P2 Cavity Operator with Metric-Based Volume and Surface Curvature

Lucien Rochery, Adrien Loseille

P^2 CAVITY OPERATOR WITH METRIC-BASED VOLUME AND SURFACE CURVATURE

Lucien Rochery[1] and Adrien Loseille[2]

[1] Phd Student, lucien.rochery@inria.fr, INRIA, 1 Rue Honoré d'Estienne d'Orves, 91120 Palaiseau, France
[2] Research scientist, adrien.loseille@inria.fr, INRIA, 1 Rue Honoré d'Estienne d'Orves, 91120 Palaiseau, France.

ABSTRACT

This paper describes theoretical developments and algorithms involved in the design of a P^2 cavity operator to generate anisotropic curved meshes. Both volume and surface are adapted. A high-level approach is chosen, such that the existing P^1 cavity operator is used as-is to handle topology. The P^2 extension performs the curving process and ensures geometric validity. Volume curvature is based on Riemannian edge length minimization, first requiring a description of the metric field along a Bézier edge: this leads to the proposed high-order extension of the log-Euclidean scheme and differentiation of geometrical quantities in this framework. Surface curvature is based on similar principles, with the added difficulty of CAD or CAD surrogate projection. Numerical results illustrating the P^2 cavity operator's ability to recover curvature, from surface geometry to boundary layers to metric fields are presented. Examples are based on 3D real-world geometries encountered in Computational Fluid Dynamics (CFD). This framework allow us to curve highly anisotropic meshes with around 10 million elements within minutes.

Keywords: high-order, mesh generation, anisotropic mesh adaptation, metric based, cavity operator

1. INTRODUCTION

High-order numerical methods for the resolution of Partial Derivatives Equations (PDEs) have been attracting interest over their ability to solve a wide range of numerical problems with greater precision-over-cost ratio [1, 2]. The need for curved meshes has been established as far back as the 70s with the proof that optimal convergence of high-order methods is only possible with a curved boundary in the case of elliptic problems [3, 4] and later for hyperbolic problems, where physical features are lost when the boundary is left piecewise linear [5].These curved meshes are polynomial in nature with high-order (hereafter P^k) elements defined from polynomial mappings on the reference element.

Despite this need, the robust and automatic construction of valid curved meshes remains an open problem. Existing methods rely on input P^1 meshes that are then elevated to a higher degree, preventing back-to-back high-order adaptation. Indeed, a curved mesh would no longer be elligible as input. Likewise, the main preoccupation lies in curving the boundary, whereas volume curvature is kept to the minimum that affords validity. Some methods employ *ad-hoc* PDEs [6, 7, 8, 9, 10] or a variational approach [11] to displace volume nodes causing invalidity. Others use direct optimization procedures [12, 13] to correct invalid meshes with curved boundary.

Anisotropic mesh adaptation has established itself as an essential element of efficient numerical simulation, namely for CFD where strongly anisotropic physical phenomena are observed. Again, several approaches exist. The first, labelled p-adaptation, enriches the polynomial space on a per-element basis and, therefore, requires strong coupling with the solver [14]. The second, commonly called r-adaptation, limits mesh operations to vertex moving, thus sticking very close to the constraint on mesh complexity [15]. The third, h-adaptation, relies on local modifications to adhere to prescribed sizes [16, 17, 18] by applying the com-

plete range of meshing operators. These methods all share in common that they attempt to minimize simulation error at a given number of mesh vertices (mesh complexity). As such, they provide drastically optimal meshes in the sense of the error-over-cost ratio. This process has been applied to about all common physical situations, such as the steady [19, 20] and unsteady [21, 22] Euler equations, the steady Navier-Stokes equations in the context of RANS simulation [23, 24], fluid-structure interaction [25], acoustics, electromagnetics, magnetohydrodynamics, solid mechanics [26, 27, 28, 29]... This work inscribes itself in the third family of mesh adaptation methods, which resorts to mesh topology changes.

Metric fields link particular error estimates with automatic mesh adaptation. Both low-order [17, 18] and high-order metrics [30] have been derived to control interpolation error. Metric-based adjoint estimates allow to control the solution of a chosen PDE [20] or derived quantity of interest such as drag or lift [31]. A metric field locally distorts the measure of distance such that, when the mesh adaptation algorithm has constructed an uniform mesh in the induced Riemannian space, it happens to be strongly anisotropic in the usual Euclidean (physical) space. Therefore, anisotropy appears naturally, without it ever being explicitly sought by the (re)meshing algorithm.

Robustness and modularity of the general remeshing algorithm may be derived from the use of a single topological operator such as the cavity operator [32]. This operator remeshes element subsets (so-called cavities) by reconnecting cavity boundary nodes to a given point in space, already belonging to the mesh or otherwise. This very elementary operation can handle the most common topological operations: insertions, collapses, edge or face swaps. Therefore, it is central both to mesh adaptation (node creation, deletion) and to mesh optimization (mainly swaps).

1.1 Contributions

This work inscribes itself in an attempt to extend the unit-mesh framework [17, 18] of anisotropic mesh adaptation to high-order meshes. While a metric field's values prescribe element anisotropy, we propose that its *variations* drive the curvature of polynomial elements. The metric field's own intrinsic curvature may derive from any error estimate, such as a boundary approximation [33, 34] or interpolation error estimate. So far, interpolation error estimates on high-order elements are isotropic ([35] in L^2 and [36] in L^1 norms) or require that the curvature of the element be bounded, essentially establishing a range where it behaves as linear ([37] on non-linear quads).

The work presented here is twofold. Firstly, a practical method for mapping metrics onto element curvature must be devised. Ideally, the quantity of interest would be interpolation error directly. However, for lack of anisotropic estimates, the present work settles for Riemannian edge length minimization. This is conjectured to be correlated to interpolation error minimization, as in [38]. Fig. 1 illustrates the concepts at work. A new extension of the classic log-Euclidean metric interpolation scheme is first proposed, defining metric interpolation on high-order elements while conserving the desireable qualities of the classic scheme, such as convexity of interpolated metric determinant. From this new scheme, a closed form expression of P^2 Bézier edge length under a metric field is derived and differentiated with regards to control node position. This is then used for fast Riemannian edge length optimization. More general applications are considered, such as anisotropic distortion measure.

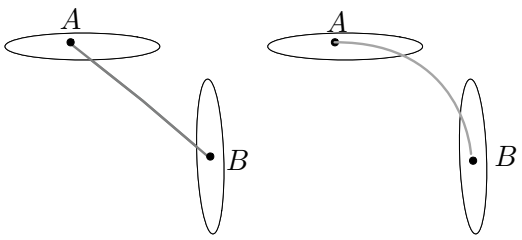

Figure 1: A metric field's intrinsic curvature recovered through Riemannian edge length minimization: the second edge is shorter than the first.

Secondly, curving must be carried out in the global adaptation algorithm. This takes the form of an extension of the classic cavity operator. This new P^2 cavity operator integrates edge curving in its reconnection phase. Through systematic Jacobian-based validity checks, it guarantees global validity at all times. Furthermore, it is the single topological operator in use in our existing anisotropic mesh adaptation algorithm AMG/feflo.a [39] and is capable of handling degree 2 meshes as both input as output.

A first prototype was implemented and cases illustrating the first results of this operator on industrial cases (NASA Common Research Model, C608 Low-Boom Flight Demonstrator) show that it is able to curve a highly anisotropic mesh (maximum ratio of anisotropy around 5000) based on metric and surface curvature while maintaining reasonable execution times (20M element mesh in 9min).

1.2 Metric-based mesh adaptation

Mesh adaptation places mesh generation and PDE solving in a loop. Using error estimates, it produces successive meshes that converge to optimality with re-

gards to precision over cost ratio [40] for a quantity of interest. Metric fields translate error estimates into geometrical data readily useable by the remeshing algorithm. In the case of Hessian-based error estimation, the recipe is simple: diagonalize the Hessian, apply absolute value to the eigen-values, then recompose and normalize for desired mesh complexity. A similar process exists for surface error estimation using the first fundamental forms of the surface [33]. This metric field then distorts how distances, angles and volumes are computed, making it so that an uniform mesh in the induced space may be highly anisotropic in physical space [17, 18]. The metric-based adaptation loop (illustrated Fig. 2) can be broken down into two steps:

1. A metric field is derived from an error estimate of the quantity of interest on the current mesh
2. The mesh is modified to be uniform in the Riemannian space induced by the metric field

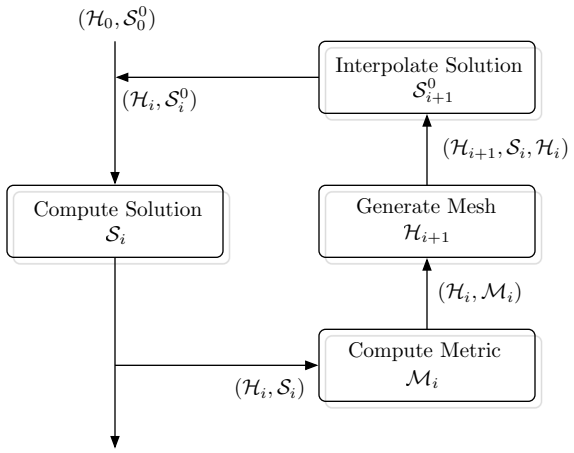

Figure 2: The mesh adaptation loop iterating over successive meshes $\mathcal{H}_i$, metric fields $\mathcal{M}_i$ and PDE solution $\mathcal{S}_i$.

We now define metrics more precisely, beginning with the single metric before moving on to metric fields and discrete metric fields extended by log-Euclidean interpolation. Let $d \in \{2, 3\}$ the space dimension, $\mathcal{M} \in \mathcal{M}_d(\mathbb{R})$ symmetric and positive-definite and $(\cdot, \cdot)$ the canonical scalar product. The bi-linear mapping

$$(\cdot, \cdot)_{\mathcal{M}} : (x, y) \in (\mathbb{R}^d)^2 \mapsto (x, \mathcal{M}y) = {}^t x \mathcal{M} y$$

defines a scalar product on $\mathbb{R}^d$. Despite the name, it is $\mathcal{M}$ rather than $(\cdot, \cdot)_{\mathcal{M}}$ that is said to be the metric. Due to it being positive-definite, $\mathcal{M}$ admits an orthogonal factorization $\mathcal{M} = \mathcal{R}\Lambda {}^t\mathcal{R}$ where the coefficients $\lambda_1 \leq ... \leq \lambda_d$ of the diagonal matrix Λ are strictly positive and the columns of $\mathcal{R}$ form an orthonormal basis of $\mathbb{R}^d$. To understand how $\mathcal{M}$ alters norm computations, let us look at its unit ball $\mathcal{B}_{\mathcal{M}}(0, 1)$:

$$\begin{aligned} Y \in \mathcal{B}_{\mathcal{M}}(0,1) &\iff {}^t Y \mathcal{R} \Lambda {}^t\mathcal{R} Y = 1 \\ &\iff {}^t(\Lambda^{1/2}\, {}^t\mathcal{R} Y)(\Lambda^{1/2}\, {}^t\mathcal{R} Y) = 1 \\ &\iff \Lambda^{1/2}\, {}^t\mathcal{R} Y \in \mathcal{B}(0,1) \end{aligned}$$

Therefore, $\mathcal{B}_{\mathcal{M}}(0, 1) = R\Lambda^{-1/2}\mathcal{B}(0, 1)$, *i.e.* the unit ball of $\mathcal{M}$ is a rotation by $\mathcal{R}$ of the unit ball distorted by $\Lambda^{-1/2}$. It comes that $\mathcal{B}_{\mathcal{M}}(0, 1)$ is an ellipsoid of axes along the columns of $\mathcal{R}$ and respective sizes $\frac{1}{\sqrt{\lambda_i}}$. Conversely, an ellipsoid uniquely defines such a metric, which justifies the graphical representation of metrics as ellipsoids (as in fig. 3). Furthermore, the Euclidean space $(\mathbb{R}^d, (\cdot, \cdot)_{\mathcal{M}})$ is isometric to $(\mathbb{R}^d, (\cdot, \cdot))$ through the mapping $\mathcal{R}\Lambda^{-1/2}$.

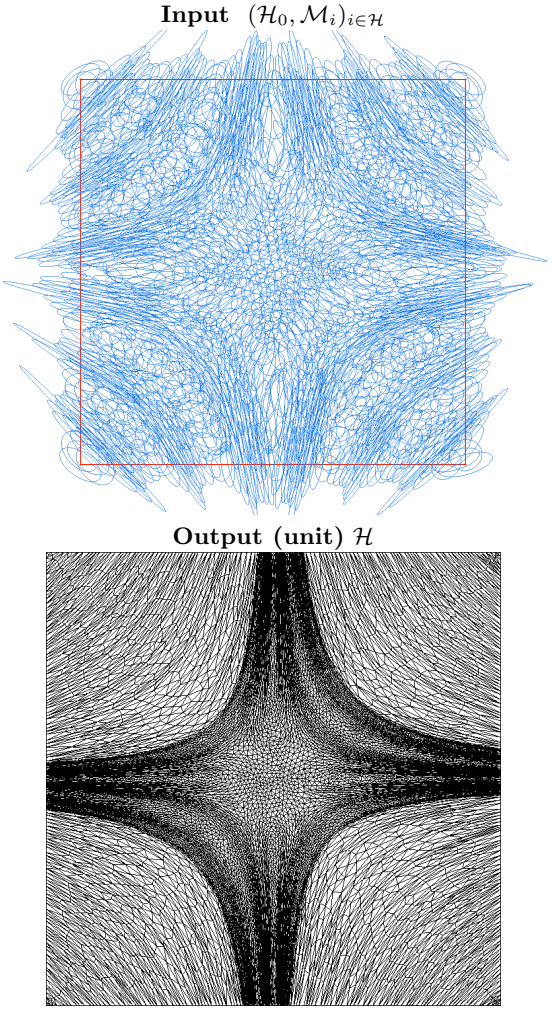

Figure 3: Representation of a metric field as ellipsoids (top) and corresponding unit mesh (bottom)

A metric field is simply a field of metrics, *i.e.* a map-

ping $\mathcal{M} : x \in \mathbb{R}^d \mapsto \mathcal{M}(x)$ onto the set of positive-definite symmetric matrices. In practical applications, metric fields may be assumed infinitely regular. Geometric quantities are generalized by integration of their Euclidean counterparts. For instance, the length of a curve parameterized by γ is given by

$$\ell_{\mathcal{M}}(\gamma) = \int_0^1 \sqrt{{}^t\gamma' \mathcal{M}(\gamma(t))\gamma'(t)}dt.$$

Situations foreign to Euclidean spaces arise from the local nature of distance computations. A shortest path between two points is no longer necessarily straight and can be arbitrarily long in Euclidean space. Likewise, an infinity of shortest paths may exist between two points, for instance taking a metric field that is radially invariant around a straight edge under which a given curved path is shorter than the straight edge.

The link between metric fields and meshes lies in the definition of a unit mesh: a mesh is said to be unit with regards to $\mathcal{M}$ if all element edges are of length 1 in the metric field. In practice, the condition is relaxed so that the lengths lie within $[\frac{1}{\sqrt{2}}, \sqrt{2}]$. The concept of the continuous mesh goes further by establishing a duality between discrete meshes (*i.e.* the usual acception of mesh) and continuous meshes (metric fields they are unit in [17, 18]). Metric fields are a powerful tool for translating error bounds of numerical schemes into geometrical data readily usable by a mesh adaptation algorithm. Indeed, the log-simplex method described in [30] approximates optimal metric fields for which the following element-wise interpolation error estimation holds provided that the mesh is unit for $\mathcal{M}$:

$$\|u - \Pi_k u\|_{L^p(\Omega)} \leq C N^{-\frac{k+1}{3}}$$

where Π_k is the Lagrange interpolation operator of degree k, $C > 0$ a constant independent of k and $\mathcal{M}$ and N is the mesh complexity (general number of vertices). In particular, this result is a generalization of the process described earlier involving the solution Hessian.

In practice, a metric field is discrete, being known at the vertices of the mesh. A continuous field is then built up by interpolation. The so-called log-Euclidean framework introduced in [41] considers the vector space comprised of log-metrics, and shows that the geodesic $\mathcal{M}_\gamma$ — the path that minimizes change — between two metrics $\mathcal{M}_1$ and $\mathcal{M}_2$ is given by

$$\log \mathcal{M}_\gamma(u, v) = u \log \mathcal{M}_1 + v \log \mathcal{M}_2$$

where the exponential and logarithm are the matrix operators acting directly on eigenvalues. This yields a unique metric, contrarily to other schemes such as simultaneous reduction. Furthermore, anisotropy is well preserved and determinant varies monotonically between the two extremities, contrarily to strictly linear metric interpolation. This scheme is generalized to, for example, triangles by noticing that

$$\begin{aligned} &uP_{100} + vP_{010} + wP_{001} \\ &= uP_{100} + (1-u)\left(\frac{v}{1-u}P_{010} + \frac{w}{1-u}P_{001}\right) \end{aligned}$$

and then applying the scheme twice. This yields the following log-Euclidean metric interpolation scheme on the triangle:

$$\log \mathcal{M}(u, v, w) = u \log \mathcal{M}_{100} + v \log \mathcal{M}_{010} + w \log \mathcal{M}_{001}$$

The previous properties on the geodesic are preserved on the triangle. The value of a metric at a given point is then only a matter of which element it belongs to, which can be no more than one in a conforming mesh. The two most common cases where this scheme is used are as follows:

- A new point is inserted into the mesh: it is first localized in a so called *back mesh* which stores the initial metric information and its metric is interpolated from the back element it lies in
- On edge splitting, the point is known to be along the edge: it is faster and reliable enough a first approximation to directly interpolate from the edge extremities.

This process, though more conservative than linear interpolation, still dissipates a metric field into uniformity and isotropy if applied over too many iterations. For this reason, the original mesh and metric field are kept in a so called *back* or *reference mesh*.

1.3 P^1 cavity operator

We now recall the original P^1 cavity operator, about which more in-depth information can be found at [32]. The cavity operator takes as input:

- An arbitrary collection of elements, boundary faces and ridges, *i.e.* the *cavity*,
- A point in space, to be inserted or reinserted

The principle is that cavity entities are removed from the mesh, *i.e.* the cavity is emptied out. These entities may be of pure volume (elements) or lie on the boundary (boundary faces, ridges which discretize CAD lines). The resulting cavity boundary (which may be interior to the mesh) is then starred against the point to (re)insert in a hierarchical fashion: ridges are first reconstructed, then faces, then elements. If this step fails, the operator (in its simplest form) rejects the operation and leaves the mesh as it was before deletion of the cavity. In P^1, this step may only fail if a resulting element:

- is of negative volume (i.e. an edge crosses the cavity boundary or lies completely outside of the cavity)
- does not respect general quality criteria (such as the minimal height of the tetrahedra)

A more sophisticated version of the cavity operator used in practice attempts to circumvent failure by correcting the cavity [42]. For instance, the element outside the cavity against a face that yields a negative volume element is added to the cavity in an attempt to recover validity. This operator is very flexible and, indeed, it is capable of the three most common operations on a mesh, all the while controlling quality and guaranteeing validity at every step:

- Collapse: the cavity is the point's ball, a neighbouring point is reinserted
- Insertion: the cavity is any collection of elements of which one contains the point, typically its local Delaunay cavity, the point itself is inserted
- Swap: the cavity is an edge shell, any point on the boundary of the cavity but not on the edge is reinserted.

These are further illustrated by figure 4 with $2D$ examples.

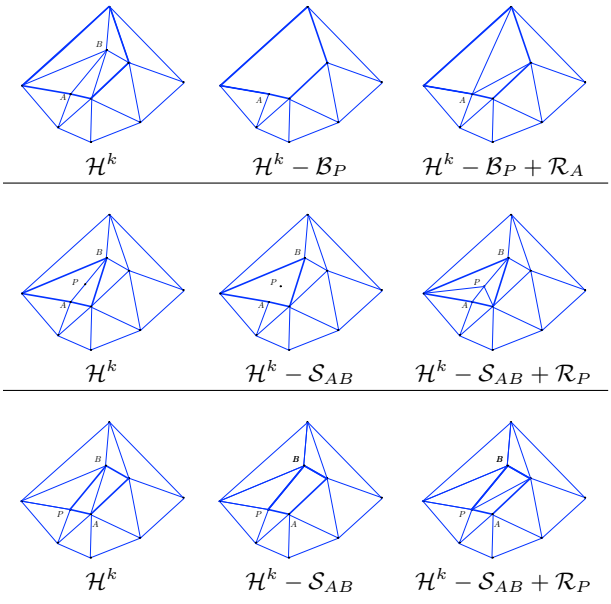

Figure 4: Mesh operations (collapse, insertion, edge swap, from top to bottom) in cavity terms.

1.4 High-order Bézier elements

In this section, we present the building blocks of high-order meshes: Bézier elements. Since the work that follows addresses both two and three dimensional meshes, as well as considers Bézier edges, we introduce general notation directly for all these cases. Detailed proofs for the results announced here can be found, for example, in [43]. The k-simplex of degree n, or P^n simplex, is defined by the mapping

$$F_K : \xi \in \widehat{K}^k \mapsto \sum_{\alpha \in I_n^k} B_\alpha^n(\xi) P_\alpha,$$

where $\widehat{K}^k$ is the reference k-simplex, *i.e.* the set of real $(k+1)$-tuples in $[0,1]$ that sum to 1, I_n^k is the set of integer $(k+1)$-tuples between 0 and n that sum to n, $(P_\alpha)_{\alpha \in I_n^k}$ are the Bézier control nodes of the element and where the Bernstein polynomials of degree n are defined by

$$\forall \alpha \in I_n^k,\ B_\alpha^n(\xi) = \frac{n!}{\prod_{i=0}^k \alpha_i!} \prod_{i=0}^k \xi_i^{\alpha_i}$$

There exists an alternate expression using Lagrange instead of Bézier control nodes. These are defined by

$$\forall \alpha \in I_n^k, P_\alpha^\ell = F_K(\alpha/n),$$

with α/n denoting component-wise division of the $(k+1)$-tuple by n. In the particular case of there being $0 \leq i \leq k$ s.t. $\alpha_i = n$, the Lagrange and Bézier control nodes coincide. In fact, these are the vertices of the k-simplex. Otherwise, they are not equal, but one can deduce the Bézier control nodes from the Lagrange control nodes by the aptly called Lagrange-to-Bézier process (inverting a linear system). Indeed, there is the same count of each and $F_K(\alpha/n)$ is a linear expression of the Bézier control nodes. Introducing the degree n Lagrange basis functions $(\phi_\alpha^n)_{\alpha \in I_n^k}$, the mapping F_K can be rewritten with the help of the Lagrange nodes

$$F_K = \sum_{\alpha \in I_n^k} \phi_\alpha^n P_\alpha^\ell.$$

In both cases, F_K is a convex combination of the either Lagrange or Bézier control nodes at each barycentric coordinate, since

$$\sum_{\alpha \in I_n^k} B_\alpha^n = \sum_{\alpha \in I_n^k} \phi_\alpha^n = 1. \tag{1}$$

Fig. 5 illustrates the indexing scheme and distinction between Lagrange and Bézier control nodes with a P^2 Bézier triangle. Lagrange nodes P_{ijk}^ℓ lie on the element itself, unlike the Bézier nodes P_{ijk}. Bézier control points are the intersection of tangents at the extremities of the given edge for a degree 2 triangle.

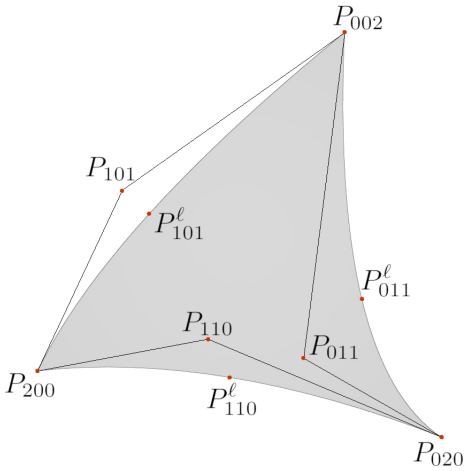

Figure 5: Example P^2 Bézier triangle. Tangents at the vertices as black lines, whose intersections are the Bézier edge nodes.

2. THE HIGH-ORDER LOG-EUCLIDEAN FRAMEWORK AND DIFFERENTIATION OF ANISOTROPIC GEOMETRIC QUANTITIES

As said in the introduction, we seek to minimize curved edge length in the metric field. Therefore, a derivative of P^n edge length with regards to the position of the Bézier control nodes would help us greatly in devising an efficient optimization procedure. Another quantity of wider interest, for instance, is the anisotropic quality measure involving the ratio of the Frobenius norm and determinant of matrix ${}^tJ_K\mathcal{M}J_K$, where J_K is the Jacobian matrix of the reference to physical element mapping [44]. Likewise, derivatives with regards to control node position would be of interest in devising efficient mesh smoothing or untangling algorithms for anisotropic meshes.

In all practical cases, only a discrete metric field is known, which is given at the vertices of the mesh. One extends it to the domain of interest (P^n edge or tetrahedron in the above cases) by a metric interpolation scheme. The principle of this scheme is to localize a given point in the so-called back mesh — essentially, the input mesh left unchanged to preserve metrics from interpolation-induced dissipation — and to interpolate the metric using the log-Euclidean scheme [41] on the back element found. This currently used log-Euclidean scheme (hereafter linear log-Euclidean scheme) deals only with P^1 elements and, thus, cannot extend a metric field on P^n tetrahedra or edges. When considering back-to-back high-order adaptation, it becomes non-trivial to recover a linear back mesh. Indeed, an arbitrary high-order mesh need not produce a valid linear mesh when stripped of its high-order nodes. And, even if this were not the case, the resulting P^1 representation of the log-metric field would be lacking in fidelity compared to a similar scheme that applied directly on the high-order elements. Indeed, high-order numerical schemes reach the same accuracy as their low-order counterparts on coarser meshes.

For these reasons, we propose an extended log-Euclidean metric interpolation scheme, which applies on elements of arbitrary order. It is a direct generalization of the linear log-Euclidean scheme, meaning the metric fields obtained by both methods coincide in the case of a degenerated high-order element (regularly placed Bézier control nodes). This generalization is, in fact, based on the quantity behind the linear log-Euclidean scheme [45], *i.e.* the Fréchet mean of a weighted set of metrics.

In this section, we begin by presenting this new extension of the log-Euclidean metric interpolation scheme. We then apply this new scheme to the case of P^2 edge length computation, by replacing the metric field under the integral by a proper continuous description. In particular, this allows us to compute derivatives of edge length with regards to control node position with relative ease, all the while taking the variations of the metric field into account. This was *a priori* not immediate given the non-linearity that edge length depends on the metric field along the edge, which in turn depends on its shape and position.

2.1 High-order metric interpolation

Let us begin by recalling how the original linear log-Euclidean metric interpolation scheme has been constructed. This can be found in some detail in the research report [45], whereas applications and numerical results are more deeply explored in the publication [41]. In the log-Euclidean framework, the Fréchet mean deriving from any Euclidean norm on symmetric matrices of N metrics $\mathcal{M}_i$ with weights w_i is given by

$$\mathbb{E}\left(\mathcal{M}_1, ..., \mathcal{M}_N\right) = \exp\left(\sum_{i=1}^{N} w_i \log(\mathcal{M}_i)\right).$$

In non-probabilistic terms, the metric that minimizes a convex combination of metric distances (the so-called metric dispersion) is the exponential of the convex combination of the log-metrics. Or, in simpler words, the metric that is closest to a set of metrics for any matrix norm on the log-metrics is given by the above expression. This motivates the classic log-Euclidean interpolation scheme on simplexes. Indeed, given a point P in triangle $P_{100}, P_{010}, P_{001}$, there exists a single triplet (u, v, w) of barycentric coordinates such that

$$P = uP_{100} + vP_{010} + wP_{001},$$

and that sum to 1. These will be the weights, and the metrics naturally those at the vertices: $\mathcal{M}_{100}, \mathcal{M}_{010}$ and $\mathcal{M}_{001}$. The metric at P is then the one that is least dissipated from the vertices of the triangle, *i.e.* the Fréchet mean of associated weights:

$$\mathcal{M}(P) = \exp\left(u \log \mathcal{M}_{100} + v \log \mathcal{M}_{010} + w \log \mathcal{M}_{001}\right).$$

This Fréchet mean constitutes the classic linear log-Euclidean scheme found in [41].

We now turn to the new extension of this scheme to high-order elements. Clearly, the objective is to apply the previous Fréchet mean principle to high-order elements. Following the notations of section 1.4, let now a degree n k-simplex K of control nodes $(P_\alpha)_{\alpha \in I_n^k}$. Let also a point P in K with barycentric coordinates $\xi \in \widehat{K}^k$. P is then given by

$$P = \sum_{\alpha \in I_n^k} B_\alpha^n(\xi) P_\alpha.$$

Due to the convexity property of Bernstein polynomials (eq. (1)), we can choose them as the weights $w = (B_\alpha^n(\xi))_{\alpha \in I_n^k}$ whose sum is therefore 1 as required. Naturally, the chosen metrics are the ones at the control nodes $(\mathcal{M}(P_\alpha))_{\alpha \in I_n^d}$. The resulting Fréchet mean on these log-metrics with these weights is

$$\mathcal{M}(\xi) = \exp\left(\sum_{\alpha \in I_n^d} B_\alpha^n(\xi) \log \mathcal{M}(P_\alpha)\right) \qquad (2)$$

and corresponds to the metric at the vertex of same barycentric coordinates ξ, *i.e.* point P. This does constitute a generalization of the linear log-Euclidean scheme, since evaluating the above with $n = 1$ and $d \in \{1, 2, 3\}$ yields the classic log-Euclidean interpolation scheme on edges, triangles and tetrahedra respectively.

This formulation uses the metric at the Bézier control nodes which are liable to lie far away from the element and, therefore, to contribute with possibly irrelevant (distant) metric information. An analogous result can be derived by expressing the mapping F_K in Lagrange form, yielding the Fréchet mean

$$\mathcal{M}(\xi) = \exp\left(\sum_{\alpha \in I_n^k} \phi_\alpha^n(\xi) \log \mathcal{M}(P_\alpha^\ell)\right), \qquad (3)$$

at point $P = F_K(\xi)$. The best of both worlds (simplicity of the Bernstein formulation, physical significance of the Lagrange nodes lying on the curve) is attained by refactoring this last expression as

$$\mathcal{M}(\xi) = \exp\left(\sum_{\alpha \in I_n^d} B_\alpha^n(\xi) \log \mathcal{M}_\alpha\right), \qquad (4)$$

where the metrics $\mathcal{M}_\alpha$ are the ones obtained through the Lagrange-to-Bézier transformation applied on the log-metrics. Let us clarify this last point through the simple case of a P^2 edge. Its Bézier control nodes are P_{20}, P_{11}, P_{02} and its Lagrange nodes $P_{20}, P_{11}^\ell, P_{02}$ with relationship

$$P_{11}^\ell = \frac{1}{4}\left(P_{20} + 2P_{11} + P_{02}\right).$$

Inverting this yields

$$P_{11} = \frac{1}{2}\left(-P_{20} + 4P_{11}^\ell - P_{02}\right).$$

Denoting $\mathcal{M}_{20}, \mathcal{M}_{02}$ the metrics at the vertices (for which Bézier and Lagrange nodes coincide), the virtual log-metric $\mathcal{M}_{11}$ used in (4) becomes

$$\log \mathcal{M}_{11} = \frac{1}{2}\left(-\log \mathcal{M}_{20} + 4 \log \mathcal{M}(P_{11}^\ell) - \log \mathcal{M}_{02}\right).$$

Starting at degree 3 elements, Lagrange-to-Bézier coefficients require the inversion of a linear system. Finally, let us insist on the fact that formulations "Bézier refactored Lagrange" (4) and "pure Lagrange" (3) are strictly equivalent due to algebraical properties of the Bernstein and Lagrange polynomials coupled with the choice of the virtual metrics $\mathcal{M}_\alpha$. The "Bézier refactored Lagrange" formulation is simply more convenient from an algebraic standpoint, given the ease of expressing Bernstein polynomials from their indices at any degree. However, we stress the fact that formulations "pure Bézier" (2) and "pure Lagrange" (3) are not equivalent, since the metrics used differ and are not related in an arbitrary metric field.

This metric interpolation scheme is well defined on a single element and, therefore, on conforming meshes. Indeed, a point is then either interior to a single element, in which case its metric is uniquely defined, or on a vertex, edge or face of one or several elements. In this case, there is continuity. Indeed, take the case of a tetrahedron's face: it is the domain where the opposite barycentric coordinate vanishes. Therefore, the contributions coming from any metrics not on the face are zero and equal for both elements sharing the face. The same goes for an edge, where two barycentric coordinates vanish. The vertex case is even more immediate, cancelling all but one of the barycentric coordinates in (4) yields exactly the metric at the vertex.

This metric interpolation scheme preserves the desireable qualities of the classic log-Euclidean interpolation scheme on linear elements. For one, eigenvalue positivity is kept. Another interesting property of the classic log-Euclidean interpolation scheme shown in [45] is that the interpolated metric determinant is the geometric mean of the interpoland determinant. More precisely, on the triangle for instance,

$$\begin{aligned}&\det \exp\left(u \log \mathcal{M}_{100} + v \log \mathcal{M}_{010} + w \log \mathcal{M}_{001}\right)\\ &= \det(\mathcal{M}_{100})^u \det(\mathcal{M}_{010})^v \det(\mathcal{M}_{001})^w\end{aligned}$$

This property is fundamental since it establishes a form of volume conservation through lower and upper bounds on the determinant in the general case and even monotony when considering interpolation between two metrics only. This is particularly important since metric determinant is monotonically linked to prescribed volume of elements in metric based mesh adaptation. Moreover, it is shown in [45] that the log-Euclidean mean preserves anisotropy, a property that is therefore kept here. In our case, the determinant of the metric interpolated by scheme (4) becomes

$$\det \mathcal{M}(\xi) = \prod_{\alpha \in I_n^k} \det(\mathcal{M}_\alpha)^{B_\alpha^n(\xi)} \tag{5}$$

Finally, this high-order scheme is consistent with its low-order counterpart. Assuming evenly placed control nodes, a curved element's mapping simplifies and becomes linear. If the metrics at these evenly placed nodes are interpolated with the log-Euclidean scheme from the P^1 vertices of the element, then the argument of the exponential in (4) is actually linear. Indeed, the same arithmetic simplifications occur on the polynomial log-metric field as the ones that turn the polynomial element mapping linear. This means that if a high-order mesh is initialized from a straight mesh with no additional information, the metric field is conserved by the elevation in degree. However, as soon as a control node is moved, this no longer holds.

2.2 Metric field induced volume curvature

In this section, we use the new high-order log-Euclidean metric interpolation to curve volume edges. Riemannian length of the P^2 Bézier edge is computed and differentiated using the log-Euclidean metric field along the edge. It can then be used as the cost function of efficient differentiable optimization algorithms in order to recover natural metric field curvature. The considered metric field $\mathcal{M}$ is discrete, known only at the vertices of the so-called back mesh (linear).

2.2.1 Riemannian P^2 edge length differentiation

We now compute and differentiate P^2 edge length. The chosen variable is Lagrange rather than Bézier control node position, since its metric is of greater physical significance. The edge mapping, with Lagrange nodes $P_{20}, P_{11}^\ell, P_{02}$, is

$$l : t \in [0,1] \mapsto (1-t)(1-2t)P_{20} + 4t(1-t)P_{11}^\ell + t(2t-1)P_{02}.$$

We recall that edge length is given by

$$\ell_{\mathcal{M}}(e) = \int_0^1 \sqrt{{}^t l' \mathcal{M}(l) l'}\, dt$$

and set $d\ell^2 = {}^t l' \mathcal{M}(l) l'$ which depends upon t but also P_{11}^ℓ as a parameter of l. We now differentiate $d\ell^2$ at any t with regards to the Lagrange node P_{11}^ℓ. The main difficulty lies in treating $\mathcal{M}$, which is interpolated from the nodes of the curve using the metrics at Lagrange nodes. But these must first be interpolated from the back mesh. Coming back to $d\ell^2$:

$$\partial_i^L d\ell^2 = {}^t l' \partial_i^L(\mathcal{M}(l)) l' + 2 l' \mathcal{M}(l) \partial_i^L l',$$

where ∂_i^L, L for Lagrange, denotes component-wise derivation with regards to the i-th coordinate of P_{11}^ℓ. The simplest term to treat is $\partial_i^L l'$, since $l'(t) = B + 4(1-2t)P_{11}^\ell$, B independent of P_{11}^ℓ,

$$\partial_i^L l'(t) = 4(1-2t)\vec{e}_i,$$

with $\vec{e}_i$ the i-th base vector. We now turn to the metric. Following eq. (3), we write the metric at $l(t)$ as

$$\mathcal{M}(l(t)) = \exp\left(C + 4t(1-t)\log \mathcal{M}_{11}^\ell\right), \tag{6}$$

where $\mathcal{M}_{11}^\ell = \mathcal{M}(P_{11}^\ell)$ and C is the matrix holding the remaining terms with $\mathcal{M}_{20}$ and $\mathcal{M}_{02}$. The metric at P_{11}^ℓ is now constructed by log-Euclidean interpolation on the back mesh, *i.e.* the input mesh holding unmodified metric information to avoid interpolation-induced dissipation. The process is as follows:

1. Localize P_{11}^ℓ in back mesh, *i.e.* produce reference element K of vertices with metrics $\mathcal{M}_i$ and barycentric coordinates u_i of P_{11}^ℓ in K
2. Interpolate $\mathcal{M}(P_{11}^\ell) \leftarrow \exp\left(\sum u_i \log \mathcal{M}_i\right)$

Injecting this in (6),

$$\mathcal{M}(l) = \exp\left(C + 4t(1-t)\sum u_i \log \mathcal{M}_i\right). \tag{7}$$

Were these terms under the exponential scalar, we would have applied the morphism property and differentiated the expression as a product. Unfortunately, this is a consequence of product commutativity which matrices lack. Therefore, we revert to the definition of the exponential, *i.e.*

$$\mathcal{M}(l) = \sum_{k \geq 0} \frac{1}{k!}\left(C + 4t(1-t)\sum u_i \log \mathcal{M}_i\right)^k.$$

The series' general term is differentiated by recursion:

$$\partial_i T = 4t(1-t)\sum_j \partial_i^L u_j \mathcal{M}_j$$
$$\partial_i (T^{n+1}) = T^n \partial_i T + \partial_i [\, T^n]\, T$$

with $T = \log \mathcal{M}(l)$ for clarity. In practice, the *scaling and squaring* principle from [46] was applied for greater stability and speed. This consists of a first

scaling step where $\lambda = 2^{-m}$, m integer, is found such that the matrix norm of the argument is lesser than 1. The matrix exponential of the scaled matrix λT and its derivative are then accumulated. These matrices are finaly rescalled using relationships $\exp(2Q) = \exp(Q)^2$ and $\partial_i^L \exp(2Q) = 2 \exp Q \partial_i^L (\exp Q)$. Although this might seem cumbersome, it tends to be as fast or faster than first computing eigenvalues (using LAPACK Fortran implementations), computing their exponentials, and recomposing with the eigenvectors. As for the derivative, this also presents the advantage of being an analytical computation.

The issue remains to differentiate the barycentric coordinates u_i of P_{11}^ℓ in the back-mesh element K. We distinguish the $2D$ and $3D$ cases.

2D case Let $K = ABC$ the reference triangle that contains P_{11}^ℓ. Computing the areas of the triangles formed by edges of K with P_{11}^ℓ gives the barycentric coordinates of the Lagrange node in K:

$$\begin{aligned} u &= \frac{\mathcal{A}_u}{\mathcal{A}_K} \quad \text{with } \mathcal{A}_u = \mathcal{A}(P_{11}^\ell BC) \\ v &= \frac{\mathcal{A}_v}{\mathcal{A}_K} \quad \text{with } \mathcal{A}_v = \mathcal{A}(AP_{11}^\ell C) \\ w &= \frac{\mathcal{A}_w}{\mathcal{A}_K} \quad \text{with } \mathcal{A}_w = \mathcal{A}(ABP_{11}^\ell) \\ \text{with} & \quad \mathcal{A}_K = \det\left(A - C \;\; B - C\right). \end{aligned}$$

Simplex sign is preserved through vertex permutations of positive signature. In the case of triangles, vertex cycles are allowed. Using this property yields the symetric expressions

$$\begin{aligned} \mathcal{A}_u &= \det\left(P_{11}^\ell - C \;\; B - C\right) \\ \mathcal{A}_v &= \det\left(P_{11}^\ell - A \;\; C - A\right) \\ \mathcal{A}_w &= \det\left(P_{11}^\ell - B \;\; A - B\right) \end{aligned}$$

which are quite convenient for differentiation and yield

$$\nabla_L \mathcal{A}_u = \begin{pmatrix} y_B - y_C \\ -(x_B - y_C) \end{pmatrix} \quad , \quad \nabla_L \mathcal{A}_v = \begin{pmatrix} y_C - y_A \\ -(x_C - y_A) \end{pmatrix}$$
$$\text{and } \nabla_L \mathcal{A}_w = \begin{pmatrix} y_A - y_B \\ -(x_A - y_B) \end{pmatrix}$$

Where ∇_L denotes derivation with regards to P_{11}^ℓ. Going back to the previous notations, we have $(u_1, u_2, u_3) = (u, v, w)$. Likewise, $\mathcal{M}_1 = \mathcal{M}(A), \mathcal{M}_2 = \mathcal{M}(B)$ and $\mathcal{M}_3 = \mathcal{M}(C)$. Dividing the previous area derivatives by the total area of K yields the $\nabla_L u_j$.

3D case Let $K = ABCD$ the reference triangle that contains P_{11}^ℓ. The barycentric coordinates of the Lagrange node in K are now the subvolumes:

$$\begin{aligned} t &= \frac{\mathcal{V}_t}{\mathcal{V}_K} \quad \text{with } \mathcal{V}_t = \mathcal{V}(P_{11}^\ell BCD) \\ u &= \frac{\mathcal{V}_u}{\mathcal{V}_K} \quad \text{with } \mathcal{V}_u = \mathcal{V}(AP_{11}^\ell CD) \\ v &= \frac{\mathcal{V}_v}{\mathcal{V}_K} \quad \text{with } \mathcal{V}_v = \mathcal{V}(ABP_{11}^\ell D) \\ w &= \frac{\mathcal{V}_w}{\mathcal{V}_K} \quad \text{with } \mathcal{V}_w = \mathcal{V}(ABCP_{11}^\ell) \\ \text{with} & \quad \mathcal{V}_K = \det\left(A - D \;\; B - D \;\; C - D\right). \end{aligned}$$

Full cycles on tetrahedron vertices invert the sign, but face cycles are allowed (same as with triangles). Using $1234 \leftrightarrow 2431 \leftrightarrow 3412 \leftrightarrow 4213$,

$$\begin{aligned} \mathcal{V}_t &= \det\left(P_{11}^\ell - D \quad B - D \quad C - D\right) \\ \mathcal{V}_u &= \det\left(P_{11}^\ell - A \quad D - A \quad C - A\right) \\ \mathcal{V}_v &= \det\left(P_{11}^\ell - B \quad D - B \quad A - B\right) \\ \mathcal{V}_w &= \det\left(P_{11}^\ell - C \quad B - C \quad A - C\right) \end{aligned}$$

which we do not detail further: the variable appears only once in each expression, and the derivatives are simply the first column of the comatrices.

2.2.2 Length minimization and concluding remarks

We have shown in 2.2.1 how the high-order log-Euclidean scheme introduced in 2.1 can be used to define the metric field along a P^2 edge and, thus, its Riemannian length. Likewise, edge length was differentiated with regards to control node position P_{11}^ℓ, allowing the optimization problem

$$\min_{P_{11}^\ell \in \Omega_h} \ell_{\mathcal{M}}(P_{20} P_{11} P_{02})$$

to be solved numerically with some efficiency. This involved essentially three steps:

- differentiate metric components $\mathcal{M}$,
- differentiate metric-independent terms g,
- compose the two.

The first step is common to all geometric quantities that involve the metric field, *e.g.* the element distortion measure involving the ratio of Frobenius norm and determinant of ${}^t J_K \mathcal{M} J_K$, with J_K the Jacobian matrix. The second is unchanged from computing derivatives of Euclidean quantities. Finally, the third is relatively trivial ($g^T \mathcal{M} g$, g a vector) if tedious in some cases ($\det(g^T \mathcal{M} g)$, g a matrix). Finally, in placing ourselves in the case of a discrete metric field with back mesh interpolation, we have guaranteed that the resulting optimization problem is solvable in practice with no additional work.

The problem was stated in the physical varialbe P_{11}^{ℓ}. Another approach would be to use the barycentric coordinates (u_i) of P_{11}^{ℓ} in the back element of vertices A_i it lies in instead of its physical coordinates. Substituting in

$$l'(t) = (4t-3)P_{20} + 4(1-2t)\sum u_j A_j + (4t-1)P_{02}$$

we readily have, in 3D for instance, that

$$\partial_i^L l'(t) = 4(1-2t)(A_i - A_4),$$

where ∂_i^L denotes derivation to the i-th barycentric coordinate. In $2D$, A_3 would stand for A_4. The expression of metric derivatives is greatly simplified:

$$\partial_i^L \log \mathcal{M}_{11}^{\ell} = \log \mathcal{M}(A_i) - \log \mathcal{M}(A_4)$$

This cost function restrains the problem to the back element P_{11}^{ℓ} initially lies in. Therefore, crossing a face requires restarting the optimization algorithm. On the contrary, the formulation in physical coordinates of the previous subsection allows seamless transitions between back elements. Therefore, we have chosen not to use this formalism in the volume. However, it will enable much easier surface optimization, without any projection operators nor constrained optimization.

The choice has been made here of a linear back mesh. This is because, although the P^2 operator presented in the following sections is able to handle P^2 input meshes, the current state of our global adaptation algorithm expects a linear back mesh. A natural extension is thus to replace linear by P^2 back-mesh interpolation, which only changes the derivatives of the barycentric coordinates u_i. This involves inverting a 3×3 matrix.

Finally, the length integral calls for quadrature:

$$\ell_{\mathcal{M}}(e) \simeq \ell_{\mathcal{M}}^n(e) = I_0^1\left(\sqrt{{}^t l' \mathcal{M}(l) l'}\right)$$

with $I_a^b(f)$ any linear operator approximating the integral of f between a and b. Linearity means differentiating it brings no particular difficulty once the derivatives of the integrand are known. In the setting of mesh adaptation, a small number of quadrature points are used, 3 being the most natural choice for an edge with as many nodes. However, for diagnostic purposes, lengths can be computed sporadically with higher precision, especially if the metric field's variations are strong along the edges.

3. CAVITY OPERATOR WITH P^2 CURVATURE RECOVERY

We now describe modifications to the cavity operator used in anisotropic (re)meshing algorithms presented briefly at 1.3 that allow it to handle P^2 elements as input and output. This new P^2 cavity operator recovers volume curvature from the optimization procedure of the previous section. Regarding surface edges, CAD or CAD surrogate projection is used.

3.1 Surface curvature: projection on a P^3 CAD surrogate

The leading error term for surface representation can be controlled with the help of a metric field [33] even in high-order and in a manner that depends only on the surface or curve and not on its parametrization [34]. There remain the lower polynomial degree error terms, *e.g.* quadratic error terms for P^3 meshes, that may be optimized through correct control node placement. We briefly present how surface curvature may be retrieved either by direct projection on the CAD or on a P^3 surrogate instead. Using this background surface mesh, high-order Lagrange nodes may be projected to propose curved edges leading to optimal representation of the geometry.

Computer-Aided Design objects (CADs) provide a continuous description of the domain geometry by means of a collection of (patch,parametrization) couples. Therefore, it is often the case that the geometry is ill-defined, with the following typical obstacles to efficient use in meshing and remeshing:

- Non watertight contact between patches, with high tolerances
- Overlapping/interpenetrating patches
- Singularities/ill-conditioning of the parametrization (pole of the sphere for instance)

In the mesh adaptation loop, the surface mesh is modified to conform to the new metric field. To do so, new vertices are projected on the geometry, of which it is critical to use an accurate depiction rather than the current surface mesh to avoid converging to the wrong geometry. For instance, if the object is a sphere meshed with P^1 elements, refining with vertex projection on the elements will always yield the same polyhedron as in the first iteration. Two solutions come to mind:

- Perform CAD projection on the fly
- On first meshing the surface, produce a second so-called back surface mesh of sufficient accuracy for the expected final mesh complexity that will only be used for surface projection

The first solution is potentially more accurate, but slower and less robust. The second approach is much costlier in memory but cheaper in computations: projection on triangular elements is rather simple. This

is therefore the one we have extended by elevating the degree of the surface back mesh to P^3 elements. This was mainly chosen for the drastic increase in precision afforded by a P^3 surface mesh over the corresponding piece-wise linear surface, and for the simplicity of its construction. The CAD model may even be set aside, with the fundamental forms of the surface being computed by discrete approximations. Moreover, $\mathcal{G}^1$ continuity at the vertices can easily be obtained while still affording degrees of freedom for further error reduction.

The resulting surface is called a *geometrical back mesh* in that it is present throughout the adaptation process without being the basis for the volume mesh. Projection on P^3 elements is not much more difficult than on P^1 elements, albeit requiring some optimization. Fig. 6 illustrates a very simple example of a P^3 surrogate and its ability to accurately render geometry despite coarseness.

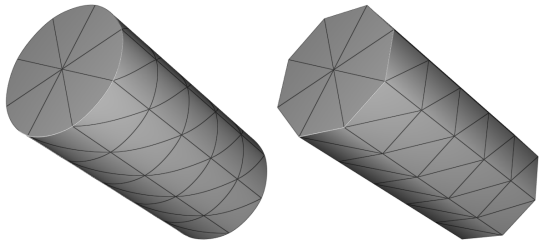

Figure 6: Illustration of P^3 (left) surface approximation on a very coarse cylinder as compared to P^1 (right).

3.2 Cavity-based P^2 correction

To extend the operator to P^2 meshes, two steps are added. First, linear entities are elevated to their high-order counterparts, creating new control nodes. Second, geometric *criteria* are be replaced by high-order ones. For instance, volume positivity is replaced by minimum Jacobian control coefficient positivity as a measure of validity. The difficulty lies in that the new mesh configuration is not unique, but rather indexed on a domain of $\mathbb{R}^{3N}$, N being the number of new interior edges. Indeed, new high-order control points need not be placed so as to form straight edges and such edges may even not guarantee validity given a curved cavity boundary.

The P^2 cavity operator builds upon the previous linear operator by adding an edge curving phase based on the Riemannian edge length optimization algorithm described at 2.2.2. Optimization is carried-out on a per edge basis, keeping the cost strictly linear. The P^2 correction proceeds in two main steps separated by a call to the regular P^1 operator where geometrical checks are disabled. Indeed, a valid P^2 element may very well be an invalid P^1 element , which would elicit a rejection from the P^1 validity checks.

The first main step:

1. Initialize small edge hash table with room for control nodes and tag
2. Loop over cavity boundary faces (3D) or edges (2D)
3. For each, add the 3 (resp. 2) edges against the point to insert in hash table and create corresponding high-order node.
4. Curve edges separately. Inner edges optimized for length. Outter edges projected on geometry if new (in hash table).
5. Loop over new elements. On first invalid element encountered, correct curvature until valid. Exit on failure. Restart loop until no modifications and exit with success.

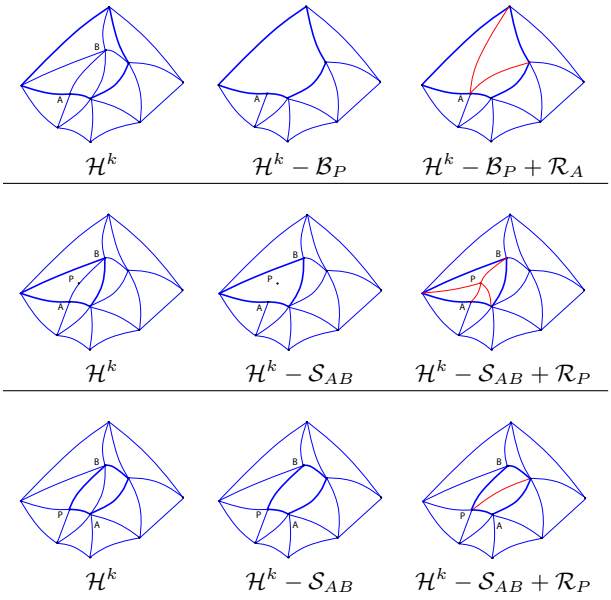

Figure 7: High-order mesh operations (collapse, insertion, edge swap, from top to bottom) in cavity terms. Edges to curve highlighted in red.

There is an exception to step 3): if the point to insert is already a vertex of the mesh and lies on the boundary of the cavity, then the edges that emanate from it must not be recreated (they are not inner edges). They will, however, be curved by projection if lying on the surface, and therefore added to the hash table. The correction in step 4) is possibly the first thing that should be further investigated. For now, a very crude curvature relaxation is used: the correction consists

in applying $P_{11} \leftarrow \theta P_{11} + (1-\theta)M$, where M is the middle of the segment and $0 < \theta < 1$, until a specified maximum number of corrections is reached in which case the cavity is rejected. We add that volume edges are relaxed before surface edges, so as to preserve surface curvature as much as possible. Instead, more sophisticated untangling algorithms could be used such as [13, 47]. Since edge optimization is carried out on a per-edge basis, the added complexity to the P^1 operator is strictly linear. Figure 7 illustrates this starring and curving step. The second main step is a simple update:

1. Loop over new elements provided by P^1 cavity operator
2. For each
   - exterior face, get control nodes from neighbour
   - interior face, look up edge hash table for control nodes
3. Perform secondary updates (if volumes, qualities, etc... are kept)

The resulting P^2 cavity operator presents an overhead to its P^1 counterpart that is proportional to the number of created edges and elements. It also guarantees validity rather than curvature. This contrasts with other methods, for which it is the other way around. A simple global curving algorithm can be devised from this operator, by reinserting each of the mesh's N_P vertices with their ball as the cavity. Assuming that maximum vertex ball size is bounded independently of mesh size, this adds linear overall cost. This hypothesis is common and leads to finite element matrices having $\mathcal{O}(N_P)$ non-zero entries, as a point of comparison with some PDE-based global curving methods. Thus, the resulting algorithm should be asymptotically faster than global methods involving a number of matrix-vector product iterations that increases with mesh size. A comparison in [8] evidences a method with iteration count independent of mesh size (the very method presented in the reference) as well as another PDE-based method [6] with a required iteration count that increases with mesh size. In the latter case, overall complexity is more than linear. In the former, asymptotical cost should remain similar as that of the cavity-based approach.

4. NUMERICAL RESULTS

Implementation was carried out in the metric-based mesh adaptation software AMG/feflo.a [39] and visualization on Vizir4 [48, 49]. Full integration is still in progress, preventing full P^2 adaptation. Moreover, the surface length optimization procedure described at 2.2.2 is not robust enough yet, and simple projection of the middle point is used. As such, the test-cases presented here are the result of a simple global algorithm used for benchmarking:

1. Carry out P^1 adaptation or take as input an adapted P^1 mesh
2. Propose surface curvature by projection on a P^3 CAD surrogate
3. Call the P^2 cavity operator on each existing volume point for reinsertion: parts of the cavity lying on the domain boundary recover their curvature from step 2)

The presentation of numerical examples proceeds in three stages. In the first part, we present how the metric-based P^2 cavity operator is able to naturally propagate surface-induced curvature to the volume. In the second part, boundary layers are curved using the natural mesh metric. Finally, we present real-world examples from computational fluid dynamics with boundary layers and strongly anisotropic metric fields. The meshes used are as follows:

- Meshes 1: the volume between two concentric spheres with boundary-layer variants
- Meshes 2: the NASA Common Research Model (CRM) used on the occasion of the 6th AIAA CFD Drag Prediction Workshop [50]; an adapted mesh was used generated by the remesher AMG/feflo.a [39] and the Wolf [51] solver as well as boundary-layer variants
- Mesh 3: computation on a C608 Low-Boom Flight Demonstrator using the interpolation-based (L^2-norm) error estimates. The C608 is a modified preliminary design of the evolving Lockheed Martin X-59 QueSST for NASA's Low-Boom Flight Demonstrator program.

The quantities of interest are the minimum normalized Jacobian over the entire mesh, edge curvature and length gain. We define curvature as the normalized distance to the middle of the straight edge M_{11}

$$\frac{\| P_{11}^{\ell} - M_{11}\|}{\|P_{20} - P_{02}\|},$$

and length gain as

$$\frac{L_{ini}}{L_{min}} - 1 \tag{8}$$

where L_{min} is the Riemannian length obtained by curving and L_{ini} the original Riemannian length. Normalized Jacobian is defined as the ratio of the Jacobian

determinant of a P^2 element to the absolute value of the Jacobian determinant of corresponding P^1 element (stripped of control nodes). This measures Jacobian determinant of P^2 elements independently of scaling. Anisotropy and, in particular, maximum anisotropy r_{max} will also be considered as a measure of the case's difficulty. Since edge curving is the main addition to the new P^2 cavity opreator, edges curved per second will be counted as a measure of overhead. Finally, all test cases were run on a standard laptop with 32GB RAM and a 6-core processor at 2.9-4.8GHz with 12MB L3 cache.

4.1 Surface induced volume curvature

High-order meshes are most often proposed as a means to reduce surface approximation error. Most methods are based on first projecting boundary control nodes and then modifying the volume mesh to recover validity. To this end, specialized volume operators are used, such as direct optimization operators or PDE based approaches that penalize negative Jacobians and propagate surface curvature to the volume. Such approaches can be found in [6, 7, 8, 9, 10, 12, 13]. With the P^2 cavity operator, validity is always guaranteed but curvature is not.

Our first test case attempts to assess whether the P^2 cavity operator with Riemannian length minimization in the volume is capable of emulating the behaviour of other established methods by recovering a curved boundary. Its ability to do so depends strongly on the chosen input metric. For this task, we construct a metric not from an interpolation error estimate but from the surface approximation metric. The computationnal domain, in this particular case, is the volume between two concentric spheres. At a given point X of the surface with outgoing unit normal $\vec{n}$, the surface metric $\mathcal{M}(X)$ is given by

$$\mathcal{M}(P) = P \begin{pmatrix} \lambda_{min} & 0 & 0 \\ 0 & \lambda_{min} & 0 \\ 0 & 0 & \lambda_{max} \end{pmatrix} {}^t P,$$

with $P = \begin{pmatrix} \vec{\tau}_1 & \vec{\tau}_2 & \vec{n} \end{pmatrix}$ and $(\vec{\tau}_1, \vec{\tau}_2)$ an orthonormal basis of the tangent plane to the surface at X. The eigenvalues λ_{min} and λ_{max} are fixed in advance, with a size ratio of about 10. The higher this anisotropy ratio, the more prominently surface curvature is present in the volume. As for volume vertices, a fixed isotropic metric is set of eigenvalues λ_{min}. The surface metric is then propagated to the volume by a metric gradation scheme such as [52]. This is essentially a smoothing step. The initial P^1 mesh is isotropic and, in particular, not adapted to this chosen metric field.

The resulting meshes are illustrated in Fig. 8. The variations of the input metric field obtained by gradation of the surface-aligned metric allow inner edges to curve slightly, in turn allowing for the curvature of surface elements. Total CPU time was 0.890s (resp. 0.023s) for 23058 (resp. 1552) edges and 3577 (resp. 270) original P^1 vertices on the fine (resp. coarse) mesh. Only 627 (resp. 36) out of 18517 (resp. 1145) tetrahedra had one or more edges moved for validity. Average edge length gain, defined eq. (8), lay at 0.0093 before correction and 0.0032 afterwards in the fine case and at 0.0190 before correction and 0.0178 afterwards in the coarse case. In other words, straight edges were in the order of 1% or 2% longer than optimized edges. Especially in the finer case, a majority of vertices are in the volume with mostly isotropic meshes. This leads to less drastic possible variations of Riemannian edge length, explaining low relative decreases of edge length. The metric field being more anisotropic on most of the coarse mesh would then explain why it curves more strongly.

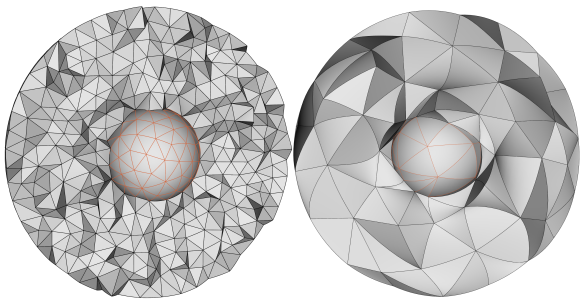

Figure 8: P^2 spheres with valid volume mesh in-between. Surface curvature is naturally present in the volume. Left: finer version. Right: coarser mesh.

4.2 Boundary layer curvature

Curved boundary layer generation is typically treated by specialized algorithms. Through this test-case, we present the ability of the P^2 cavity operator to naturally curve boundary layers that were previously created by the P^1 operator. This is done on the same geometry as the previous test-case. The input metric is chosen as the natural metric of the mesh: for each element, there exists a single metric for which it is unit. This metric is of similar anisotropy as the given element. Metrics at the vertices are then computed by a local scheme. Since boundary layer elements are naturally anisotropic with small sizes along the normal, this produces similar curvature to the previous case without resorting to computing the surface metric.

Boundary layer generation typically proceeds in a frontal manner, starting from the surface. This inevitably leads to the difficulty of managing the regions where fronts meet. In this test-case, we observe this phenomenon where the boundary layer emanating from the outer sphere meets the one coming from the

inner sphere. This particularly unstructured region with size gaps could lead to strong validity constraints.

Fig. 9 illustrates the resulting curved boundary layers. As evidenced by close-ups, the surface is curved and its curvature propagated through most edges of the boundary layer. This is still the case when two boundary layer fronts meet (bottom two figures) despite different coarseness levels. The single boundary-layer case (top two in Fig. 9) contains 12182 edges and 2095 original P^1 vertices. Total CPU time was of 0.26s. Only 276 out of 9120 tetrahedra had to be corrected. This tends to show that despite high anisotropy, a wide range of P^2 element shapes remain valid. The two boundary-layer case (bottom two in Fig. 9) is finer at 63081 edges and 9372 initial vertices. CPU time was 1.56s with 1029 corrections. Edge length gain statistics are summarized at Fig. 10. In the first case, average gain lay at 0.0258 before correction and 0.0192 afterwards and, in the second, at 0.0492 before correction and 0.0445 afterwards. Comparing right and left figures (taking into account the x-axis change) illustrates this gain decrease. This is directly linked to the very simple validity recovery algorithm used. The case with two boundary layers presents greater edge length gains. Since the inner boundary layer is similar to that of the single boundary layer case in size and anisotropy, it is likely that the less anisotropic layer emanating from the outer sphere is responsible for these high gains. This shows that variations in anisotropy foster curvature, while strong anisotropy at the scale of each element leads to more difficult optimization and validity recovery.

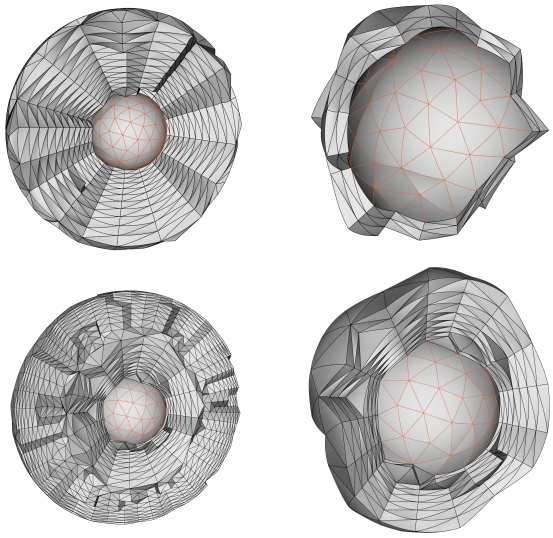

Figure 9: Curved boundary layers with valid volume. Top: single boundary layer. Bottom: two meeting boundary layers and front closure. Left: overviews. Right: close-up on the inner surface.

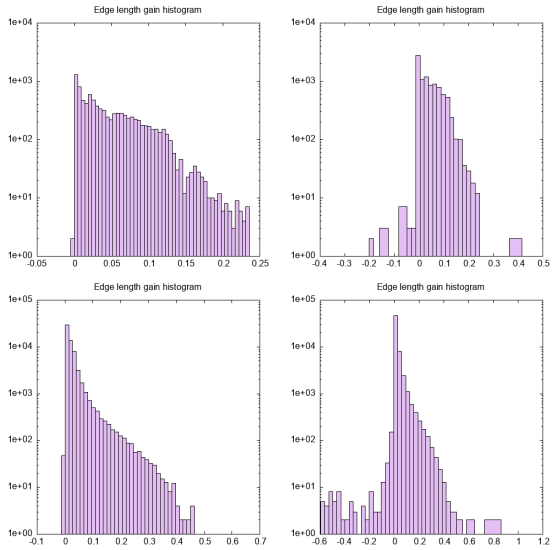

Figure 10: Histograms of edge length gain for the sphere boundary-layer cases. Top: single boundary-layer. Bottom: meeting boundary-layers. Left: before correction. Right: after correction.

4.3 Complex geometry - surface-volume curvature interaction

We now present two large cases in some detail. The first is the CRM from the 6th AIAA CFD Drag Prediction Workshop [50]. The initial adapted mesh had high anisotropy. Two versions were used, one coarser than the other. For each, boundary layers were generated beforehand or not. The second geometry is the C608 Low-Boom Flight Demonstrator adapted to a supersonic flow. Two versions were used, one with about $1M$ elements and the other $20M$. In all cases, the metric used is the natural metric of the mesh.

Meshes and results are summed up in Table 1. The first case (coarse version) is illustrated Fig. 12. At the top, a cut of the wake behind the reactor showing element curvature along the expected directions: edges mostly curl around the centre of the wake. At the bottom, a perpendicular cut of the same region shows that edges along the straight flow remain mostly straight as well. Finally, Fig. 11 illustrates a thin boundary-layer against curved surface, showing similar capacity as on the sphere to propagate surface curvature. This is done when reinserting neighbouring volume points and projecting surface edges. Fig. 13 illustrates results on the second case, the supersonic C608. The very turbulent wake is the most interesting feature when it comes to edge curvature, and we see it occurs as before with the reactor wake despite stronger anisotropy. As in the previous cases, the average length gain is small at about 1%. However, most of the mesh is isotropic

with the far regions of the domain unaffected by the solution close to the geometry. Maximum gains in the dozens illustrate the potential of the process.

The P^2 cavity operator delivers on the promise to keep valid meshes valid, since normalized Jacobians remain positive at all times. Minimum values are as low as 10^{-9} with a tolerance set to 10^{-12}. This illustrates that our placeholder validity recovery algorithm is too conservative and that progress could be made in this direction. Given high-order numerical solvers may set a minimal acceptable value for the Jacobian. This can be enforced by setting the tolerance higher, though it will hamper curvature. On the subject of performance, the edge length optimizer will create a slow-down of the overall cavity operator in the order of $10\times$ independently of mesh size. Indeed, its speed of 40k edges optimized per second is very close to the speed of insertion of the P^1 cavity operator. Reasoning on structured meshes, each vertex collapse and insertion costs an average of 6 edge curvings. This simplistic hypothesis yields, nonetheless, the order of magnitude of the slow-down factor, which remains acceptable given that it is independent of problem size: every new feature of the operator is linear since strictly local (on a per edge basis). Tab. 1 fully illustrates this fact, with meshes ranging from $\approx 500k$ elements to $\approx 2M$ elements at edge optimization speeds within variance of each other.

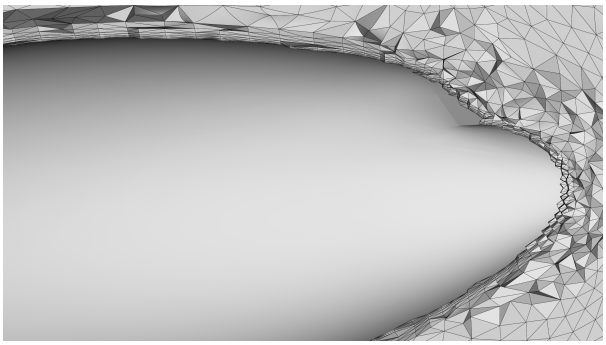

Figure 11: Close-up on the curved surface and curved boundary layer. To the right, nearby volume elements naturally adopt the curvature of the surface.

5. CONCLUSION AND FUTURE WORK

High order mesh generation, and adaptation, is one of the key requirements to validate and reach an extreme level of fidelity in complex flow solutions. If many mesh generation techniques exist to curve a linear mesh for a given geometry, less work exists to generate fully adaptive curved meshes. In this paper, we have introduced a framework for anisotropic curved mesh adaptation. It naturally extends the standard unit-mesh framework used in anisotropic meshes for

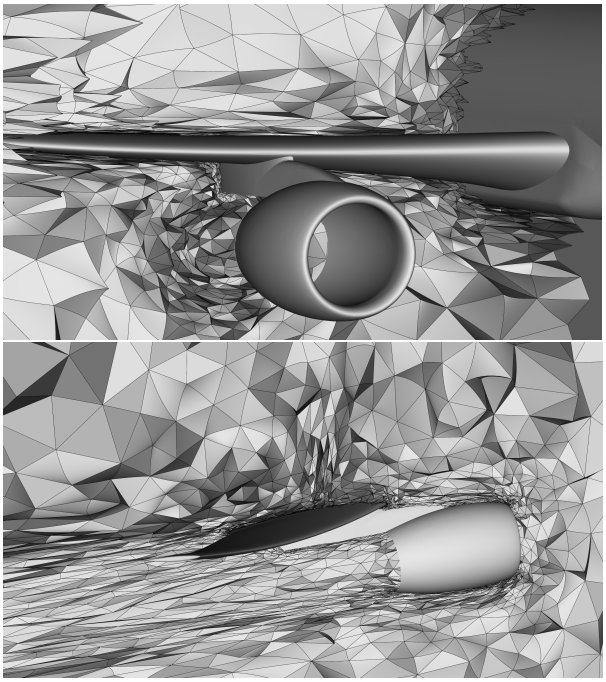

Figure 12: Top: Front-view of the CRM reactor. Elements behind the reactor twist along with the trailing turbulence, creating curved edges consistent with the metric field's variations. Bottom: Side-view of the CRM reactor and wing. Elements are mostly straight from this view: the flow is mostly stretched out but straight in this direction.

linear elements. It is based on the idea that a given metric field provides a natural global curvature information that can be used advantageously to curve the mesh everywhere in the domain. The curvature is then not only provided by the geometry but also by the variation of the metric in sizes and orientations.

This high-order mesh generation framework is based on the extension of the linear cavity operator. The P^2 cavity operator relies on the linear cavity operator for topological checks. The main differences occurs in the handling of validity, where internal edges in the cavity needs to be curved while ensuring positive Jacobian everywhere. This local curving process is based on a local Riemannian P^2 edge length optimization technique. These meshes are equipped with a metric field accounting for geometric approximation or to control some approximation errors on the solution.

To be fully compliant with an adaptive framework, the curved mesh generation process is based on the optimization of the length of edges based on a background mesh and the log-Euclidean interpolation of metrics. We show that for second order meshes, optimal mid-control points can be obtained through local and fast

	Min/Avg/Max nor. Jacobian	Avg/Max gain
(CRM) Coarse	6.5×10^{-6}/1.00/5.6	0.49/17 %
(CRM) Coarse-bl	1.6×10^{-6}/1.00/42	1.6 /82 %
(CRM) Fine	3.1×10^{-6}/1.00/7.3×10^{3}	0.54/73 %
(CRM) Fine-bl	1.9×10^{-8}/1.00/230	0.48/66 %
(C608)Coarse	7.8×10^{-7}/1.00/48	0.69/35 %
(C608)Fine	1.8×10^{-9}/1.00/73	0.54/34 %

	# tet.	# pts ini/end	r_{max}	Edge opt. p/s	Total time (except IO)
(CRM) Coarse	475304	85921 / 660355	960	41328 edg/s	12.9s
(CRM) Coarse-bl	967452	167985 / 1316564	2200	40568 edg/s	27.3s
(CRM) Fine	1566755	263266 / 2134534	3100	41038 edg/s	42.6s
(CRM) Fine-bl	1896348	338367 / 2614330	6100	42570 edg/s	50.5s
(C608)Coarse	925184	165233 / 1276149	1400	42024 edg/s	24.9s
(C608)Fine	19297489	3271359 /25943687	6200	40821 edg/s	548s (9min)

Table 1: Curvature and performance summary for Case1

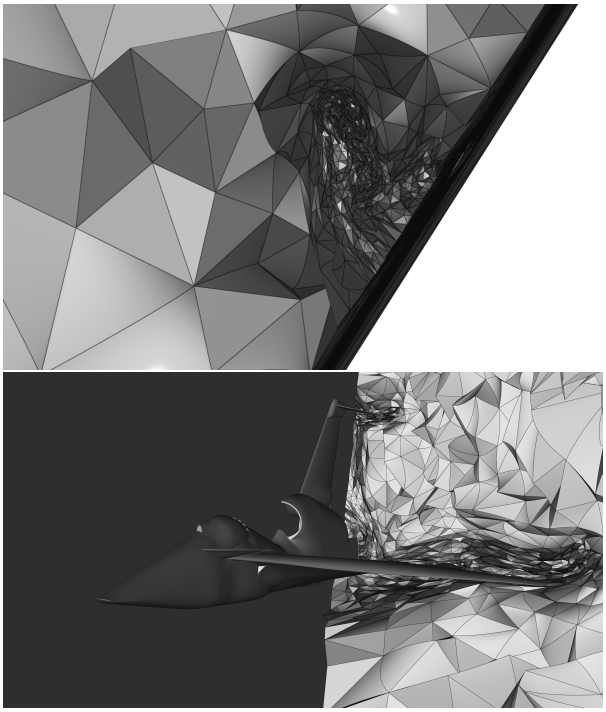

Figure 13: Turbulent wake of the supersonic aircraft (front and rear view). Edge curvature is visible curling along with the vortex.

optimization. The points are then re-inserted within the cavity-based framework.

Several complex examples are provided. Within minutes ($20M$ elements in 9 minutes), second order meshes are generated. The high-level of anisotropy is handled automatically as the metric field complies with this anisotropy. The process uses CAD geometry or a CAD surrogate geometry to project the surface onto the geometry.

This paper is thus a first step in developing a fully adaptive curved mesh generation process. Future work will be directed at implementing all the remaining mesh modification operators such as insertion, collapse and swaps which can be recast within the high-order cavity framework. An additional step will be to extend the local optimization approach to higher order approximation from P^3 to P^5. Other possibilities include extending the log-Euclidean consistent optimization approach to other quantities such as the anisotropic distortion measure.

ACKNOWLEDGEMENTS

The development of INRIA adapted mesh adaptation AMG/feflo.a is supported by ANR IMPACTS (ANR-18-CE46-0003).

References

[1] Huerta A., Angeloski A., Roca X., Peraire J. "Efficiency of high-order elements for continuous and discontinuous Galerkin methods." *International Journal for numerical methods in Engineering*, vol. 96, no. 9, 529–560, 2013

[2] Vanharen J. *High-order numerical methods for unsteady flows around complex geometries*. Ph.D. Thesis, Université de Toulouse, 2017

[3] Ciarlet P.G., Raviart P.A. "The combined effect of curved boundaries and numerical integration in isoparametric finite element methods." *The mathematical foundations of the finite element method with applications to partial differential equations*, pp. 409–474. Elsevier, 1972

[4] Lenoir M. "Optimal isoparametric finite elements and error estimates for domains involving curved boundaries." *SIAM journal on numerical analysis*, vol. 23, no. 3, 562–580, 1986

[5] Bassi F., Rebay S. "High-order accurate discontinuous finite element solution of the 2D Euler equations." *Journal of computational physics*, vol. 138, no. 2, 251–285, 1997

[6] Persson P.O., Peraire J. "Curved mesh generation and mesh refinement using Lagrangian solid mechanics." *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, p. 949. 2009

[7] Moxey D., Ekelschot D., Keskin Ü., Sherwin S., Peirò J. "High-order curvilinear meshing using a thermo-elastic analogy." *Computer-Aided Design*, vol. 72, 130 – 139, 2016

[8] Fortunato M., Persson P.O. "High-order Unstructured Curved Mesh Generation Using the Winslow Equations." *J. Comput. Phys.*, vol. 307, 1–14, Feb. 2016

[9] Abgrall R., Dobrzynski C., Froehly A. "A method for computing curved meshes via the linear elasticity analogy, application to fluid dynamics problems." *International Journal for Numerical Methods in Fluids*, vol. 76, no. 4, 246–266, 2014

[10] Hartmann R., Leicht T. "Generation of unstructured curvilinear grids and high-order discontinuous Galerkin discretization applied to a 3D high-lift configuration." *International Journal for Numerical Methods in Fluids*, vol. 82, no. 6, 316–333, 2016

[11] Turner M., Peirò J., Moxey D. "A Variational Framework for High-order Mesh Generation." *Procedia Engineering*, vol. 163, no. Supplement C, 340 – 352, 2016. 25th International Meshing Roundtable

[12] Karman S.L., Erwin J.T., Glasby R.S., Stefanski D. "High-Order Mesh Curving Using WCN Mesh Optimization." *46th AIAA Fluid Dynamics Conference*, AIAA AVIATION Forum. American Institute of Aeronautics and Astronautics, 2016

[13] Toulorge T., Geuzaine C., Remacle J.F., Lambrechts J. "Robust untangling of curvilinear meshes." *Journal of Computational Physics*, vol. 254, 8 – 26, 2013

[14] Huang W., Kamenski L., Lang J. "A new anisotropic mesh adaptation method based upon hierarchical a posteriori error estimates." *Journal of Computational Physics*, vol. 229, 2179–2198, 2010

[15] Babuska I., Szabo B.A., Katz I.N. "The p-version of the finite element method." *SIAM journal on numerical analysis*, vol. 18, no. 3, 515–545, 1981

[16] Berger M.J., Oliger J. "Adaptive mesh refinement for hyperbolic partial differential equations." *Journal of computational Physics*, vol. 53, no. 3, 484–512, 1984

[17] Loseille A., Alauzet F. "Continuous mesh framework part I: well-posed continuous interpolation error." *SIAM Journal on Numerical Analysis*, vol. 49, no. 1, 38–60, 2011

[18] Loseille A., Alauzet F. "Continuous mesh framework part II: validations and applications." *SIAM Journal on Numerical Analysis*, vol. 49, no. 1, 61–86, 2011

[19] Vallet M.G. *Génération de maillages éléments finis anisotropes et adaptatifs.* Ph.D. Thesis, Université Pierre et Marie Curie - Paris VI, 1992

[20] Loseille A. *Anisotropic 3D hessian-based multiscale and adjoint-based mesh adaptation for Computational fluid dynamics: Application to high fidelity sonic boom prediction.* Ph.D. thesis

[21] Alauzet F. *Adaptation de maillage anisotrope en trois dimensions. Application aux simulations instationnaires en Mécanique des Fluides.* Ph.D. Thesis, Université de Montpellier, Oct. 2003

[22] Dobrzynski C. *3D anisotropic mesh adaptation and application for air-conditionning.* Ph.D. Thesis, Université Pierre et Marie Curie - Paris VI, Nov. 2005

[23] Menier V. *Numerical methods and mesh adaptation for reliable RANS simulations.* Ph.D. Thesis, Université Pierre et Marie Curie - Paris VI, Nov. 2015

[24] Frazza L. *3D anisotropic mesh adaptation for Reynolds Averaged Navier-Stokes simulations.* Ph.D. Thesis, Sorbonne Université , UPMC, Dec. 2018

[25] Vanharen J., Feuillet R., Alauzet F. "Mesh adaptation for fluid-structure interaction problems." *AIAA Fluid.* AIAAP 2018-3244, Atlanta, GA, USA, Jun 2018

[26] Chaillat S., Groth S., Loseille A. "Metric-based anisotropic mesh adaptation for 3D acoustic boundary element methods ." *Journal of Computational Physics*, vol. 372, 473–499, 11 2018

[27] Borouchaki H., Grosges T., Barchiesi D. "Improved 3D adaptive remeshing scheme applied in high electromagnetic field gradient computation."

Finite Elements in Analysis and Design, vol. 46, no. 1, 84 – 95, 2010

[28] Amari T., Canou A., Aly J.J., Delyon F., Alauzet F. "Magnetic cage and rope as the key for solar eruptions." *Nature*, vol. 554, 211–215, 2018

[29] Borouchaki H., Laug P., Cherouat A., Saanouni K. "Adaptive remeshing in large plastic strain with damage." *International Journal for Numerical Methods in Engineering*, vol. 63, no. 1, 1–36, 2005

[30] "Very High Order Anisotropic Metric-Based Mesh Adaptation in 3D." *Procedia Engineering*, vol. 163, 353 – 365, 2016. 25th International Meshing Roundtable

[31] "Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations." *Journal of Computational Physics*, vol. 229, no. 8, 2866 – 2897, 2010

[32] Loseille A. "Recent Improvements on Cavity-Based Operators for RANS Mesh Adaptation." *2018 AIAA Aerospace Sciences Meeting*, p. 0922. 2018

[33] FREY P. "About Surface Remeshing." *Proceedings of the 9th International Meshing Roundtable. Sandia National Laboratories, 2000*, 2000

[34] Feuillet R., Coulaud O., Loseille A. "Anisotropic Error Estimate for High-order Parametric Surface Mesh Generation."

[35] Botti L. "Influence of Reference-to-Physical Frame Mappings on Approximation Properties of Discontinuous Piecewise Polynomial Spaces." *Journal of Scientific Computing*, vol. 52, 09 2012

[36] Moxey D., Sastry S.P., Kirby R.M. "Interpolation error bounds for curvilinear finite elements and their implications on adaptive mesh refinement." *Journal of Scientific Computing*, vol. 78, no. 2, 1045–1062, 2019

[37] Apel T. *Anisotropic finite elements: local estimates and applications*, vol. 3. Teubner Stuttgart, 1999

[38] Zhang R., Johnen A., Remacle J.F. "Curvilinear mesh adaptation." *International Meshing Roundtable*, pp. 57–69. Springer, 2018

[39] "Chapter 10 - Unstructured Mesh Generation and Adaptation." R. Abgrall, C.W. Shu, editors, *Handbook of Numerical Methods for Hyperbolic Problems*, vol. 18 of *Handbook of Numerical Analysis*, pp. 263 – 302. Elsevier, 2017

[40] George P.L. *Automatic mesh generation: applications to finite element methods*. John Wiley & Sons, Inc., 1992

[41] Arsigny V., Fillard P., Pennec X., Ayache N. "Log-Euclidean metrics for fast and simple calculus on diffusion tensors." *Magnetic Resonance in Medicine*, vol. 56, no. 2, 411–421

[42] George P.L. "Improvements on Delaunay-based three-dimensional automatic mesh generator." *Finite Elements in Analysis and Design*, vol. 25, no. 3-4, 297–317, 1997

[43] Borouchaki H., George P.L. *Meshing, Geometric Modeling and Numerical Simulation 1: Form Functions, Triangulations and Geometric Modeling*. John Wiley & Sons, 2017

[44] Aparicio-Estrems G., Gargallo-Peiró A., Roca X. "Defining a stretching and alignment aware quality measure for linear and curved 2D meshes." *International Meshing Roundtable*, pp. 37–55. Springer, 2018

[45] Arsigny V., Fillard P., Pennec X., Ayache N. "Fast and Simple Computations on Tensors with Log-Euclidean Metrics." Tech. rep., INRIA, 2005

[46] Moler C., Van Loan C. "Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later." *SIAM review*, vol. 45, no. 1, 3–49, 2003

[47] Feuillet R., Loseille A., Alauzet F. "P 2 Mesh Optimization Operators." *International Meshing Roundtable*, pp. 3–21. Springer, 2018

[48] Loseille A., Feuillet R. "Vizir: High-order mesh and solution visualization using OpenGL 4.0 graphic pipeline." *2018 AIAA Aerospace Sciences Meeting*, p. 1174. 2018

[49] "On pixel-exact rendering for high-order mesh and solution." *Journal of Computational Physics*, vol. 424, 109860, 2021

[50] Vassberg J., Dehaan M., Rivers M., Wahls R. "Development of a Common Research Model for Applied CFD Validation Studies." *26th AIAA Applied Aerodynamics Conference*. 2008

[51] Alauzet F., Frazza L. "3D RANS anisotropic mesh adaptation on the high-lift version of NASA's Common Research Model (HL-CRM)." *AIAA Aviation 2019 Forum*, p. 2947. 2019

[52] Alauzet F. "Size gradation control of anisotropic meshes." *Finite Elements in Analysis and Design*, vol. 46, no. 1-2, 181–202, 2010