-meshes could also be interesting and justify the extension of the present method to three dimensions.

## 7. ACKNOWLEDGMENTS

The authors would like to thank NSERC for its financial support. Furthermore, please note that the solutions, meshes and iconic tensor visualizations were plotted using `medit`, a mesh visualization program developed by Pascal J. Frey of INRIA, France, as well as `VU`, a configurable visualization software tool for the display and analysis of numerical solutions developed by Benoit Ozell at CERCA, Québec.

## REFERENCES

- [1] M.-G. Vallet, *Génération de maillages éléments finis anisotropes et adaptatifs*. PhD thesis, Université Pierre et Marie Curie, Paris VI, France, 1992.
- [2] P.-L. George and H. Borouchaki, *Delaunay Triangulation and Meshing. Applications to Finite Elements*. Paris: Hermès, 1998.
- [3] P. J. Frey and P.-L. George, *Mesh Generation. Application to Finite Elements*. Paris: Hermès, 2000.
- [4] K.-F. Tchon, M. Khachan, F. Guibault, and R. Camarero, “Three-dimensional anisotropic geometric metrics based on local domain curvature and thickness,” *Comp.-Aided Design*. In Press.
- [5] H. Borouchaki, F. Hecht, and P. J. Frey, “Mesh gradation control,” *Int. J. Numer. Meth. Engng*, vol. 43, pp. 1143–1165, Nov. 1998.
- [6] X. Li, J.-F. Remacle, N. Chevaugéon, and M. S. Shephard, “Anisotropic mesh gradation control,” in *Thirteenth International Meshing Roundtable*, (Williamsburg, VA), Sandia National Laboratories, Sept. 2004.
- [7] F. Alauzet and P. J. Frey, “Estimateur d’erreur géométrique et métriques anisotropes pour l’adaptation de maillage. Partie I : aspects théoriques,” Tech. Rep. RR-4759, Institut National de Recherche en Informatique et en Automatique, France, Mar. 2003.
- [8] R. Dickinson, “A unified approach to the design of visualization software for the analysis of field problems,” in *Three-Dimensional Visualization and Display Technologies*, vol. 1083 of *SPIE Proceedings*, (Los Angeles, CA), pp. 173–180, SPIE – The International Society of Optical Engineering, Jan. 1989.
- [9] T. Delmarcelle and L. Hesselink, “Visualizing second-order tensor fields with hyperstreamlines,” *IEEE Computer Graphics and Applications*, vol. 13, pp. 25–33, July 1993.
- [10] T. Delmarcelle and L. Hesselink, “The topology of symmetric, second-order tensor fields,” in *IEEE Visualization ’94*, (Washington, D.C.), pp. 140–147, 1994.
- [11] X. Zheng and A. Pang, “Volume deformation for tensor visualization,” in *IEEE Visualization ’02*, (Boston, MA), pp. 379–386, 2002.
- [12] G. Kindlmann, D. Weinstein, and D. Hart, “Strategies for direct volume rendering of diffusion tensor fields,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, pp. 124–138, Apr. 2000.
- [13] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun, “Anisotropic polygonal remeshing,” *ACM Transactions on Graphics. Special issue: Proceedings of ACM SIGGRAPH 2003*, vol. 22, pp. 485–493, July 2003.
- [14] Ph. G. Ciarlet, “Basic error estimates for elliptic problems,” in *Handbook of Numerical Analysis* (P. G. Ciarlet and J.-L. Lions, eds.), vol. II, pp. 17–351, Amsterdam: North-Holland, 1991.
- [15] X. Tricoche, *Vector and Tensor Field Topology Simplification, Tracking, and Visualization*. PhD thesis, Schriftenreihe Fachbereich Informatik (3), Universität Kaiserslautern, Germany, 2002.
- [16] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C : The Art of Scientific Computing*. UK: Cambridge University Press, 2nd ed., 1992.
- [17] J. Bonet and J. Peraire, “An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems,” *Int. J. Numer. Meth. Engng*, vol. 31, pp. 1–17, 1991.
- [18] B. Jobard and W. Lefer, “Creating evenly-spaced streamlines of arbitrary density,” in *8th Eurographics Workshop on Visualization In Scientific Computing*, (Boulogne-sur-Mer, France), pp. 45–55, Apr. 1997.
- [19] V. Verma, D. Kao, and A. Pang, “A flow-guided streamline seeding strategy,” in *IEEE Visualization ’00*, (Salt Lake City, UT), pp. 163–170, 2000.
- [20] P. Labbé, J. Dompierre, M.-G. Vallet, F. Guibault, and J.-Y. Trépanier, “A measure of the conformity of a mesh to an anisotropic metric,” in *Tenth International Meshing Roundtable*, (Newport Beach, CA), pp. 319–326, Sandia National Laboratories, Oct. 2001.
- [21] K.-F. Tchon, J. Dompierre, and R. Camarero, “Automated refinement of conformal quadrilateral and hexahedral meshes,” *Int. J. Numer. Meth. Engng*, vol. 59, pp. 1539–1562, Mar. 2004.
- [22] J. Dompierre, P. Labbé, and F. Guibault, “Controlling approximation error,” in *Second M.I.T. Conference on Computational Fluid and Solid Mechanics* (K. Bathe, ed.), (Massachusetts Institute of Technology, Cambridge, MA), pp. 1929–1932, June 2003.
- [23] K.-F. Tchon, F. Guibault, J. Dompierre, P. Labbé, and R. Camarero, “Solution adaptive refinement of multi-block decompositions,” in *16th AIAA Computational Fluid Dynamics Conference*, no. AIAA-2003-3821, (Orlando, FL), June 2003.
- [24] M. A. Price, C. G. Armstrong, and M. A. Sabin, “Hexahedral mesh generation by medial surface subdivision: Part I. Solids with convex edges,” *Int. J. Numer. Meth. Engng*, vol. 38, pp. 3335–3359, Oct. 1995.