P2 Cavity Operator and Riemannian Curved Edge Length Optimization: a Path to High-Order Mesh Adaptation

Lucien Rochery, Adrien Loseille

P^2 cavity operator and Riemannian curved edge length optimization: a path to high-order mesh adaptation

Lucien Rochery* and Adrien Loseille†
INRIA, 1 Rue Honoré d'Estienne d'Orves, 91120 Palaiseau, France

We present a new P^2 extension of the P^1 cavity operator used as the basis for topological modification in $3D$ metric based mesh adaptation, with notable success in strongly anisotropic industrial cases of CFD. The P^2 operator inherits the P^1 cavity operator's robustness - mesh validity is guaranteed at all times - and manages to recover a metric field's inherent curvature through a Riemannian edge length optimization algorithm. This generic approach allows it to tackle a variety of problems, which are defined only by the input metric field such as the classic problem of surface approximation - through a geometric error surface metric propagated to the volume - or unit mesh construction such as for interpolation error minimization through high-order L^p error estimates.

Consistence with the log-euclidian metric interpolation scheme used in P^1 adaptation is obtained by a rigorous formulation of the optimization problem. This guarantees full compliance of the operator with the general adaptation process, by accurately measuring Riemannian edge lengths.

Particular stress was put on the performance of the operator because of its central role in anisotropic mesh adaptation. All curving operations are carried out locally: this is in opposition with global approaches, be they optimization or PDE based. The optimization itself is carried out by an inhouse solver tailored to the problem at hand. As a result, the added cost is strictly linear.

Numerical results illustrating the P^2 cavity operator's ability to recover curvature, be it surface curvature extended to boundary layers or metric field induced curvature of the volume, will be presented through cases representative of real-world geometries encountered in CFD. Finally, the operator's ability to handle rather large cases (10M elements) in minutes will be demonstrated.

I. Introduction

High-order numerical methods for the resolution of Partial Derivatives Equations (PDEs) have known a surge in popularity in the recent years due to the increased maturity of methods able to solve a wide range of physical problems with greater precision over cost ratio than schemes based on piecewise linear approximations of the solution [1, 2] and reduced errors other than due to interpolation only [3].

High-order numerical methods are still most often carried out on linear meshes. Despite this, the need for curved meshes is not a recent discovery, dating as far back as the 70s with the theoretical proof that optimal convergence of high-order methods is only accessible with a curved boundary in the case of elliptic problems [4, 5]. In the case of the Navier-Stokes and Euler equations used in aeronautics simulations, physical features are lost when the boundary is left piecewise linear [6] and it might even be necessary to represent the geometry with a higher polynomial degree than the solution [7].

The robust and automatic construction of valid curved meshes remains an open problem. Most existing methods rely on input P^1 meshes that are then elevated to a higher degree, which excludes the emergence of a fully high-order mesh adaptation process. Likewise, the main preoccupation lies in recovering a curved boundary, interior curvature being but a necessary condition to the obtention of validity. Some methods are based on *ad-hoc* PDEs [8–12] or on a variational approach [13]. Others are optimization based [14–16] starting from a possibly invalid mesh with the curved boundary. More recently, a frontal approach following geodesics was illustrated, which constitutes direct P^2 metric-based generation [17].

*Phd Student, lucien.rochery@inria.fr, INRIA, 1 Rue Honoré d'Estienne d'Orves, 91120 Palaiseau, France
†Research scientist, adrien.loseille@inria.fr, INRIA, 1 Rue Honoré d'Estienne d'Orves, 91120 Palaiseau, France, AIAA Member.

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, labeled *p*-adaptation, enriches the polynomial space on a per-element basis and, therefore, requires strong coupling with the solver [18]. The second, commonly called *r*-adaptation, limits mesh operations to vertex moving, thus sticking very close to the constraint on mesh complexity [19]. The third, *h*-adaptation, relies on local modifications of the initial mesh to adhere to prescribed sizes [20–23] by applying the complete 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 [24, 25] and unsteady [26, 27] Euler equations, the steady Navier-Stokes equations in the context of RANS simulation [28, 29], fluid-structure interaction [30], acoustics, electromagnetics, magnetohydrodynamics, solid mechanics[31–34]...

Metric fields are the link between particular error estimates - be they for low-order [22, 23] or high-order methods [35], for the solution of a PDE [25] or a quantity of interest derived from it such as drag or lift [36] - and automatic mesh adaptation. In the case of linear meshes, a metric field locally distorts the measure or distance such that, when the mesh adaptation algorithm has constructed an uniform mesh in the induced Riemannian space, it is strongly anisotropic in the usual Euclidean (physicial) space. As such, anisotropy arrises naturally, without it ever being explicitely sought by the (re)meshing algorithm.

We seek to extend these principles of metric-based P^1 adaptation to high-order meshes. In particular, we expect the meshing process to naturally recover curvature from the variations of the metric field, such as illustrated in Fig. 1, very much like P^1 remeshing recovers anisotropy from local values of the metric field. As such, curvature must be the consequence of a simple geometric property computed in the Riemannian space, like anisotropy is the consequence of unitness in a space where distances are distorted. Therefore, we propose Riemannian edge length minimization (or geodesic seeking as in [17]) as the driver for metric field curvature recovery.

The metric field's own intrinsic curvature may derive from any error estimate, be it boundary approximation error [37, 38] or an interpolation error estimate. So far, interpolation error estimates on high-order elements are limited to isotropy ([39] in L^2 and [40] in L^1 norms) or require that the curvature of the element be bounded, essentially establishing a range where it may be considered linear [41].

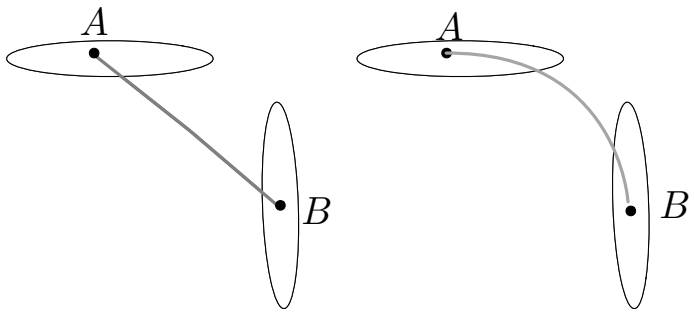

Fig. 1 A metric field's intrinsic curvature recovered on the right through simple edge length minimization: in the metric field, the second edge is shorter than the first.

If genericity with regards to error estimation is achieved through the use of a metric field, robustness and modularity of the general remeshing algorithm may be derived from the use of a single topological operator such as the cavity operator [42–44]. This is the reason why we chose to extend the original P^1 operator to work with P^2 meshes as input and output.

This work deals with a new P^2 cavity operator based, for the volume, on a purely metric-based curving procedure - that remains consistent with log-euclidean metric interpolation - and, for the surface, on CAD or CAD surrogate (typically P^3) projection. The first part provides a short introduction of metric-based adaptation, the classic P^1 cavity operator and high-order Bézier elements. The second section covers the problem of Riemannian edge length minimization. In particular, Riemannian edge length is expressed as a function of edge node positions and metrics. These metrics are then, themselves, expressed as a function of interpolation against the *back* or *reference* mesh which holds original metric information, as is necessary to avoid dissipation of anisotropy through insertion/collapse. Following this work, edge length may be analytically differentiated as a function of control node placement. Finally, we present how the P^2 cavity

operator is built by a high-level approach, by delegating topological work to the existing P^1 operator and using this edge length curving algorithm.

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 delivers on ability to recover metric and surface curvature while maintaining reasonable execution times (20M element mesh in 9min).

A. Metric-based mesh adaptation

Mesh adaptation places mesh generation and PDE solving in a loop instead of a direct execution. Using error estimates on the desired quantity, it produces successive meshes that converge to optimality with regards to precision over cost ratio [45, 46]. As for metric fields, they are a very powerful tool for translating error estimate information into geometrical data readily useable by the remeshing algorithm. In the case of Hessian-based error estimation, the recipe is straightforward: 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, this time using the first fundamental forms of the surface [37]. In both cases, anisotropy appears naturally and is fully kept in the metric. 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 [22, 23]. The metric-based adaptation loop (otherwise illustrated Fig. 2) can therefore 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

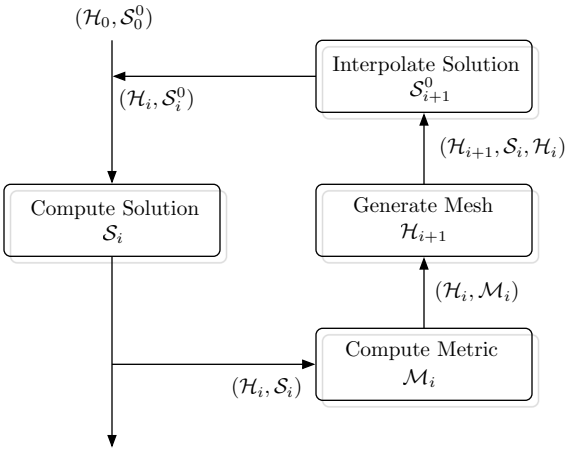

Fig. 2 The mesh adaptation loop

Let us now define more precisely this notion of metric field. We begin with an introduction of the single metric, then move on to the metric field and discrete metric field extended by log-euclidean interpolation. Let d the space dimension (either 2 or 3). Given a positive-definite symmetric matrix $\mathcal{M}$, the bi-linear mapping

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

defines a scalar product on $\mathbb{R}^d$. Such a matrix is said to be the metric, though its induced scalar product would provide an equivalent definition. The positive-definite property also induces that $\mathcal{M}$ has an orthogonal factorization $\mathcal{M} = \mathcal{R}\Lambda\mathcal{R}^T$ where the coefficients $\lambda_1 \leq ... \leq \lambda_d$ of the diagonal matrix Λ are strictly positive. To understand how $\mathcal{M}$ alters norm computations, let us look at its unit ball $\mathcal{B}_{\mathcal{M}}(0, 1)$:

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

Therefore, $\mathcal{B}_{\mathcal{M}}(0, 1) = R\Lambda^{-1/2}\mathcal{B}(0, 1)$, which is to say that the unit ball of $\mathcal{M}$ is a rotation by $\mathcal{R}$ of the unit ball distorted by $\Lambda^{-1/2}$. In other words, $\mathcal{B}_{\mathcal{M}}(0, 1)$ is an ellipsoid of axes along the columns of $\mathcal{R}$ and respective size $\frac{1}{\sqrt{\lambda_i}}$. Conversely,

an ellipsoid uniquely defines such a metric, which justifies the graphical representation of metrics as ellipsoids (as in fig. 3). Furthermore, this shows the equivalence between the Euclidean spaces $(\mathbb{R}^d, (,)_{\mathcal{M}})$ and $(\mathcal{R}\Lambda^{-1/2}\mathbb{R}^d, (,))$.

A metric field is simply a field of metrics, *i.e.* a mapping $\mathcal{M} : x \in \mathbb{R}^d \mapsto \mathcal{M}(x)$ onto the set of positive-definite symmetric matrices. In practical applications, metric fields may be assumed infinitely regular. Usual Euclidian quantities are then defined by integration of their usual counterparts along a path. For instance, given two vertices A and B, the length of the straight segment $[A, B]$ is given by

$$\ell([A,B])_{\mathcal{M}} = \int_0^1 \sqrt{(B-A)^T \mathcal{M}((1-t)A + tB)(B-A)}dt.$$

The fact that distance computations vary depending on the position in space creates situations that do not exist in Euclidean spaces. For instance, the shortest path between two points is no longer necessarily the straight line.

The link between metric fields and meshes lies in the definition of unitness: 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, however, the condition may be 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 the metric fields they are unit in (see [22, 23]). As such, 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 [35] approximates optimal metric fields for which the following element-wise interpolation error estimation holds provided that the mesh is unit for $\mathcal{M}$:

$$\|\Pi_k u - u\|_{L^p(K)} \leq C_Q |K|^{\frac{k+1}{3}} \|\mathcal{M}\|^{\frac{k+1}{2}}_{L^p(K)} \tag{1}$$

where Π_k is the Lagrange interpolation operator of degree k on a tetrahedron and C_Q a constant independent of K and $\mathcal{M}$.

In practice, a metric field is only known at the vertices of a discrete mesh. A continuous field is then built up by interpolation. The interpolation scheme that best conserves mesh anisotropy as well as guarantees eigenvalue positivity is the log-euclidian scheme [47]. Let the point-metric couples $(X_i, \mathcal{M}_i)$, the log-euclidian interpolated metric at any convex combination of the vertices is given by

$$\mathcal{M}\left(\sum t_i X_i\right) = \exp\left(\sum t_i \log \mathcal{M}_i\right), \tag{2}$$

where the exponential and logarithm are the matrix operators that act directly on the eigenvalues. In practice, 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 as illustrated in Fig. **??**. For this reason, the original mesh and metric field are kept in a so called *back* or *reference mesh*.

B. High-order Bézier elements

In this section, we present the building blocks of high-order meshes: Bézier elements. Since the work that follows concerns itself with both two- and three-dimensional simplex meshes, the following defines the P^n tetrahedron and triangle where $n \in \mathbb{N}^*$ is a polynomial degree. The P^n curve is first introduced (illustrated Fig. 4).

P^n segment. Let $\widehat{K}_n^1 = \{0 \leq (i,j) \leq n,\ i + j = n\}$ and the points $(P_{ij})_{(i,j)\in\widehat{K}_n^1}$ of the plane or volume. The points P_{n0} and P_{0n} may be called the P^1 nodes of the Bézier segment K. Indeed, they are defined in the $n = 1$ case as well as lie at the extremities of the curved segment. The other vertices are called the Bézier control nodes of K.

The P^n Bézier segment K is defined by the mapping:

$$F_K : (u,v) \in \widehat{K} \mapsto \sum_{(i,j)\in\widehat{K}_n^1} \frac{n!}{i!j!} u^i v^j P_{ij} \tag{3}$$

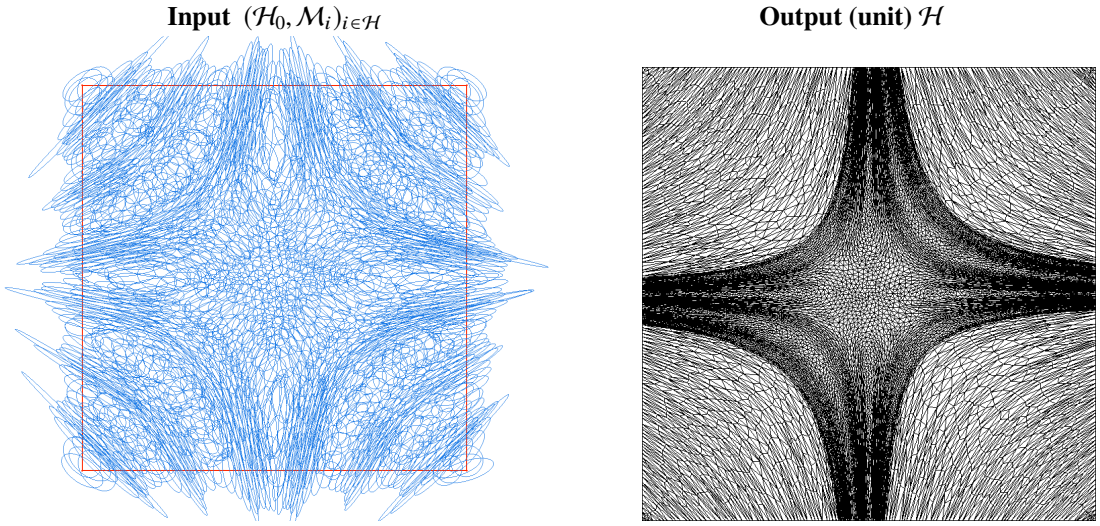

Fig. 3 Representation of a metric field as ellipsoids (left) and corresponding unit mesh (right)

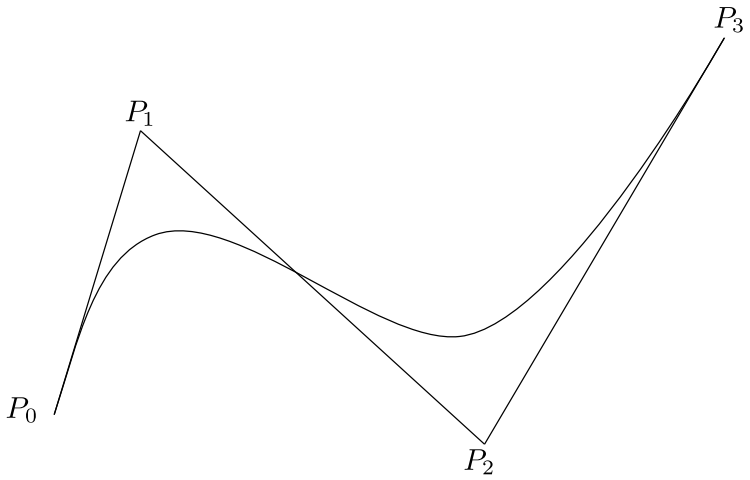

Fig. 4 A Bézier curve of degree 3.

where $\widehat{K}$ denotes the reference segment $\{(u,v) \in [0,1]^2, u+v=1\}$.

The case where the vertices P_{jk} belong to $\mathbb{R}$ is not considered since it would only be encountered with $1D$ numerical methods. Instead, P^n edges will be encountered as the edges of high-order triangles or tetrahedra. As such, the question of edge validity does not arise, since the Jacobian determinant is not defined. Furthermore, the case at hand is that of $n \in \{2,3\}$, which yields the mappings:

$$F_K : (u,v) \mapsto \begin{cases} u^2 P_{20} + 2uvP_{11} + v^2 P_{02} & \text{if } n=2 \\ u^3 P_{30} + 3u^2 v P_{21} + 3uv^2 P_{12} + v^3 P_{03} & \text{if } n=3 \end{cases} \tag{4}$$

An equivalent description is obtained by replacing the Bézier control nodes by their Lagrange counterparts P^ℓ_{ij} defined by

$$P^\ell_{ij} = F_K(i/n, j/n),$$

which lie on the segment unlike the former (with the exception of the vertices all being aligned). The Bézier formulation, however, offers greater geometrical clarity: for instance, the tangents at the extremities are given by

$$\begin{aligned} \partial_u(u \mapsto F_K(u,1-u)).(1) &= n(P_{n,0} - P_{n-1,1}) \\ \partial_u(u \mapsto F_K(u,1-u)).(0) &= n(P_{1,n-1} - P_{0,n}) \end{aligned} \tag{5}$$

whereas the expression derived from a Lagrange formulation would be more unwieldy.

P^n triangle. Let $\widehat{K}_n^2 = \{0 \leq (i,j,k) \leq n,\ i+j+k=n\}$ and the points $(P_{ijk})_{(i,j,k)\in\widehat{K}_n^2}$ of the plane. As with the P^n segment, $P_{n00}, P_{0n0}, P_{00n}$ are the P^1 vertices of K and the others its Bézier control nodes. The P^n Bézier triangle K is defined by the mapping:

$$F_K : (u,v,w) \in \widehat{K} \mapsto \sum_{(i,j,k)\in\widehat{K}_n^2} \frac{n!}{i!j!k!} u^i v^j w^k P_{ijk} \tag{6}$$

where $\widehat{K} \subset \mathbb{R}^2$ is a reference triangle assimilated to its three barycentric coordinates. Let us seize the opportunity to define the Bernstein polynoms of degree n:

$$B_{ijk}^n = \frac{n!}{i!j!k!} u^i v^j w^k$$

Much like the P^n segment, the mapping may be reformulated in terms of the Lagrange nodes which, as before, lie on the element unlike the Bézier control nodes. More generally, the retriction of F_K to an edge of $\widehat{K}$ defines a P^n edge, which yields that the P^n triangle's edges inherit the geometrical properties of the P^n edges seen above.

When K is embedded in $\mathbb{R}^2$, F_K has a square Jacobian matrix which therefore admits a determinant J_K. The finite element convergence proof requires that the mapping be invertible [48] which is a trivial condition when dealing with linear elements (a geometrical restatement is that no element must be flat) but induces the somewhat more complex necessary condition that J_K stay of constant sign for $n > 1$. By convention, $J_K > 0$ is expected.

J_K is a polynom of degree $2(n-1)$ [49] which can be expressed as a combination of the Bernstein polynoms of its degree (see [50] for practical computations over a variety of elements), the weights of which are called J_K's control coefficients.

The sign of the control coefficients does not directly yield the sign of the Jacobian in the element: it is possible for an edge control coefficient to be negative while $J_K > 0$ all over K. On the other hand, if all control coefficients are positive, the Jacobian is guaranteed positive all over the element. To overcome false negatives, a subdivision procedure may be carried out until all coefficients are strictly positive or a tolerance has been reached [51]. The normalized (divided by area of the P^1 element) Jacobian is prefered for its homogeneity allowing for a meaningful lower bound on control coefficients to be set. Fig. 5 illustrates that this Jacobian positivity constraint is not overly restrictive.

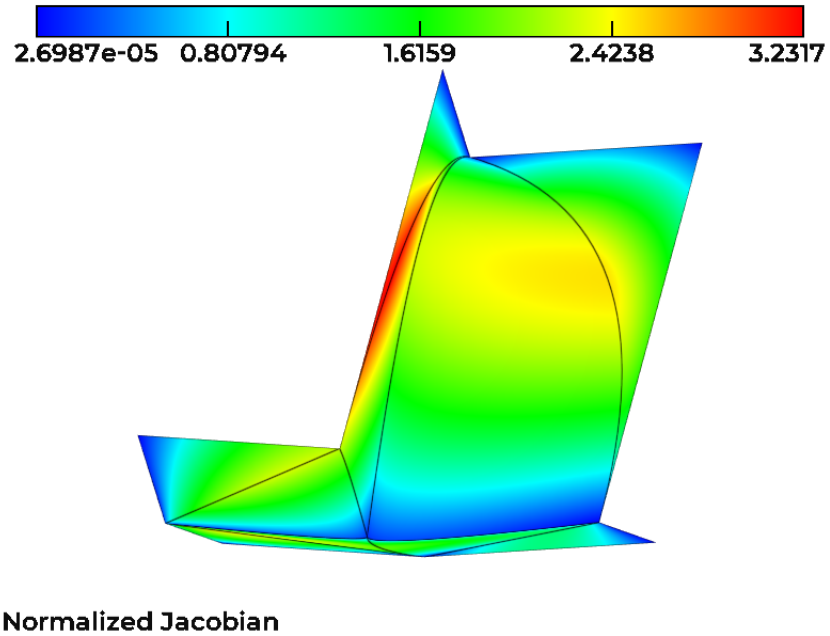

Fig. 5 An example P^2 mesh with positive normalized Jacobian: a wide range of shapes remains valid.

P^n tetrahedron. The P^n tetrahedron is constructed much in the same fashion. Given $\widehat{K}_n^3 = \{0 \leq (i,j,k,l) \leq n,\ i+j+k+l = n\}$ and the points $(P_{ijkl})_{(i,j,k,l)\in\widehat{K}_n^3}$ of the volume, it is defined by the mapping

$$F_K : (t,u,v,w) \in \widehat{K} \mapsto \sum_{(i,j,k,l)\in\widehat{K}_n^3} \frac{n!}{i!j!k!l!} t^i u^j v^k w^l P_{ijkl}$$

The polynomial degree of its Jacobian determinant is now $3(n-1)$ and the same remarks on the sign with relation to the control coefficients hold.

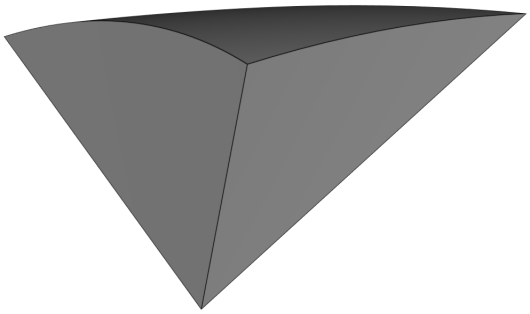

Fig. 6 Illustration of a P^2 tetrahedron

II. Intrinsic curvature recovery

In this section, we show how edge curvature can be deduced from mesh and metric for both surface and volume. When it comes to the volume, edge curvature is the consequence of the metric field's own curvature since edges are optimized for Riemannian length in the metric field. In the case of the surface, edge curvature is obtained through projection on the CAD or a P^3 CAD surrogate.

A. Metric field induced volume curvature

Riemannian length of a degree 2 edge immersed in a metric field is first approximated, then differentiated. Rigorous log-euclidian interpolation is used, and the quadrature of the length integral yields an arbitrarily precise/costly result (including very cheap and imprecise). In doing this, we are able to feed Riemannian edge length to efficient differentiable optimization algorithms in order to recover natural metric field curvature.

We begin by showing that, restricted to a P^2 Bézier curve, the metric field itself can be expressed as a quadratic convex combinations of the metrics at the Bézier nodes (said somewhat abusively, a P^2 edge of the metrics). This enables the derivation of a closed form of the edge length depending only on the Bézier nodes and metrics thereon. Expressing the metric at the h.o. node as an interpolation of metrics at the nodes of the back-mesh element containing it, we are finally able to differentiate length in function of h.o. node position.

1. Setting the problem

In the following, e denotes the P^2 edge of vertices P_{20}, P_{11}, P_{02}. This edge is parametrized by

$$e = \{u^2 P_{20} + 2uvP_{11} + v^2 P_{02}, (u,v) \in [0,1]^2, s.t.\ u+v = 1\},$$

which we rewrite in 1 variable form,

$$e = \{l(t) := (1-t)^2 P_{20} + 2t(1-t)P_{11} + t^2 P_{02}, 0 \leq t \leq 1\},$$

Let a metric field $\mathcal{M}(.)$, we recall that the edge length is given by

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

Since l' is a degree one polynomial and $\mathcal{M}$ the matrix exponential of a polynom of t as will be shown below. If linear metric interpolation had been chosen, $\mathcal{M}$ would have been a degree 2 polynomial. At any rate, the length must be approximated rather than a closed form of the integral sought, so that

$$\ell^n_{\mathcal{M}}(e) = I_0^1\left(\sqrt{l'^T \mathcal{M}(l) l'}\right) \tag{7}$$

stands for $\ell_{\mathcal{M}}(e)$ for practical purposes, where $I_a^b(f)$ is any linear operator that approximates the integral of f between a and b. We chose a degree 1 approximation (trapezoidal rule) but linearity makes it of no real consequence. This approach offers arbitrary cost/precision. 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.

Although it was not made explicit in the writing, l depends of P_{11}. In the following, the dependency of $\mathcal{M} \circ l$ on P_{11} will be derived. This is not straightforward since we want the metric field to follow the log-euclidean interpolation scheme. Therefore, we begin by showing that $\mathcal{M} \circ l$ can be expressed only from the metrics at the three vertices. We then seek an expression of $\mathcal{M}_{11}$ as a function of P_{11} rigorously interpolated from the back mesh (a dual mesh that holds the original metric information).

This expression will then be differentiated with regards to the control node P_{11}, or rather the Lagrange node P^ℓ_{11} that bears greater physical meaning. The main difficulty lies in computing the differential of the term $\mathcal{M}(l)$ because of the chosen log-euclidean interpolation.

2. Metric interpolation on a quadratic edge

Here, we prove the result that, under the log-euclidean interpolation scheme, the metric along the P^2 edge verifies

$$\mathcal{M}(l(u,v)) = \exp\left(u^2 \log \mathcal{M}(P_{20}) + 2uv \log \mathcal{M}(P_{11}) + v^2 \log \mathcal{M}(P_{02})\right)$$

Let us begin by constructing the P^1 triangle K_e of nodes P_{20}, P_{11}, P_{02}. It is clear to see that every point of the P^2 edge lies within the triangle. Indeed, by setting $\lambda_1(u,v) = u^2, \lambda_2(u,v) = 2uv, \lambda_3(u,v) = v^2$, the edge can be rewritten

$$l(u,v) = \lambda_1(u,v)P_{20} + \lambda_2(u,v)P_{11} + \lambda_3(u,v)P_{02}$$

Furthermore, for every $0 \leq u, v, \leq 1$ s.t. $u + v = 1$,

$$\forall\ 1 \leq i \leq 3,\ 0 \leq \lambda_i(u,v) \leq 1 \quad \text{and} \quad \sum_{i=1}^{3} \lambda_i(u,v) = 1$$

Therefore $l(u,v)$ is a convex combination of P_{20}, P_{11}, P_{02} and lies in K_e. In truth, this result is generalized to say that a P^n curve lies within the convex hull of its nodes, the only thing of notice in the P^2 case is that the convex hull is a triangle on which we can perform the log-euclidian interpolation at $P(t)$:

$$\begin{aligned}\mathcal{M}(l(u,v)) &= \exp\left(\lambda_1(u,v) \log \mathcal{M}_{20} + \lambda_2(u,v) \log \mathcal{M}_{11} + \lambda_3(u,v) \log \mathcal{M}_{02}\right) \\ &= \exp\left(u^2 \log \mathcal{M}_{20} + 2uv \log \mathcal{M}_{11} + v^2 \log \mathcal{M}_{02}\right),\end{aligned} \tag{8}$$

which is a generalization to the degree 2 of the log-euclidian interpolation scheme (2) on a straight edge.

The Lagrange node was defined by evaluating the mapping at $(1/2, 1/2)$,

$$P^\ell_{11} = \frac{1}{4}\left(P_{20} + 2P_{11} + P_{02}\right) \tag{9}$$

Evaluating (8) at $(1/2, 1/2)$ or applying the log-euclidian interpolation in K_e on (9),

$$\log \mathcal{M}^\ell_{11} = \frac{1}{4} \log \mathcal{M}_{20} + \frac{1}{2} \log \mathcal{M}_{11} + \frac{1}{4} \log \mathcal{M}_{02}.$$

By reversing this identity, we can rewrite the metric field along the edge using the Lagrange instead of the Bézier node:

$$\mathcal{M}(l(u,v)) = \exp\left((u^2 - uv) \log \mathcal{M}_{20} + 4uv \log \mathcal{M}^\ell_{11} + (v^2 - uv) \log \mathcal{M}_{02}\right), \tag{10}$$

Though this might seem like an arbitrary complication, the Lagrange node holds greater physical meaning than the Bézier node in that it lies on the edge; the latter may even lie outside the mesh where the metric field is undefined. Even without reaching such extremes, when an edge is rather curved, its Bézier and Lagrange nodes can be distant. Defining the metric along the edge using the Bézier node would lead to possibly irrelevant (distant) metric information being used.

3. Back-mesh interpolation

The metric $\mathcal{M}_{11}$ at P_{11} will now be 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 simple:

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

There are cases where not every point of the edge lies in the same reference element. This does not mean that the metric is interpolated at every quadrature point of (7). Doing so would be costly on top of invalidating the previous expression, essential to differentiating edge length with regards to the high order node position.

We seek

$$\text{argmin}_{P^\ell_{11}} \int_0^1 \sqrt{d\ell^2(t, P^\ell_{11})} dt \tag{11}$$

where, substituting (u, v) by $(t, 1-t)$ in (10),

$$d\ell^2(t) = l'(t)^T \exp\left((1-t)(1-2t)\log \mathcal{M}_{20} + 4t(1-t)\log \mathcal{M}^\ell_{11} + t(2t-1)\log \mathcal{M}_{02}\right) l'(t) \tag{12}$$

with an implicit dependency in P^ℓ_{11} found in l'. Let us denote ∇_L the derivation with regards to P^ℓ_{11} for simplicity of notations.

Formally, the expression to differentiate is $g(X) = u^T(X)E(X)u(X)$ where u, X are column vectors of size 2 and E a 2×2 matrix. Using the symetry of E, simple computations yield the derivatives

$$\partial_i g = u^T \partial_i E u + 2u^T E \partial_i u \tag{13}$$

with ∂_i standing for term-wise matrix differentiation. The term $\partial_i u$ is assumed straightforward to compute (as it stands for $\partial_i l'$). As for $\partial_i E$, let us begin by fleshing out the term. We now distinguish between the $2D$ and $3D$ cases, although they are quite similar.

2D case . Let $K = ABC$ the reference triangle that contains P^ℓ_{11}. The barycentric coordinates of the Lagrange node in K are obtained by computing the areas of the triangles formed by edges of K and P^ℓ_{11}:

$$\begin{aligned} u &= \frac{\mathcal{A}_u}{\mathcal{A}_K} \quad \text{with } \mathcal{A}_u = \mathcal{A}(P^\ell_{11}BC) \\ v &= \frac{\mathcal{A}_v}{\mathcal{A}_K} \quad \text{with } \mathcal{A}_v = \mathcal{A}(AP^\ell_{11}C) \\ w &= \frac{\mathcal{A}_w}{\mathcal{A}_K} \quad \text{with } \mathcal{A}_w = \mathcal{A}(ABP^\ell_{11}) \\ \text{and} \quad & \mathcal{A}_K = \det\left(A - C \;\; B - C\right). \end{aligned} \tag{14}$$

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

$$\begin{aligned} \mathcal{A}_u &= \det\left(P^\ell_{11} - C \;\; B - C\right) \\ \mathcal{A}_v &= \det\left(P^\ell_{11} - A \;\; C - A\right) \\ \mathcal{A}_w &= \det\left(P^\ell_{11} - B \;\; A - B\right) \end{aligned} \tag{15}$$

which are quite convenient for differentiation and yield

$$\nabla_L \mathcal{A}_u = \begin{pmatrix} y_B - y_C \\ -(x_B - y_C) \end{pmatrix} \quad , \quad \nabla_L \mathcal{A}_v = \begin{pmatrix} y_C - y_A \\ -(x_C - y_A) \end{pmatrix} \quad , \quad \nabla_L \mathcal{A}_w = \begin{pmatrix} y_A - y_B \\ -(x_A - y_B) \end{pmatrix} \tag{16}$$

3D case . Likewise, let $K = ABCD$ the reference triangle that contains P^ℓ_{11}. The barycentric coordinates of the Lagrange node in K are now the subvolumes:

$$\begin{aligned} t &= \frac{\mathcal{V}_t}{\mathcal{V}_K} \quad \text{with } \mathcal{V}_t = \mathcal{V}(P^\ell_{11}BCD) \\ u &= \frac{\mathcal{V}_u}{\mathcal{V}_K} \quad \text{with } \mathcal{V}_u = \mathcal{V}(AP^\ell_{11}CD) \\ v &= \frac{\mathcal{V}_v}{\mathcal{V}_K} \quad \text{with } \mathcal{V}_v = \mathcal{V}(ABP^\ell_{11}D) \\ w &= \frac{\mathcal{V}_w}{\mathcal{V}_K} \quad \text{with } \mathcal{V}_w = \mathcal{V}(ABCP^\ell_{11}) \\ \text{and} \quad & \mathcal{V}_K = \det(A-D \;\; B-D \;\; C-D)\,. \end{aligned} \tag{17}$$

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

$$\begin{aligned} \mathcal{V}_t &= \det\begin{pmatrix} P^\ell_{11}-D & B-D & C-D \end{pmatrix} \\ \mathcal{V}_u &= \det\begin{pmatrix} P^\ell_{11}-A & D-A & C-A \end{pmatrix} \\ \mathcal{V}_v &= \det\begin{pmatrix} P^\ell_{11}-B & D-B & A-B \end{pmatrix} \\ \mathcal{V}_w &= \det\begin{pmatrix} P^\ell_{11}-C & B-C & A-C \end{pmatrix} \end{aligned} \tag{18}$$

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.

4. Differentiating

From now on, the $2D$ and $3D$ cases are practically identical. Let R_i the log-metrics at the vertices of the back-element K and λ_i the above barycentric coordinates of P^ℓ_{11} in K. The log-euclidean interpolation scheme yields

$$\log \mathcal{M}^\ell_{11} = \sum \lambda_i R_i$$

whose derivatives become

$$\partial_i \log \mathcal{M}^\ell_{11} = \sum \partial_i\lambda_j R_j$$

with $\partial_i\lambda_j$ obtained from the previous volume or area ratios. Using this, the matrix E has the form

$$E(X) = \exp\left(Q + 4t(1-t)\sum \lambda_i(X)R_i\right)$$

where $Q = (1-t)(1-2t)\log\mathcal{M}_{20} + t(2t-1)\log\mathcal{M}_{02}$ is the component independent of P^ℓ_{11}.

Unfortunately, Q and R_i do not commute, which prevents using the exponential's morphism property (*i.e.* $\exp(Q + \sum\lambda_i R_i) \neq \exp(Q)\Pi\exp(\lambda_i R_i)$). Furthermore, the eigenvalues and eigenvectors are not a straightforward expression of P^ℓ_{11}. Therefore, we propose going back to the matrix exponential definition:

$$E(X) = \sum_{k\geq 0}\frac{1}{n!}T(X)^n$$

with $T = Q + 4t(1-t)\sum\lambda_i R_i$. The general term of the series is differentiated by recursion:

$$\partial_i T = 4t(1-t)\sum_j \partial_i v_j R_j$$

$$\partial_i(T^{n+1}) = T^n\partial_i T + \partial_i[\,T^n\,]\,T$$

In practice, T^n and $\partial_i[\,T^n]$ are stored in memory which makes computing the main series the matter of one matrix product and sum per iteration, and the computation of the derivative two matrix products and three sums. A residue on

the matrix norm of the current term to add is computed at every iteration and the procedure stopped when under some tolerance. Even in two dimensions, this algorithm converges slightly faster than computing eigenvalues, exponentiating them, and conjugating by the eigenvector matrix. Furthermore, it is very easy to extend this to higher order derivatives (as was done to compute the Hessian). The "scaling and squaring" principle from [52] was applied for greater stability.

To conclude, one simply replaces $\partial_i v_j$ in the above by the derivatives $\nabla_L \mathcal{A}_\nu / \mathcal{A}_K$ or $\nabla_L \mathcal{V}_\nu / \mathcal{V}_K$.

5. Length minimization and concluding remarks

We have shown here how edge length can be differentiated with regards to high order node position despite the somewhat complex log-euclidean metric interpolation scheme chosen. This formulation has length as a function of P^ℓ_{11} directly, allowing the optimization problem

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

to be solved numerically.

Another approach, which we will apply to the surface, would have the problem formulated in function of the barycentric coordinates (λ_i) of P^ℓ_{11} in the back element of vertices A_i it lies in. Substituting in

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

we readily have that

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

where ∂_i denotes derivation to the i-th barycentric coordinate and in $3D$ (replace A_4 with A_3 otherwise). Since the derivatives of the barycentric coordinates cease to exist, the expression of the metric is greatly simplified:

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

We have chosen not to use this cost function in the volume because it would induce a sequence of element-bound optimization problems. As such, crossing a face requires that the optimization algorithm be restarted, which seemed more costly. However, it will allow us to constrain a control node to a surface element.

B. Surface curvature

We now present how surface curvature may be retrieved either by direct projection on the CAD or on a P^3 surrogate instead. Using this background mesh, high-order Lagrange nodes may be projected to propose curved edges leading to optimal representation of the geometry.

1. Projection on a P^3 CAD surrogate

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 air-tight 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 a surface mesh. It is critical that they be projected from an accurate depiction of the geometry rather than the current surface mesh to avoid producing a seemingly accurate discretization of a false object. 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:

- Project new vertices on the CAD
- 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 projection 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 *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. 7 illustrates a very simple example of a P^3 surrogate and its ability to accurately render geometry despite coarseness.

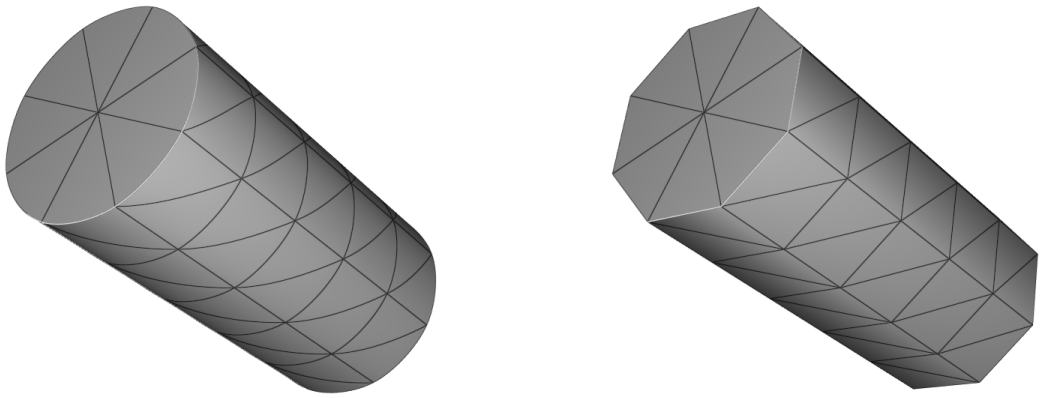

Fig. 7 Illustration of P^3 surface approximation on a very coarse cylinder.

2. Surface curvature recovery

As mentioned in the introduction, high-order meshes appear particularly powerful in their ability to provide a better approximation of the boundary. Using the above P^3 CAD surrogate or full CAD geometry, control points are proposed to produce curved surface elements. As a first approximation, one projects the Lagrange node of the straight edge onto the CAD directly. These control point positions may either be stored in a hash table for access by the P^2 cavity operator or they may be dynamically computed in the curving phase of the operator.

III. P^2 cavity correction

In this section we describe the cavity operator used in Delaunay based meshing algorithms and how it was modified to handle P^2 elements with minimal effort. This new P^2 cavity operator builds upon the previous section, by recovering curvature from the optimization procedure or from P^3 projection.

A. P^1 cavity operator

Let us begin by recalling the original P^1 cavity operator, about which more indepth information can be found at [43, 44]. The cavity operator takes as input:

- A collection of elements, i.e. the *cavity*
- A vertex, be it already inserted in the mesh or not

The cavity is emptied, that is to say the elements are removed from the mesh. The resulting boundary is then starred against the point to (re)insert. 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 on height)

A more sophisticated version of the cavity operator used in practice attempts to circumvent failure by correcting the cavity [53].

This operator is capable of the three most common operations on a mesh, all the while controling quality (such as by selecting Delaunay cavities) 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 8.

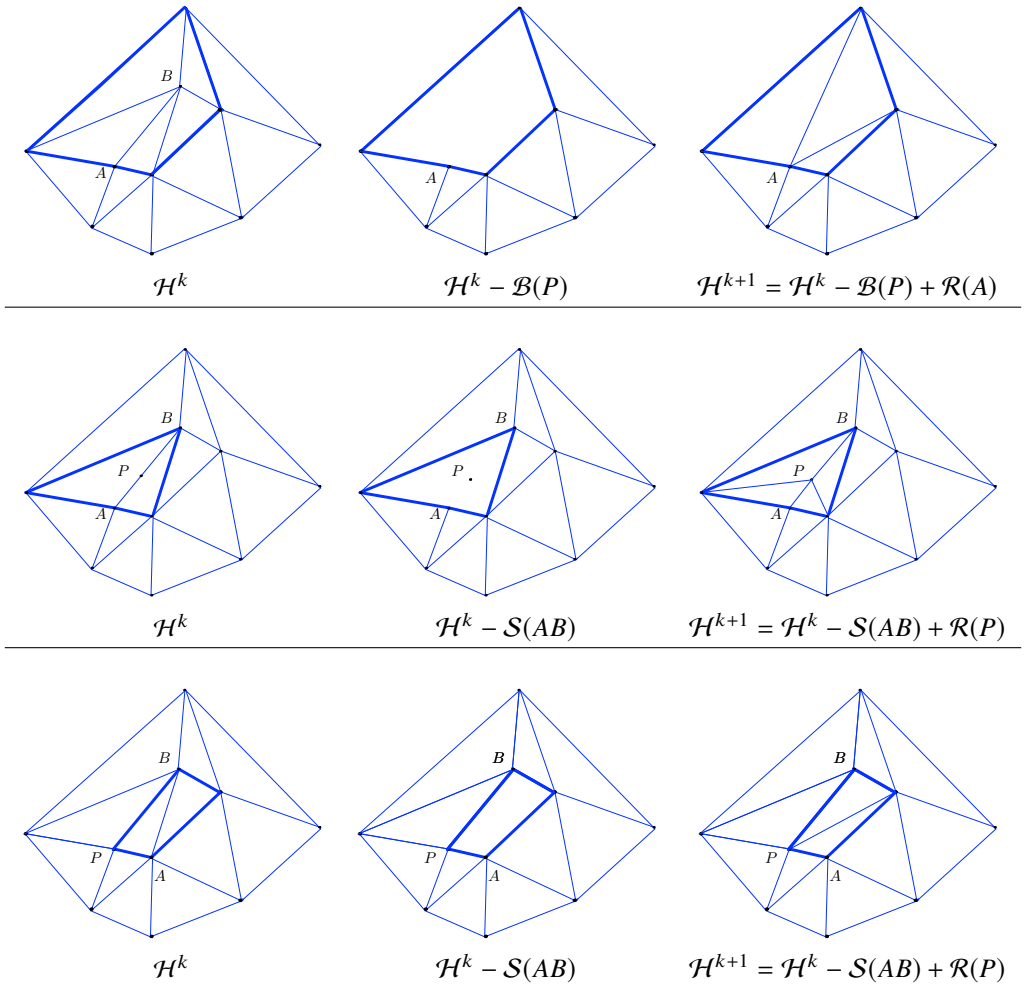

Fig. 8 Mesh operations (collapse, insertion, edge swap, from top to bottom) in cavity terms.

B. Cavity-based P^2 correction

The P^2 cavity operator builds on the previous linear operator by adding an edge curving phase based on the Riemannian edge length optimization algorithm of the previous section. 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 (see fig. 9), which would elicit a rejection from the P^1 validity checks.

The first main step:

1) Initialize small edge hash table with room for a high-order node
2) Loop over initial cavity seeking boundary faces (3D) or edges (2D)
3) For each boundary face, create the edge of vertices (vertex of face, point to insert) if not already in hash table. In particular, create a high-order node and initialize it at straight edge position.

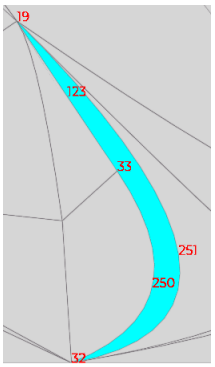

Fig. 9 A valid P^2 element that becomes an invalid P^1 element: the vertex order is $33-32-19$ *i.e.* clockwise (in this mesh, convention has it trigonometric elements are valid)

4) After loop, curve edges separately. New edges are optimized for length. Outer edges are checked for existence in the hash table created when proposing curved boundary edges.
5) Loop over new elements. If all jacobian elements are positive, exit with success. Otherwise, on first invalid element encountered, correct curvature until valid. Restart loop until no modifications.

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). 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. Since edge optimization is carried out on a per-edge basis, the added complexity to the P^1 operator is strictly linear.

The second main step is a simple update:

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

IV. Numerical results

Implementation was carried out in the metric-based mesh adaptation software AMG/feflo.a [54] and the following visualization results obtained on Vizir4 [55]. Full integration is still in progress, but the described operator has already been fully implemented. 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 P^2 cavity operator is able to recover surface-induced curvature in the volume. In the second part, we present how boundary layer curvature may be recovered. Finally, we present real-world examples 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 [56]; an adapted mesh was used generated by the remesher AMG/feflo.a [54] and the Wolf [57] 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, obtained curvature and length gain. We define length gain as

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

where L_{min} is the Riemannian length obtained by curving and L_{ini} the original Riemannian length. Maximum anisotropy ρ_{max} will also be considered as a measure of the case's difficulty. As for performance, its measure will be directly linked with that of the edge curving algorithm. Indeed, it is the main addition to the P^1 operator and counting curved edges over time will give an accurate measure of the overhead of using the P^2 operator instead.

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.

A. Surface induced volume curvature

This test-case illustrates the validity of our approach in dealing with the traditional problem of recovering surface curvature as stated in the introduction. Usually, the boundary is first curved and the mesh is made valid by a global approach: either by solving a PDE through a physical analogy, by solving a quality optimization problem, *etc*... The shortcoming of these approaches is that they usually cannot guarantee a valid volume mesh. With the P^2 cavity operator, the burden is shifted on curvature rather than validity, which is always guaranteed. Moreover, the overhead of edge length optimization is linear in mesh complexity which is often not the case with, for example, PDE based approaches.

The initial mesh is the volume between two concentric spheres. A surface metric is computed along the surface with small sizes in the normal direction. It is then propagated to the volume by a metric gradation scheme [58]. The initial mesh is isotropic and no P^1 adaptation is carried out with this metric field.

This is taken as input by our P^2 cavity reinsertion algorithm. First, a P^3 CAD surrogate is computed for the surface. High-order nodes are then projected on this CAD surrogate but not updated yet. Instead, these proposed positions are kept in a hash table that will be consulted by the cavity operator on the boundary of the input cavity. Each volume point is then re-inserted:

- Each inner edge of the cavity is optimized (these are the volume edges) as usual
- When a cavity boundary edge is found in the surface hash table, the position of the high-order node is set to the one obtained by projection onto the P^3 CAD surrogate.

The operator then proceeds as usual and manages to recover curvature on most edges as evidenced by Fig. 10. The slight variations of the input metric field obtained by gradation of the surface error metric allow inner edges to curve slightly, allowing for the curvature of surface elements.

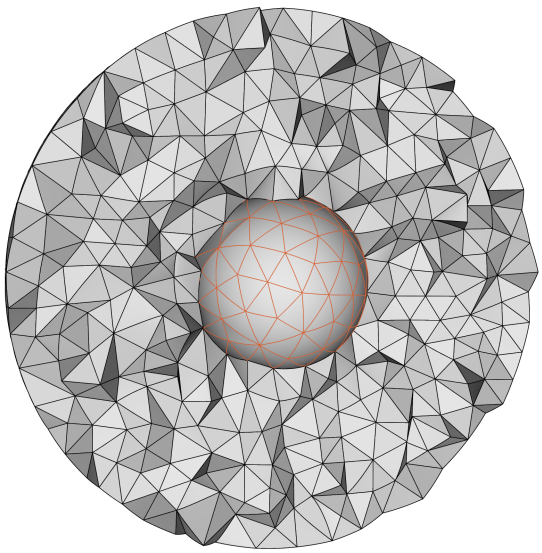

Fig. 10 P^2 spheres with valid volume mesh in-between

B. 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. In this case, the input metric is simply the natural metric of the mesh: for each element, there exists a single metric for which it is unit. 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 should induce similar curvature to the previous case if not for the fact that the elements are anisotropic.

One of the difficulties of boundary layer generation lies in the regions where fronts meet. Here, this phenomenom is present when a boundary layer emmanating 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. 11 illustrates that curved boundary layers could be obtained. The surface is, itself, curved (as evidenced by the close-ups) and its curvature is propagated through most edges of the boundary layer. This is still the case when two boundary layer fronts meet (bottom two figures).

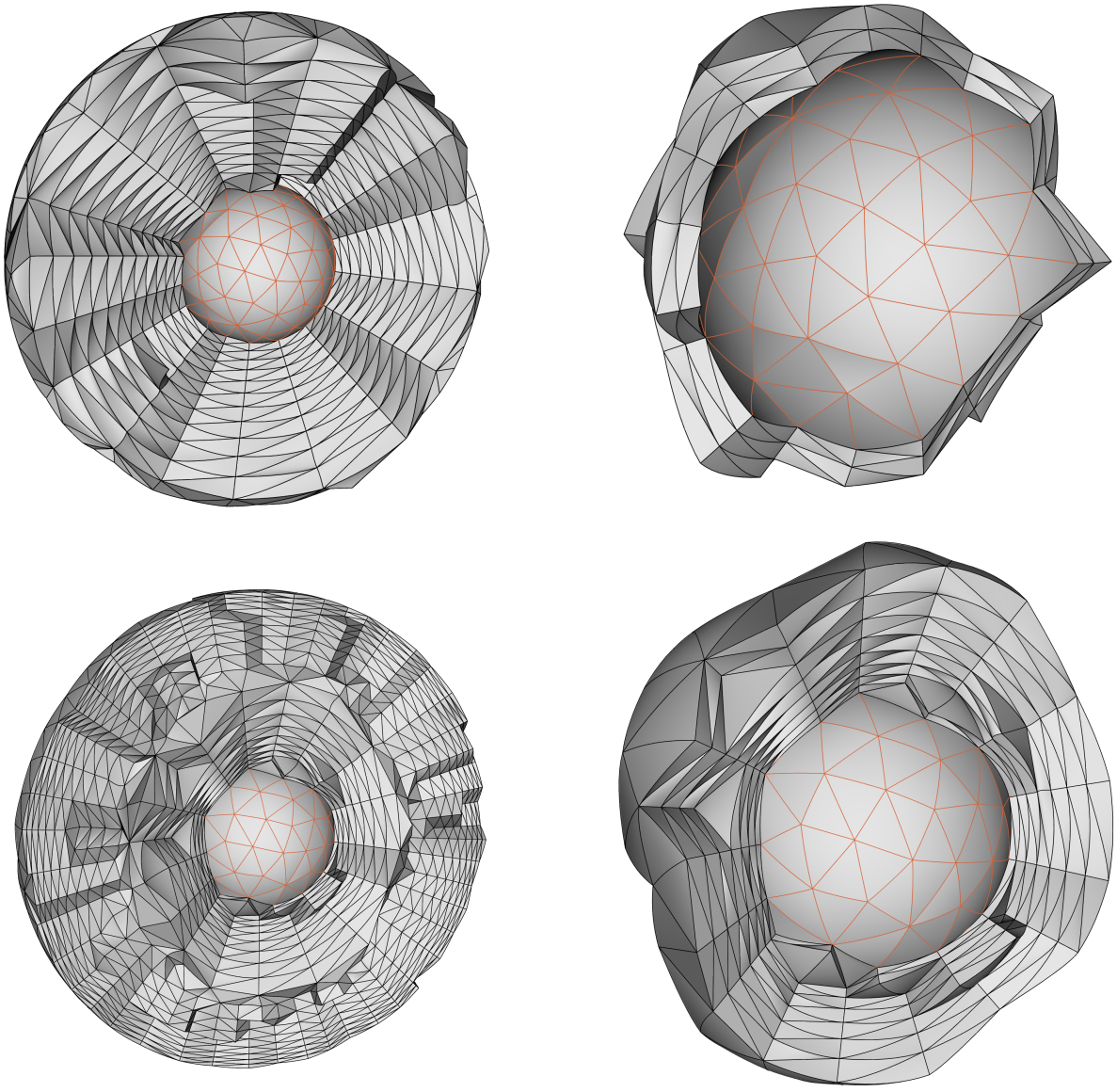

Fig. 11 Curved boundary layers with valid volume. Top: single boundary layer. Bottom: two meeting boundary layers and front closure. Left: overviews. Right: close-up on the inner surface.

C. Complex geometry - surface-volume curvature interaction

We now present two large cases in some detail. The first geometry is the CRM from the 6th AIAA CFD Drag Prediction Workshop [56]. The initial adapted mesh had high anisotropy. Two versions were used, one coarser than the other. For each version, boundary layers were generated beforehand or not. The second geometry is the C608 Low-Boom Flight Demonstrator adapted to a supersonic flow solution. Two versions were used, one with about $1M$ elements and the other close to $20M$. These meshes and the curving execution results are summed up in Tab. 1.

Figures 12, 13, 14 and 15 illustrate the meshes for the first case. The coarse regions shown in Fig. 12 show that consistent curvature accross neighbouring edges is obtained where the metric field is still somewhat anisotropic. In Fig. 13, a cut of the wake of the reactor is seen, which shows element curvature along the expected directions: edges mostly curl around the center of the wake. The perpendicular cut of the same region seen Fig. 14 shows that edges along the straight flow remain mostly straight. Finally, Fig. 15 illustrates a thin boundary-layer against a rather curved part of the surface. We see that boundary-layer elements are curved and that nearby volume elements adopt similar curvature.

Figure 16 illustrates results on the supersonic C608 case. 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.

The average length gain seems small at about 1%. However, one must keep in mind that 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.

Validity is always obtained since the initial mesh is valid. The P^2 cavity operator thus delivers on the promise to keep valid meshes valid. Minimum Jacobians are as low as 10^{-9} but the tolerance was set to 10^{-12}. This illustrates that our placeholder validity recovery algorithm (edge relaxation) is too conservative and that progress could be made in this direction.

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 4.10^4 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 $\sim 500K$ elements to $\sim 2M$ elements at edge optimization speeds within variance of each other.

	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	ρ_{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

Table 1 Curvature and performance summary for Case1

V. 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,

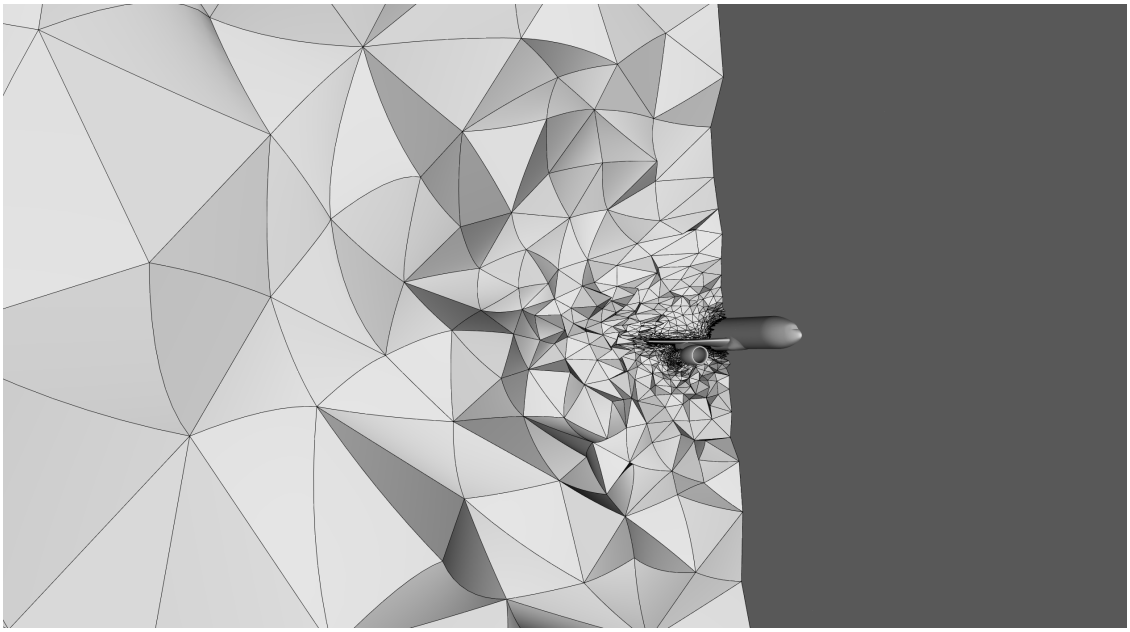

Fig. 12 Cut through the volume in a far view from the curved mesh of the CRM model.

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 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 validity part where internal edges in the cavity needs to be curved while ensuring positive jacobian everywhere. This local curving process is based on several local optimization techniques. 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 optimisation 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 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 optimisation 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 anisotropic quality measures.

Acknowledgments

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

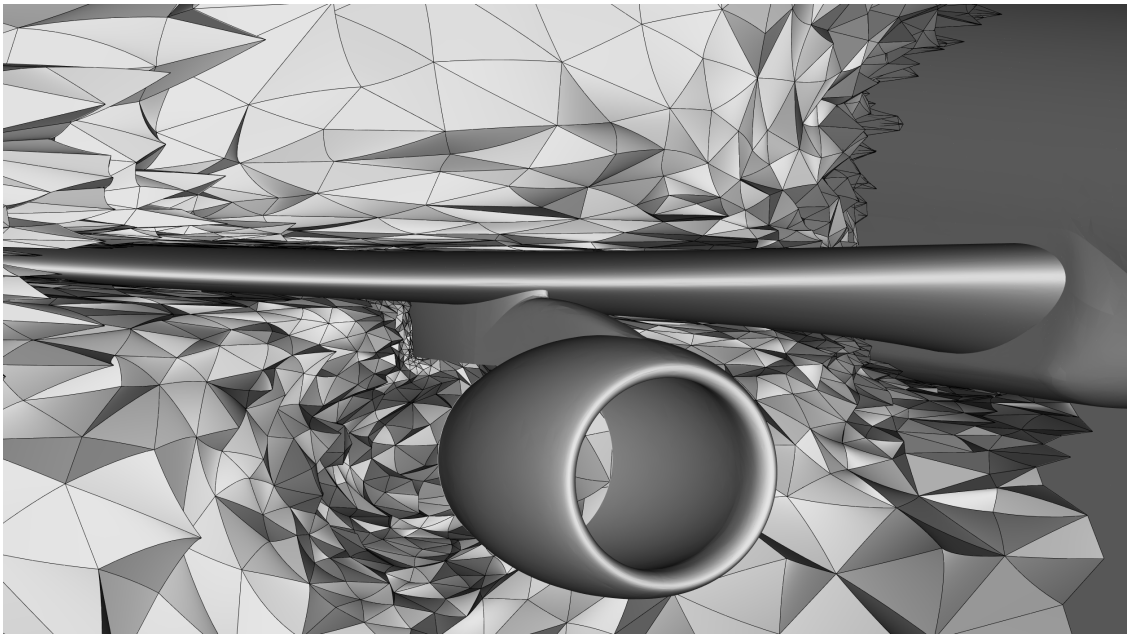

Fig. 13 Front-view of the CRM reactor. Elements behind the reactor twist along with the trailing turbulence, creating curved edges consistent with the metric field's variations.

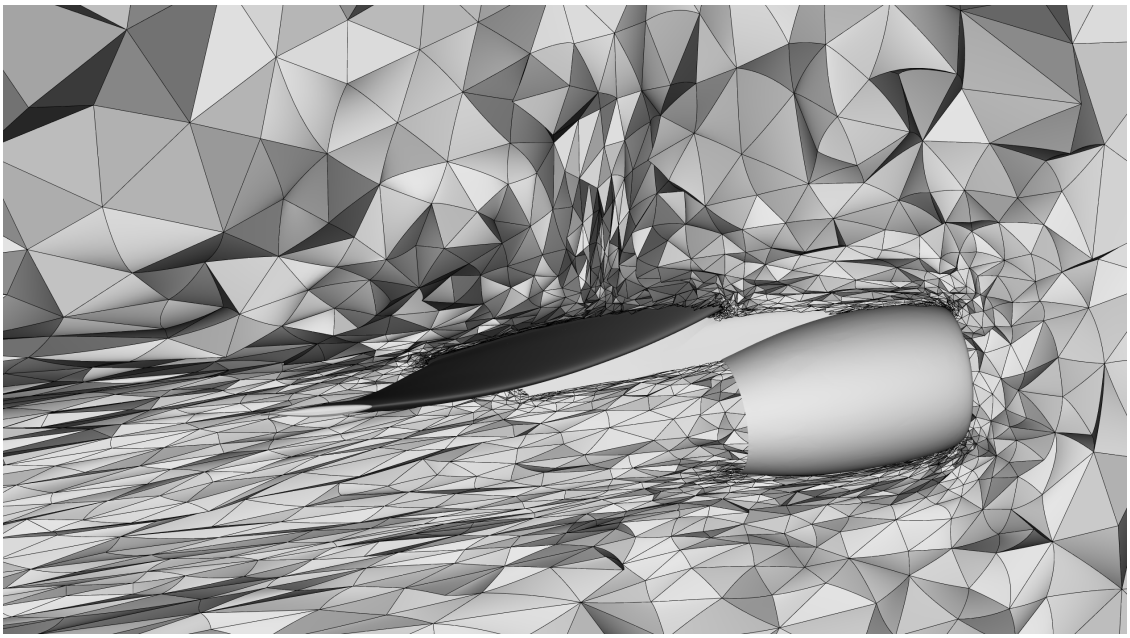

Fig. 14 Side-view of the CRM reactor and wing. Elements are mostly straight from this view: the flow is mostly stretched out but straight in this direction.

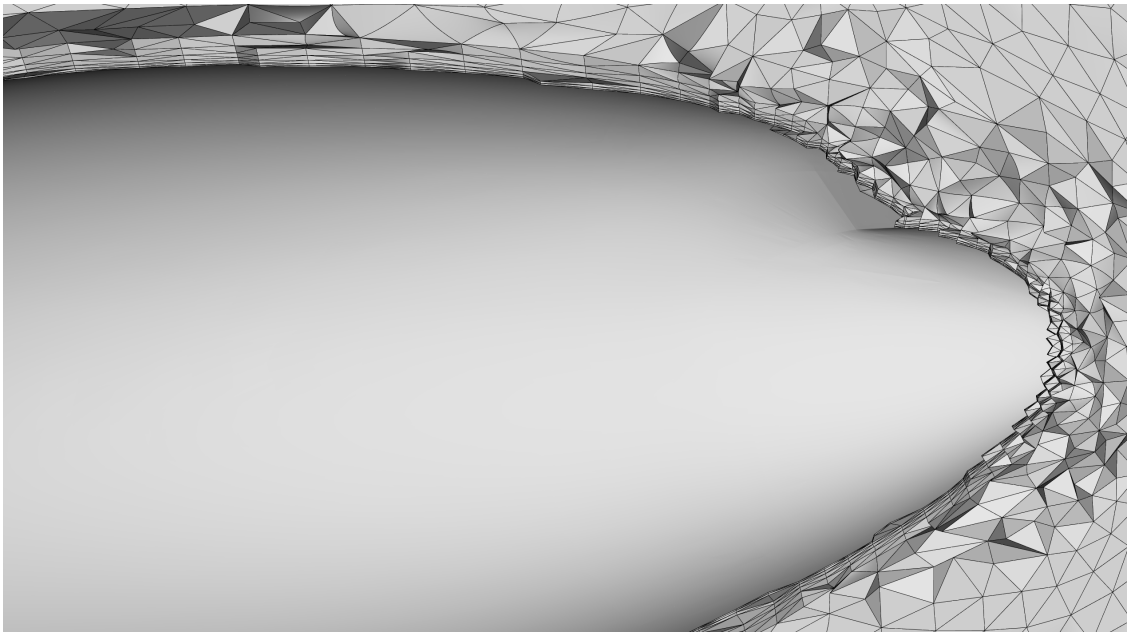

Fig. 15 Close-up on the curved surface and curved boundary layer. To the right, nearby volume elements naturally adopt the curvature of the surface.

Fig. 16 Turbulent wake of the supersonic aircraft (front and rear view). Edge curvature is visible curling along with the vortex.

References

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

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

[3] Vanharen, J., Puigt, G., Vasseur, X., Boussuge, J.-F., and Sagaut, P., "Revisiting the spectral analysis for high-order spectral discontinuous methods," *Journal of Computational Physics*, Vol. 337, 2017, pp. 379 – 402.

[4] Ciarlet, P. G., and 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*, Elsevier, 1972, pp. 409–474.

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

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

[7] Zwanenburg, P., and Nadarajah, S., "On the Necessity of Superparametric Geometry Representation for Discontinuous Galerkin Methods on Domains with Curved Boundaries," *23rd AIAA Computational Fluid Dynamics Conference*, 2017.

[8] Abgrall, R., Dobrzynski, C., and 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, 2014, pp. 246–266.

[9] Persson, P.-O., and 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*, 2009, p. 949.

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

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

[12] Hartmann, R., and 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, 2016, pp. 316–333.

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

[14] Karman, S. L., Erwin, J. T., Glasby, R. S., and 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.

[15] Gargallo-Peiró, A., Roca, X., Peraire, J., and Sarrate, J., "Defining quality measures for mesh optimization on parameterized CAD surfaces," *Proceedings of the 21st International Meshing Roundtable*, Springer, 2013, pp. 85–102.

[16] Toulorge, T., Geuzaine, C., Remacle, J.-F., and Lambrechts, J., "Robust untangling of curvilinear meshes," *Journal of Computational Physics*, Vol. 254, 2013, pp. 8 – 26.

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

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

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

[20] Löhner, R., and Baum, J. D., "Adaptive h-refinement on 3D unstructured grids for transient problems," *International Journal for Numerical Methods in Fluids*, Vol. 14, No. 12, 1992, pp. 1407–1419.

[21] Berger, M. J., and Oliger, J., "Adaptive mesh refinement for hyperbolic partial differential equations," *Journal of computational Physics*, Vol. 53, No. 3, 1984, pp. 484–512.

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

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

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

[25] Loseille, A., "Anisotropic 3D hessian-based multi-scale and adjoint-based mesh adaptation for Computational fluid dynamics: Application to high fidelity sonic boom prediction." Theses, Université Pierre et Marie Curie - Paris VI, Dec. 2008. URL `https://tel.archives-ouvertes.fr/tel-00361961`.

[26] 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.

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

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

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

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

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

[32] Borouchaki, H., Grosges, T., and Barchiesi, D., "Improved 3D adaptive remeshing scheme applied in high electromagnetic field gradient computation," *Finite Elements in Analysis and Design*, Vol. 46, No. 1, 2010, pp. 84 – 95.

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

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

[35] Coulaud, O., and Loseille, A., "Very High Order Anisotropic Metric-Based Mesh Adaptation in 3D," *Procedia Engineering*, Vol. 163, 2016, pp. 353 – 365. https://doi.org/https://doi.org/10.1016/j.proeng.2016.11.071, URL http://www.sciencedirect.com/science/article/pii/S187770581633380X, 25th International Meshing Roundtable.

[36] Loseille, A., Dervieux, A., and Alauzet, F., "Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations," *Journal of Computational Physics*, Vol. 229, No. 8, 2010, pp. 2866 – 2897. https://doi.org/https://doi.org/10.1016/j.jcp.2009.12.021, URL http://www.sciencedirect.com/science/article/pii/S0021999109007001.

[37] Frey, P. J., "About Surface Remeshing," , 2000.

[38] Feuillet, R., Coulaud, O., and Loseille, A., "Anisotropic Error Estimate for High-order Parametric Surface Mesh Generation," *28th International Meshing Roundtable*, Buffalo, NY, United States, 2019. https://doi.org/10.5281/zenodo.3653371, URL https://hal.inria.fr/hal-02345068.

[39] Botti, L., "Influence of Reference-to-Physical Frame Mappings on Approximation Properties of Discontinuous Piecewise Polynomial Spaces," *Journal of Scientific Computing*, Vol. 52, 2012. https://doi.org/10.1007/s10915-011-9566-3.

[40] Moxey, D., Sastry, S. P., and 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, 2019, pp. 1045–1062.

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

[42] Loseille, A., Alauzet, F., and Menier, V., "Unique cavity-based operator and hierarchical domain partitioning for fast parallel generation of anisotropic meshes," *Computer-Aided Design*, Vol. 85, 2017, pp. 53–67.

[43] Loseille, A., and Lohner, R., *Cavity-Based Operators for Mesh Adaptation*, chapter and pages. https://doi.org/10.2514/6.2013-152, URL https://arc.aiaa.org/doi/abs/10.2514/6.2013-152.

[44] Loseille, A., *Recent Improvements on Cavity-Based Operators for RANS Mesh Adaptation*, chapter and pages. https://doi.org/10.2514/6.2018-0922, URL https://arc.aiaa.org/doi/abs/10.2514/6.2018-0922.

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

[46] Castro-Díaz, M., Hecht, F., Mohammadi, B., and Pironneau, O., "Anisotropic unstructured mesh adaption for flow simulations," *International Journal for Numerical Methods in Fluids*, Vol. 25, No. 4, 1997, pp. 475–491.

[47] Arsigny, V., Fillard, P., Pennec, X., and Ayache, N., "Log-Euclidean metrics for fast and simple calculus on diffusion tensors," *Magnetic Resonance in Medicine*, Vol. 56, No. 2, 2006, pp. 411–421. https://doi.org/https://doi.org/10.1002/mrm.20965, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.20965.

[48] Ciarlet, P. G., "Basic error estimates for elliptic problems," 1991.

[49] George, P.-L., Borouchaki, H., and Barral, N., "Geometric validity (positive jacobian) of high-order Lagrange finite elements, theory and practical guidance," *Engineering with computers*, Vol. 32, No. 3, 2016, pp. 405–424.

[50] Feuillet, R., "Embedded and high-order meshes : two alternatives to linear body-fitted meshes," Ph.D. thesis, 2019. URL http://www.theses.fr/2019SACLY010, thèse de doctorat dirigée par Ciarlet, Patrick et Alauzet, Frédéric Mathématiques appliquées Université Paris-Saclay (ComUE) 2019.

[51] George, P., and Borouchaki, H., "Construction of tetrahedral meshes of degree two," *International Journal for Numerical Methods in Engineering*, Vol. 90, No. 9, 2012, pp. 1156–1182. https://doi.org/10.1002/nme.3364, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.3364.

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

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

[54] Loseille, A., "Chapter 10 - Unstructured Mesh Generation and Adaptation," *Handbook of Numerical Methods for Hyperbolic Problems*, Handbook of Numerical Analysis, Vol. 18, edited by R. Abgrall and C.-W. Shu, Elsevier, 2017, pp. 263 – 302. https://doi.org/https://doi.org/10.1016/bs.hna.2016.10.004, URL http://www.sciencedirect.com/science/article/pii/S1570865916300357.

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

[56] Vassberg, J., Dehaan, M., Rivers, M., and Wahls, R., *Development of a Common Research Model for Applied CFD Validation Studies*, chapter and pages. https://doi.org/10.2514/6.2008-6919, URL https://arc.aiaa.org/doi/abs/10.2514/6.2008-6919.

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

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