AIAA SciTech Forum
6-10 January 2025, Orlando, FL
AIAA SCITECH 2025 Forum

10.2514/6.2025-0779

Metris: an Open Source High-Order Metric-Based Remesher

Lucien Rochery*, Mark Chiriac†, Marshall C. Galbraith‡, David L. Darmofal§, Steven R. Allmaras¶
Massachusetts Institute of Technology, Cambridge, MA, 02139

This paper presents Metris, a new piece of open source metric-based mesh adaptation software. The main novelty of this software is its ability to handle high-order meshes, and to apply mesh curvature using metric fields. We present two methods for applying metric-based curvature. Firstly, prior work on closed-form formulas for Bézier node positions to obtain unit P^2 elements is used in conjunction with an algorithm based on Linear Programs to apply curvature while guaranteeing mesh validity. Secondly, differentiable optimization of an anisotropic quality or conformity error can be used, which is generally faster, but performs best both from a performance and unitness standpoint if applied at late stages of curving and adaptation. More generally, the algorithms within Metris are described in some detail.

2D adaptation with parametric geometry provided by a Boundary Representation (BREP) file is currently implemented. Metris can be run either through files on disk, or via memory transfer. Using the latter, a numerical solver was used to illustrate and compare adaptation with Metris and other freely-available remeshers on two test cases. One, where an analytical solution is known, to verify expected convergence is obtained. The other is a transonic flow simulation on an airfoil, with the mesh adapted to minimize an estimated drag error, to compare aerodynamics coefficients obtained with Metris and other freely-available meshers. Faster convergence of these coefficients is observed with Metris, indicating high quality meshes were produced, as corroborated by provided mesh statistics: in this case, final meshes per complexity have at least 99.64% unit length and maximum quality (in range $[1, +\infty[$) no higher than 2.26.

I. Nomenclature

C_L, C_D	=	lift and drag coefficient
Q1/Q2	=	degree 1 or 2 mesh
P1/P2	=	degree 1 or 2 discretization (solver)
$\mathcal{M}$	=	metric field
$(\cdot,\cdot)$ and $\|\cdot\|$	=	scalar product and associated norm
K	=	triangle or tetrahedron
K_0	=	ideal (equilateral) triangle or tetrahedron
$\widehat{K}$	=	barycentric domain
F_K	=	mapping of element K (triangle or tetrahedron)
$\mathcal{J}_K$	=	Jacobian matrix of F_K
J_K	=	determinant of $\mathcal{J}_K$
k	=	topological dimension of an element
d	=	polynomial degree of an element
α	=	multi-index ($(k+1)$-tuple) summing to d e.g. $(2,1,0)$ if $d=3$ and $k=2$.
B_α	=	Berstein polynomial of multi-index α, e.g. B_{110}
N_α	=	Jacobian determinant control coefficient of multi-index α
(r,θ)	=	polar coordinates
h	=	characteristic length scale
t or (u,v)	=	curve or surface parametric coordinates

*PostDoctoral Associate, Department of Aeronautics & Astronautics, AIAA Member
†Undergraduate Researcher, Department of Aeronautics & Astronautics
‡Principal Research Engineer, Department of Aeronautics & Astronautics, AIAA Senior Member
§Professor, Department of Aeronautics & Astronautics, AIAA Fellow
¶Research Engineer, Department of Aeronautics & Astronautics, AIAA Associate Fellow

II. Introduction

In metric-based mesh adaptation, a metric field prescribes element sizes and anisotropic. The remesher is tasked with producing a uniform mesh, under a scalar product (hence norms, angles) modified according to the metric field. Such metric fields are usually produced by minimizing an error estimator, whether a priori [1, 2] or a posteriori [3–6]. Metric fields to minimize L^p-norms of interpolation error have been derived in [7–9], but more general errors such as on quantities of interest in aeronautics (drag, lift) can also be minimized [1, 10, 11], or even boundary approximation error [12, 13]. As a result, metric-based mesh adaptation on linear meshes has been applied successfully to a great range of physical phenomena and error estimators [7, 14–18].

Curved elements for the Finite Element Method (FEM) were considered as early as the 60s [19]. Isoparametric finite elements could be found in textbooks by the end of the 70s [20]. Most common numerical methods have been extended to use curved Bézier meshes since then. The optimal scheme degree for several Discontinuous Galerkin (DG)-like methods is shown theoretically [21] to be greater than one, especially for lower target errors and in 3D. Order 3 methods are shown to provide optimal error over work ratio in an empirical study representing a wide range of complex Computational Fluid Dynamics (CFD) applications [22]. However, these high-order numerical methods require high-order (curved) meshes to achieve their advantageous properties. Physical features may be lost due to piece-wise linear surfaces [23] and optimal convergence may require a curved boundary [24, 25]. In some cases, it might even be necessary to represent the geometry with curved elements of a higher polynomial degree than the solution [26].

Despite this, high-order mesh generation has not yet reached a degree of maturity comparable to linear meshing, and still presents a number of major challenges, both theoretical and practical. Curving boundary elements also requires curving domain-interior elements to guarantee mesh validity, thus meshes with curved boundaries must be curved everywhere. In the worst case, interpolation error for degree q curved elements can lead to order $\lfloor p/q \rfloor + 1$ convergence rates instead of the expected order $p + 1$ [27, 28]. Beyond preserving error convergence rates, these additional element degrees of freedom can be harnessed to further minimize error at fixed computational cost, much like anisotropy is used in adaptation. Super-convergence due to domain-interior element curvature is observed in [29] for boundary approximation, in [30, 31] for L^2 projection error on P^2 bases in 2D, and in [32] for P^2 interpolation error in 3D. Except for boundary approximation in [29], these have been empirical numerical studies.

Developments on metric-based high-order meshing have been as follows. In [33, 34], interpolation error is integrated along mesh edges in 2D, leading to a simplified a priori metric-based error estimate involving curvilinear edge length, which is minimized to curve meshes. Edge length minimization is similarly used in [35] as a curvature driver in 3D. In [36], a metric-based distorsion measure is generalized to high-order elements and used to produce curved meshes in 2D and 3D. This is an algebraic generalization, which implicitly enforces a definition of high-order metric conformity formally $\mathcal{J}_K \sim \mathcal{M}^{-1/2}$, where $\mathcal{J}_K$ is an element mapping Jacobian matrix and $\mathcal{M}$ the metric field. In [37], Mesh Optimization via Error Sampling and Synthesis (MOESS) [3–6] is extended to high-order elements, assuming a mesh-implied metric also formally $\mathcal{J}_K \sim \mathcal{M}^{-1/2}$. This relationship defines unitness as: an element is unit if it is continuously deformed (in a manner determined by $\mathcal{M}$) from an ideal element (equilateral).

In [38], we studied the applicability of such a definition of unitness for high-order meshes. We define that K is unit in $\mathcal{M}$ if there exists a rotation matrix R such that, for all barycentric coordinate ξ, $\mathcal{J}_K(\xi) = \mathcal{M}^{-1/2}(F_K(\xi))R\mathcal{J}_0$. Our proposed framework has since been interpreted as elements defining isometries between reference space with identity metric (hence Euclidean) and the Riemannian physical space with prescribed metric field [31], which is in line with the original idea of generating uniform elements, but with modified geometric computations.

By considering an order $q - 1$ Taylor expansion of $\mathcal{M}^{-1/2} \circ F_K$ about some barycentric coordinate ξ_0, the unit relationship can be verified exactly against an element K of degree q, up to a remainder we show is negligible at mesh convergence. Then, we studied more particularly the case of $q = 2$, and derived exact geometric relationships for K to be unit in the metric field: its underlying Q1 element needs to be unit in $\mathcal{M}(F_K(\xi_0))$, and the control points must obey some very cheap to enforce equations. We also showed, by considering embedded metric fields $(\mathcal{M}_h)_h = (h^{-1/2}\mathcal{M})_h$ that the second derivatives (curvature) of unit elements (regardless of the particular metric $\mathcal{M}$) converge as $O(h^2)$, whereas first derivatives (size) converge as $O(h)$. Hence, elements unit in an embedded sequence of metric field are regular in the sense of [39] and the subpar convergence rates observed in [27, 28] are automatically avoided. Using these results, the unit definition can be relaxed to non-polynomial $\mathcal{M}^{-1/2} \circ F_K$ (i.e. arbitrary metric fields), which if less practical from a geometric standpoint, is much more powerful from the standpoint of error estimation, as it allows to write the element mapping derivatives everywhere as a function of the metric field and its derivatives. We derived a simple interpolation error estimate on curved meshes to illustrate this method.

In this article, we introduce a new piece of open-source mesh adaptation software, Metris, based on the theory introduced within [38]. Metris implements metric-based mesh adaptation — similarly to refine [40], feflo.a [41], avro

[42] or EPIC [43] — and was built to support high-order meshes from its conception. As high-order metric-based adaptation is generalized from linear mesh adaptation, Metris can handle classic linear meshes as well. High-order mesh validity hinges on control coefficients [44], providing rigorous validity guarantees, and tools for enforcing validity, which are detailed in this article. Metric-based curvature is a consequence of the unit mesh definition, and can be applied either through minimizing a conformity error similar to [36], or by solving Linear Programs by an approach introduced in this article, which builds upon prior work dealing with Linear Programs applied to minimum control coefficient maximization [35].

Metris was integrated within the Solution Adaptive Numerical Simulator (SANS) [45] which uses MOESS to drive adaptation. Hence, we provide two numerical applications in this article. The first is adaptation to the L^2 error of L^2 projection of an analytical solution onto Finite Element spaces of degrees 1, 2, and 3. Metris both in Q1 (linear) and Q2 (curved, degree 2) modes is compared to state-of-the-art freely available remeshers refine and avro. The second case is a transonic Reynolds-averaged Navier–Stokes (RANS) adaptation of the ONERA OAT15A airfoil. The final meshes present shocks, boundary-layers and corner singularities illustrated to be well resolved. Metris is compared to refine, and drag and lift coefficients are shown to be slightly better resolved using Metris.

Section III recalls the basic concepts used within Metris: metric-based adaptation, high-order elements, and high-order unit elements. Section IV describes Metris and the algorithms within, as well as the new method based on Linear Programs (LP) to apply optimal Bézier offsets (unit curvature) while keeping mesh validity. Finally, the numerical results are reported in Section V.

III. High-order unit meshes

In this Section, we present the theoretical framework in which Metris operates. Subsection III.A outlines the continuous mesh framework in the case of linear meshes. Subsection III.B introduces the notations used in this work for high-order elements as well as their basic definitions. Finally, Subsection III.C summarizes the generalization of the unit mesh framework to high-order meshes introduced in [38].

A. The continuous mesh framework

We refer to the synthesizing work [2, 46] or book [47] for a more thorough exposition.

Metrics A metric is a symmetric positive definite matrix $\mathcal{M}$ that induces a scalar product $\langle \cdot, \cdot \rangle_{\mathcal{M}}$ and its derived Euclidean norm $\| \cdot \|_{\mathcal{M}}$. $\mathcal{M}$ decomposes as $\mathcal{M} = PDP^T$, where the columns of P form an orthonormal basis of $\mathbb{R}^n$ and D is a diagonal matrix of the positive eigenvalues of $\mathcal{M}$. The power function $\mathcal{M}^s = PD^sP^T$ is uniquely defined for all s and coincides with the usual power for integer s. Of interest is the power $-1/2$ of $\mathcal{M}$ as the unit ball $\mathcal{B}_{\mathcal{M}}(0, 1)$ for the norm $\| \cdot \|_{\mathcal{M}}$ verifies

$$\mathcal{B}_{\mathcal{M}}(0, 1) = \mathcal{M}^{-1/2}\mathcal{B}(0, 1) \tag{1}$$

with $\mathcal{B}(0, 1)$ the unit ball for the usual norm $\| \cdot \|$. Indeed, for all $x \in \mathcal{B}(0, 1)$,

$$\|\mathcal{M}^{-1/2}x\|^2_{\mathcal{M}} = x^T\mathcal{M}^{-1/2}\mathcal{M}\mathcal{M}^{-1/2}x = x^Tx = 1. \tag{2}$$

Thus $\mathcal{M}^{-1/2} : (\mathbb{R}^n, \langle \cdot, \cdot \rangle) \to (\mathbb{R}^n, \langle \cdot, \cdot \rangle_{\mathcal{M}})$ is an isometry and the unit ball for $\mathcal{M}$ an ellipsoid. The principal axes of this ellipsoid are the eigenvectors of $\mathcal{M}$ with associated lengths $h_i = \lambda_i^{-1/2}$ (in the usual norm $\| \cdot \|$), with λ_i the eigenvalues of $\mathcal{M}$. These are the characteristic sizes and directions implied by the metric.

Unit elements Let us denote K_0 a regular* element and $\mathcal{SO}(n)$ the set of orthogonal matrices of determinant 1 (rotation matrices). A linear element K is unit in a metric $\mathcal{M}$ if:

- Edges are all length 1 in the norm $\| \cdot \|_{\mathcal{M}}$.
- There exists $R \in \mathcal{SO}(n)$ such that $K = \mathcal{M}^{-1/2}RK_0$

These two definitions are equivalent. Hence, we can either characterize unit elements by their edge lengths, or by saying they are produced by rotating an equilateral element and then mapping it through $\mathcal{M}^{-1/2}$, as illustrated in Fig. 1. An important feature of these definitions is that there is a unique metric in which an element is unit, but there are infinitely many elements unit in a given metric (a family indexed by a choice of rotation).

*equilateral in the usual sense

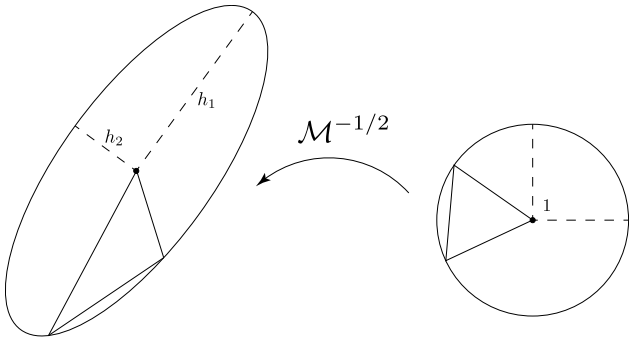


Fig. 1 $\mathcal{M}^{-1/2}$ maps the equilateral element (right) to an element unit in $\mathcal{M}$ (left). h_1 and h_2 are the characteristic sizes of $\mathcal{M}$, i.e. the eigenvalues of $\mathcal{M}^{-1/2}$.

The edge length definition is relaxed as follows: edge lengths need not be 1, but in the range $[1/\sqrt{2}, \sqrt{2}]$. This is usually referred to as quasi-unitness. Firstly, it is impossible to pave $\mathbb{R}^3$ with tetrahedra of exactly unit edges, but there exist pavings respecting the quasi-unit range [14]. Secondly, a consistent range is needed, in the sense that when a "long" edge is split, it must not produce two "short" edges, and reciprocally with merging. This is necessary for the termination of most h-adaptation algorithms. Thirdly, this range should be as small as possible. These two points argue for a range in the form of $[a, 2a]$ that includes 1. Lastly, symmetry dictates the range extremities should be multiplicatively equal distance from 1, hence of the form $[1/b, b]$. This leaves $[1/\sqrt{2}, \sqrt{2}]$ as the only possibility.

The two definitions merely offer different perspectives on unit elements. Edge length is geometrically simplest, and lends itself well to driving topological changes: short edges are collapsed and long edges split by inserting a new vertex. However, edge length alone does not inform very well on the shape of elements: it is possible to produce quasi-unit elements that are flat, rendering the mesh invalid. Usage of the Jacobian-based definition is slightly more subtle, as it is seldom written as such. It hides in the use of mesh quality functions, commonly:

$$Q_{\mathcal{M}}(K) = \frac{\det(\mathcal{J}_0^{-T}\mathcal{J}_K^T\mathcal{M}\mathcal{J}_K\mathcal{J}_0^{-1})^{1/n}}{\frac{1}{n}\mathrm{tra}(\mathcal{J}_0^{-T}\mathcal{J}_K^T\mathcal{M}\mathcal{J}_K\mathcal{J}_0^{-1})} \tag{3}$$

or its inverse. Geometrically, this can be interpreted as the ratio of volume (or area) to squared edge lengths, both under the metric $\mathcal{M}$. Algebraically, it is minimized if, and only if, there exists $\lambda > 0$ and R a rotation matrix such that:

$$\mathcal{J}_K = \sqrt{\lambda}\mathcal{M}^{-1/2}R\mathcal{J}_0 \tag{4}$$

This means that the minimizers of the quality function are the elements that are unit per the Jacobian-based definition up to scale.

Metric fields and unit meshes Ideal anisotropy and sizes vary within the domain. Thus we are not interested in single metrics, but in metric fields, i.e. (smooth enough) functions from the computational domain onto metrics. Geometric quantities such as length are extended by integration, e.g. for $\gamma : [0, 1] \to \mathbb{R}^n$ a curve parameterization,

$$\ell_{\mathcal{M}}(\gamma) = \int_0^1 \left(\gamma'(t)^T\mathcal{M}(\gamma(t))\gamma'(t)\right)^{1/2} dt \tag{5}$$

is the length of the curve $\gamma([0, 1])$ in the metric field $\mathcal{M}(\cdot)$.

A mesh is unit in a metric field $\mathcal{M}(\cdot)$ if its elements are all unit in $\mathcal{M}(\cdot)$. In turn, an element is unit in $\mathcal{M}(\cdot)$ if:

- Its integrated edge lengths in $\mathcal{M}(\cdot)$ are 1
- It is unit in the metric at a collocation point within the element (e.g. the center) or in some average sense

These definitions generalize unitness in a single metric, but they are no longer strictly equivalent. However, they are consistent when $h \to 0$. This affords some flexibility and lets the two coexist within a given piece of adaptation software. For instance, edge-based remeshing primitives [48–51] make use of the edge-length-based definition, whereas quality-based smoothing [52, 53] may collocate the metric field at element barycentres, and rely on the Jacobian-based definition as previously underlined.

Note that, even in the case of linear meshes, strict unitness cannot be achieved as, besides any practical considerations of difficulty, metric fields are arbitrary whereas meshes are limited to particular element types (linear, quadratic, etc) and offer, at best, a piecewise (usually polynomial) approximation of $\mathcal{M}^{-1/2}$. Despite this, producing meshes as close to unit as possible remains an attractive feature, which contributes to solver robustness, quality of approximation, and is paramount to the convergence of certain methods such as the MOESS [3–6] a posteriori metric-based error estimation method.

Metric-based adaptation Using the unit mesh definitions, errors of interest over a mesh can be expressed in terms of the metric field in which the mesh is unit. As such, the problem of minimizing a given error as a function of the mesh is no longer defined over a space of discrete objects (meshes) but rather a space of continuous, well-behaved functions (metric fields).

Metric fields to minimize L^p norms of interpolation error have been derived in [7–9], but more general errors such as on quantities of interest in aeronautics (drag, lift) can also be minimized [1, 10, 11], or even boundary approximation error [12, 13].

Certain methods rely on a posteriori estimation, providing generic error estimates tailored to the currently used scheme, such as MOESS [3–6]. Moess stages local mesh modifications followed by fast local resolves, thus associating metric changes to observed errors changes. These metric-error change pairs are then used to regress a generic error model. Finally, this generic error model is minimized to provide an optimal change in the implied metric field. A major advantage of this approach is the ability to provide metric-based control of any error via numerical evaluation rather than a priori analytical derivation.

B. High-order meshes

This Section introduces the notations used for high-order meshes in this paper. We denote by k the simplex topological dimension, by d a degree and by n the space dimension. K designates a k-simplex of degree d. The barycentric domain is defined by $\widehat{K} = \{\xi \in [0,1]^{k+1},\ \sum \xi_i = 1\}$. Similarly, let $\widehat{K}_k^d = \{\alpha \in [\![0,d]\!]^{k+1}, \sum \alpha_i = d\}$ be the set of multi-indices for homogeneous degree d $(k+1)$-variate polynomials. The Bernstein polynomial of index α is defined as

$$\forall \xi \in \widehat{K},\ B_\alpha(\xi) = \frac{d!}{\alpha_1!...\alpha_{k+1}!}\xi_1^{\alpha_1}...\xi_{k+1}^{\alpha_{k+1}} \tag{6}$$

The Bernstein polynomials $(B_\alpha)_{\alpha\in\widehat{K}_k^d}$ form a basis of $(k+1)$-variate homogeneous degree d polynomials. The Lagrange polynomial of index α, ϕ_α, is defined by

$$\forall \beta \in \widehat{K}_k^d,\ \phi_\alpha(\beta/d) = \delta_{\alpha\beta} \tag{7}$$

where $\delta_{\alpha\beta} = 1$ if $\alpha = \beta$ and 0 otherwise. This corresponds to regularly spaced Lagrange nodes. The Lagrange polynomials also span a basis of $(k+1)$-variate homogeneous degree d polynomials. When the multi-indices are written *in extenso* they are written e.g. 210 instead of as the tuple $(2,1,0)$. Thus we may write the Bernstein function B_{211} or the Lagrange function L_{200}.

A high-order element K is defined by a homogeneous polynomial F_K over the barycentric domain. The coefficients in the Bernstein basis of this polynomial are called Bézier control points and, in the Lagrange basis, Lagrange nodes. The former are denoted P_α and the latter L_α. Thus

$$F_K = \sum P_\alpha B_\alpha = \sum L_\alpha \phi_\alpha. \tag{8}$$

The Lagrange nodes all lie on the image by F_K of $\widehat{K}$ (the element proper). Only the control points of index de_i (such as $P_{de_1} = P_{d000}$) lie on the element; these are its vertices.

Reference element All functions previously defined in the context of k-simplices are $(k+1)$-variate, they depend on the barycentric coordinates. We can instead consider the physical coordinates within a reference simplex. The reference k-simplex is typically defined as

$$\widetilde{K} = \{x \in [0,1]^k,\ \sum_{i=1}^{k} x_i \leq 1\} \tag{9}$$

A bijection between the reference element $\widetilde{K}$ and the barycentric domain $\widehat{K}$ is usually chosen as $\phi : \xi \in \widehat{K} \mapsto (\xi_2, ..., \xi_{k+1}) \in \widetilde{K}$, corresponding to the reference element with vertices $P_1 = 0$, $P_2 = e_1$ through $P_{k+1} = e_k$ with e_i the canonical basis vectors of $\mathbb{R}^k$. The inverse of this function is given by $\phi^{-1}(x) = (1 - x_1 - ... - x_k, x_1, ..., x_k)$.

We refer to the k-variate polynomials as the above (Bernstein, Lagrange, element mapping) composed with this choice of ϕ^{-1}. It follows that, if $\widetilde{F}_K$ is the k-variate element mapping, then $\partial_i \widetilde{F}_K = (\partial_{i+1} - \partial_1) F_K$. The Jacobian matrix of F_K is then

$$\partial \widetilde{F}_K = \begin{pmatrix} (\partial_2 - \partial_1) F_K^T \\ ... \\ (\partial_{k+1} - \partial_1) F_K^T \end{pmatrix}. \tag{10}$$

We conflate $\widetilde{F}_K$ and F_K depending on context; in particular, the Jacobian matrix of F_K refers to that of $\widetilde{F}_K$, as for all other derivatives. In the most common case where space and topological dimension are the same $k = n$, this Jacobian matrix is square. We denote it $\mathcal{J}_K$.

High-order mesh validity An element is said to be valid if its element mapping defined over the reference element is invertible. This condition is necessary to define Finite Element bases over the mesh. This is equivalent to the pair of conditions: the Jacobian matrix of the mapping is everywhere invertible, and the boundary of the element does not self-intersect. By continuity of the Jacobian determinant, it must remain of constant sign, positive by convention.

We summarize here a standard technique to rigorously enforce validity of high-order meshes [44, 54]. The Jacobian matrix determinant denoted

$$J_K = \det(\mathcal{J}_K) \tag{11}$$

is itself a polynomial, of degree $d' = k(d-1)$, where we recall k is the simplex topological dimension and d its polynomial degree. Hence, the determinant can be written in the Bernstein basis: there exist so-called control coefficients N_α, such that

$$J_K = \sum_{\alpha \in \widehat{K}_k^{d'}} N_\alpha B_\alpha \tag{12}$$

The Bernstein polynomials verify a partition of unity property:

$$\forall \alpha \in \widehat{K}_k^{d'},\ 0 \le B_\alpha \le 1 \quad \text{and} \quad \sum_{\alpha \in \widehat{K}_k^{d'}} B_\alpha = 1 \tag{13}$$

Hence, the determinant J_K can be bounded using the control coefficients:

$$\min_{\alpha \in \widehat{K}_k^{d'}} N_\alpha \le J_K \le \max_{\alpha \in \widehat{K}_k^{d'}} N_\alpha \tag{14}$$

These bounds can be refined by subdivisions [44], and provide a sufficient condition for validity: if all control coefficients are above a threshold j_0, then $J_K > j_0$ everywhere. Practical derivations can be found in [55]. We present a method to enforce positivity in Section III.C.

C. Unit high-order meshes

Unit high order meshes are defined by generalizing the definitions for linear meshes. We generalize the definition by mapping, rather than length. Considering $\mathcal{J}_K$ the Jacobian matrix of the mapping that sends the reference element to K, it is equivalent to:

$$K \text{ unit in } \mathcal{M} \iff \exists R \in \mathcal{SO}(n),\ \mathcal{J}_K = \mathcal{M}^{-1/2} R \mathcal{J}_{K_0} \tag{15}$$

Of the two definitions — edge length and Jacobian-based — this one, based on the Jacobian matrix, generalizes itself the most easily to non-linear meshes. It also provides quantities useful for error estimation. Indeed, it is immediate that:

$$\Pi_K f = \Pi_{\widehat{K}} (f \circ F_K) \circ F_K^{-1} \tag{16}$$

where Π_K designates the interpolate on K, $\widehat{K}$ is the reference element on which the Galerkin basis is defined, and f is any function of interest to approximate. Hence the function of interest is never considered in physical space, but rather in reference space, hence any error analysis (usually based on Taylor expansions), will involve terms such as:

$$\partial (f \circ F_K) = \mathcal{J}_K \partial f \tag{17}$$

where ∂f designates the Jacobian matrix or gradient of f. Then, the Jacobian matrix $\mathcal{J}_K$ can be replaced by a metric-dependent quantity immediately:

$$\partial(f \circ F_K) = \mathcal{M}^{-1/2} R \mathcal{J}_{K_0} \partial f \tag{18}$$

and this will eventually give rise to an error estimate that does not depend on K anymore, but only the metric field in which it is unit.

In [38], we proposed a generalization of the unit mesh framework to high-order meshes based on continuous deformation of an ideal element by the metric field. Concretely, if $\mathcal{J}_K$ is the Jacobian matrix of the element K, K_0 is an ideal element of Jacobian $\mathcal{J}_0$, then K is unit in $\mathcal{M}(\cdot)$ if there exists R a rotation matrix, such that

$$\forall \xi \in \widehat{K},\ \mathcal{J}_K(\xi) = \mathcal{M}^{-1/2}(F_K(\xi)) R \mathcal{J}_0. \tag{19}$$

This is a formal relationship in which $\mathcal{M}$ and K are arbitary, e.g. K could be a rational or non-simplex element. The relationship cannot be verified exactly for specific types of K (e.g. triangle of degree 2) given arbitrary $\mathcal{M}$. However, we can restrict $\mathcal{M}$ to classes of metrics such that the relationship can hold exactly for the given class of elements.

We particularized to P^2 simplices in [38]. In that case, $\mathcal{J}_K$ is a linear function. Hence, the suitable class of metrics are those such that $\mathcal{M}^{-1/2} \circ F_K$ is also linear. We show that given some additional symmetry conditions involving $\mathcal{M}$ and its derivatives, Eq. (19) is analogous to the unit mesh definition on linear meshes as the two fundamental properties are verified. Firstly, letting $\mathcal{M}$ be known, Eq. (19) defines elements uniquely up to rotation in the reference frame. Secondly, letting K be known, Eq. (19) defines the metric field uniquely. We also showed, by deriving an interpolation error estimate on P^2 meshes, that this correspondence remains enough to derive purely metric-based error estimates on curved meshes, i.e. where the particular element considered (given by the choice of rotation in the reference frame) is of no importance, and the final result depends only on the metric field in which the element is unit.

Some general properties were also shown. By considering embedded metric fields $\mathcal{M}_h^{-1/2} = h\mathcal{M}^{-1/2}$, we showed curvature of unit elements (norm of second derivatives) is a $O(h^2)$, regardless of the base metric $\mathcal{M}$. This is a sufficient condition to preserve high-order interpolation error convergence rates [20] and thus avoids common fears of subpar convergence on curved meshes [27, 28]. This also means that validity becomes less of an issue as meshes are converged.

Two possible uses of this definition are used within Metris and detailed in Section IV.

IV. Metris

Metris is described in this Section. This is open-source software under license GNU GPL v3.0 accessible via its GitHub repository[†]. We begin with an overview, projected and implemented features in Subsection IV.A. Subsection IV.B presents adaptation within Metris in more detail. Lastly, Subsection IV.C presents edge-based unit curving formulation, and recalls the LP-based high-order Jacobian correction algorithm [35].

A. Overview

Purpose High-order meshing, and especially metric-based high-order adaptation, is still a relatively immature technology. As such, existing metric-based adaptation software very rarely implements high-order functionality, or as an add-on after years to decades of development of linear (re)meshing software. Supporting high-order elements requires reworking such software down to the lowest level functions (e.g. evaluation within an element) and data structures (e.g. connectivity arrays). Hence a very large proportion of the current code base needs attention when generalizing to high-order meshes. Worse, a number of algorithms will no longer be appropriate for high-order meshes at all. As such, it is often very difficult, and not necessarily very sound, to adapt existing mature linear mesh adaptation software to support high-order meshes.

Metris aims to provide 2D and 3D metric-based adaptation on simplex meshes of any degree, similar to refine [40], feflo.a [41], avro [42], EPIC [43], MMG [56], or PRAgMaTIc [57]. Some mesh generation software such as AFLR [58] and BAMG [59] also offers metric-based adaptation. We also aim to support complex and non-manifold geometry. This software is built from the ground up to support high-order meshes, but it can still provide usual linear adaptation capability. Table 1 summarizes characteristics of a sampling of existing metric-based mesh adaptation software. Though this cannot properly summarize the many ways in which these pieces of software differ, the take-away is that there is currently no freely available software that handles high-order meshes as well as geometry.

[†] `https://github.com/LucienRochery/Metris`

Table 1 Features and status of current metric-based mesh adaptation software.

Remesher	Freely available	CAD support	High-order features	Dimension
Refine	✓	✓	×	3D
feflo.a	×	✓	×	3D
avro	✓	✓	×	4D
MMG	✓	×	×	3D
EPIC	×	✓	✓	3D
BAMG	✓	✓	×	2D
AFLR	✓	×	×	3D
PRAgMaTIc	✓	×	×	3D
Metris	✓	✓	✓	2D

Current capabilities At the current stage, Metris is capable of 2D adaptation of linear and degree 2 meshes, with Computer-Aided Design (CAD) and non-manifold geometry support, and it applies mesh curvature either from the boundary, the metric field, or both, while guaranteeing validity.

All topological changes are carried out using a single so-called cavity operator such as in feflo.a [41], AFLR [58], refine [40] or avro [42]. This is a generic reconnection operator that can carry out insertions, collapses, and more complex operations depending on inputs. Operations can be rejected at any stage of the operator while keeping the mesh valid. Hence, the mesh remains valid before and after each operation, regardless of whether it succeeds. We implemented a general high-order version up to dimension two and surface, with dimension three being planned with relevant infrastructure already available (low level topological routines, data structures for tetrahedra, etc). Implementation of curving within the cavity operator is still in progress but the principle is similar to [60]: at the stage when (even in the linear case) the new cavity is checked valid, it is also curved and made valid, and rejected if this is not possible.

The flexibility of the operator is particularly desirable in this context as adaptation methods are bound to evolve — and perhaps require more sophisticated operations — as work on high-order adaptation continues. The high-level adaptation approach is similar to refine, avro, feflo.a or MMG with edge-based primitives. When meshes are degree 2 and more, we consider the curvilinear edge length approximated by a geometric length interpolation (which is also used in the linear case). The cavity operator also handles non-manifold meshes. Non manifold meshing will allow, in particular, the insertion of explicit wake sheets in meshes, as required by full potential methods [61].

Geometry is handled using the Electronic Geometry Aircraft Design System (EGADS) library [62] and the facilities in the libMeshb format [63], namely the keywords VerticesOnGeometricEdges and VerticesOnGeometricFaces. These allow us to store the t or (u, v) coordinates of a point — on a geometric edge or face, respectively — along with the mesh. This drastically reduces required inverse evaluations, which are common points of failure. Moreover, Metris can handle configurations such as edges or faces looping back on themselves, wherein a single point might have two distinct t coordinates in the same geometric edge (resp. (u, v) in the same face). This occurs very commonly, e.g. on cylinders or cones, or even simple 2D airfoil models.

The metric field is stored as a log-Euclidean [64] discrete field on a so-called back mesh. Analytical metrics can also be used, which can be of interest in a research setting, and were designed to be very simple to increase the roster of. The log-Euclidean interpolated metric at a point can be differentiated within Metris, which is of use in optimizing anisotropic quality functions or other geometric quantities. The back mesh is the input mesh by default, of which an immutable copy is made, but a distinct back mesh can also be passed in.

Meshes can be elevated to a higher degree. New points are placed on the geometry and the mesh is corrected using the LP-based approach described in Section IV.C. This is a global version of the method introduced in [35]. Metric-based curving can be carried out using either the LP-based approach, or classic quality optimization a described in IV.C, similar to [65]. Validity is controlled at all stages using control coefficients, hence the meshes can be rigorously guaranteed valid [44]. The user can set a minimum scaled Jacobian determinant threshold to control overall curvature. Applying curvature within the cavity operator follows the model from [60] and is undergoing implementation and testing. The cavity operator does currently handle curved high-order meshes, but it can only verify the operations are valid, not correct them.

Software details Metris is written in C++ and can be run either from the command line or through a C++ Application Programming Interface (API). The latter is already used within the solver SANS [45]. From the command line, meshes and metrics are provided through `.mesh(b)` files in the libmeshb format [63]. The EXODUS II [66] file format is projected to be supported.

Metris is designed to be simple to develop in, with most data stored as plain arrays inside very few master classes or structures, and most functionality implemented as loose functions taking in low level data. These are wrapped in higher level functions when this adds convenience and avoids repetition and potential for error. Our aim is for Metris to remain flexible enough to be used to rapidly prototype code involving high-order meshes (hence loose functions), but still structured and convenient enough in most common cases (hence calls from higher-level functions within classes).

A number of low-level functions are templated by degree and dimension. This is a C++ feature that leads to the compiler generating code for different values of these parameters. This allows certain arrays or execution flows to be determined at compile time, rather than at run time. A balance must be struck, as executable size and compilation time rise with increased use of this facility. To illustrate the positive effects of this approach, Table 2 provides a benchmark of a very low-level function, evaluation at a barycentric coordinate. RT refers to run-time: degree and dimension are dynamic variables. CTL corresponds to "compile-time looping" using the Boost metaprogramming library Hana [67]. De Castlejau is a recursive evaluation algorithm for Bézier elements which was either implemented via templates or via runtime recursion. In the former case, the whole execution tree is determined at compile time. The highest performances for Bézier are reached using de Casteljau with compile-time recursion: this is equivalent to writing by hand the entire recursive formula for a given degree and dimension, but is done automatically by the compiler. The difference in performance is between 50% and 3x depending on degree, compared to a naive implementation of Bézier evaluation with runtime variables. For Lagrange, the highest performance is obtained by a hand-written routine for the target degree, the difference is a factor 10x to 5x to a naive runtime implementation. The advantage of this approach wanes against the "CTL" approach at degree 3 as there is higher arithmetic complexity to compensate for the Hana library overhead. Still, we decided to use code generation (manually implemented, without template metaprogramming) to generate the Lagrange evaluation functions. The same principles are applied to other functions, when similar speedups were observed.

Table 2 Evaluation speeds on tetrahedra in millions per second using run-time degree and dimension ("RT"), compile-time looping using the Boost Hana library ("CTL") and using specialized methods ("Hand" specialized to degrees and dimension and de Casteljau recursive either runtime or compile-time).

Algorithm\Degree	P2	P3
RT Lagrange	13.4 M/s	7.1 M/s
CTL Lagrange	76.7 M/s	34.1 M/s
Hand Lagrange	145.5 M/s	34.3 M/s
RT Bézier	108.4 M/s	17.5 M/s
CTL Bézier	75.7 M/s	42.3 M/s
Casteljau (template)	166.1 M/s	72.5 M/s
Casteljau (runtime)	17.6 M/s	4.6 M/s

API A `MetrisAPI` object serves as a "file in memory". A `MetrisAPI` object can be created passing in entity counts, filled up either with batch or individual entity functions, and then passed to a `MetrisRunner`, the main object driving adaptation. Reciprocically, a `MetrisRunner` object can be passed to a `MetrisAPI` object to recover the final adapted or modified mesh. This allows back-and-forth between a solver and Metris while incurring no IO costs, and needing only knowledge of this simplified interface, rather than the internal Metris objects.

B. Mesh adaptation

We now describe how adaptation is carried out in more detail. The global 2D algorithm is very classic and summarized in Algo. 1.

Algorithm 1 Global 2D adaptation loop

```
adaptBoundary()
while Iterations less than adaptIter and operations done do
    collapseShortEdges()
    swapEdges()
    insertLongEdges()
    if all three stagnated and iterations remain then
        optimizeMesh()
        if optimizeMesh() stagnated then
            exit
        end if
        continue loop
    end if
end while
optimizeMesh()
```


Boundary remeshing The first step is to adapt the boundary agressively, which is relatively simple to do in 2D. Our approach seeks to create edges with length as close as possible to 1 along the boundary, or as close to equal length. This is subject to a tolerance ϵ_0 the user can set. The process is iterative.

If the initial mesh is too coarse, it cannot provide an accurate estimation of CAD curve length, hence nor of a reasonable mesh edge target length. For this reason, we start by computing CAD curve lengths in the metric field up to a tolerance related to ϵ_0 that guarantees sufficient precision. From these CAD curve lengths, we initialize the edge target lengths to the closest possible value to 1.

If the domain boundary is curved, the sum of mesh edge lengths will be lower than the total curve length. Hence, we begin an iterative process, where the target length is adjusted after each global CAD edge remeshing. This remains very fast as, asymptotically, only $1/h$ edges are in the mesh, against $1/h^2$ elements and vertices. A signed "error budget" is also accumulated which adjusts the target length interval of the edges as they are created. This is to prevent the situation where all edges are systematically e.g. too long (yet within the tolerance) leading to the last edge needing to be much too short (outside the tolerance).

Due to curvature, particularly if the domain is concave, certain insertions may fail. The set of edges to delete is forced, and there may be no choice of interior elements such that the final cavity will be valid. In that case, "Steiner points"[‡] can be inserted in the interior to facilitate insertion. These are placed along the normal to the boundary for greatest cavity visibility.

Though not strictly a part of this algorithm, back mesh localization on the boundary is most solicited here. To be robust, it needs to compare not only projected and sought point locations, but also the surface tangents or normals at those points. Indeed, it can happen that the background mesh is coarse, and points to be localized lie far from its elements, sometimes practically equidistant to two elements. As these points to be localized are evaluated on the CAD geometry, their normals are known. A maximum normal deviation per background boundary element is precomputed at initialization, which depends on geometry curvature locally. During localization, the ideal and element normals at projection are compared and must not exceed the element-bound maximum deviation.

This boundary remeshing approach has a tendency to decimate the mesh in the interior, but it provides high quality boundary meshes as a starting point. It also has "no-operation" checks beforehand to avoid remeshing the boundary when it is already satisfactory. However, these are perhaps too conservative and it is rare for a CAD curve not to be remeshed. Smoothing is not yet used here, but could serve to greatly reduce the need for boundary remeshing starting from the second adaptation iteration.

Adaptation loop The main loop is very classic: we collapse, swap and insert as long as operations remain to be done. The numbers of outer adaptation iterations is set either by parameters passed to the API, or by their command-line interface (CLI) option counterparts. The effective number of iterations can be lower if there remain no operations to

[‡]auxiliary points unrelated to adaptation needs

carry out. Edge length is computed by geometric length interpolation. The edge length is approximated by:

$$\forall t \in [0,1],\ \ell_{\mathcal{M}}(e)(t) \simeq \int_0^1 \ell_1^{1-t}\ell_2^t \tag{20}$$

where the lengths at the ends ℓ_1 and ℓ_2 are computed as

$$\ell_1 = e'(0)^T \mathcal{M}(e(0))e'(0) \quad \text{and} \quad \ell_2 = e'(1)^T \mathcal{M}(e(1))e'(1) \tag{21}$$

with e' the tangents to the edge e. The integral can be computed by hand. In our tests, using very fine quadrature as reference, this scheme appears slightly more accurate than linear length interpolation, but this might be specific to certain cases.

Cavity growing is carried out before calling insertion or collapse operators. It is initialized as the seed elements, either where the new point lies, or the ball of the point to collapse. It is then grown using an anisotropic Delaunay criterion. This cavity is then corrected for volume positivity. In the insertion case it is also extended to avoid new short edges as well as low height elements. Swapping is done based on edge length in the adaptation function, and on quality in the optimization function. In the case of length, a quality is attributed to the configurations based on deviation to unity.

Quality smoothing At stagnation, the mesh optimization function is called. It operates very similarly, alternating smoothing and swaps. Number of optimization iterations is also controlled by progression criteria and the upper bound set by a CLI option or parameter to the API. Smoothing is iterative per point using Newton's method on the metric conformity error norm

$$E_{\mathcal{M}}(K) = \int_{\xi \in \widehat{K}} |Q_{\mathcal{M}}(K)(\xi) - 1|^p \tag{22}$$

The quality function $Q_{\mathcal{M}}$ defined in Eq. (3) is commonly used in linear mesh optimization and has also been used to curve meshes [37, 54, 65, 68]. The integral is approximated by quadrature. The p-norm can be chosen higher to penalize more strongly the maximum deviations to 1, we use 2 by default.

A weakness of this approach is that there is no guarantee the rotation matrix is uniform. Indeed, assume that K perfectly minimizes $E_{\mathcal{M}}(K)^p = 0$. Then, all that we know is that:

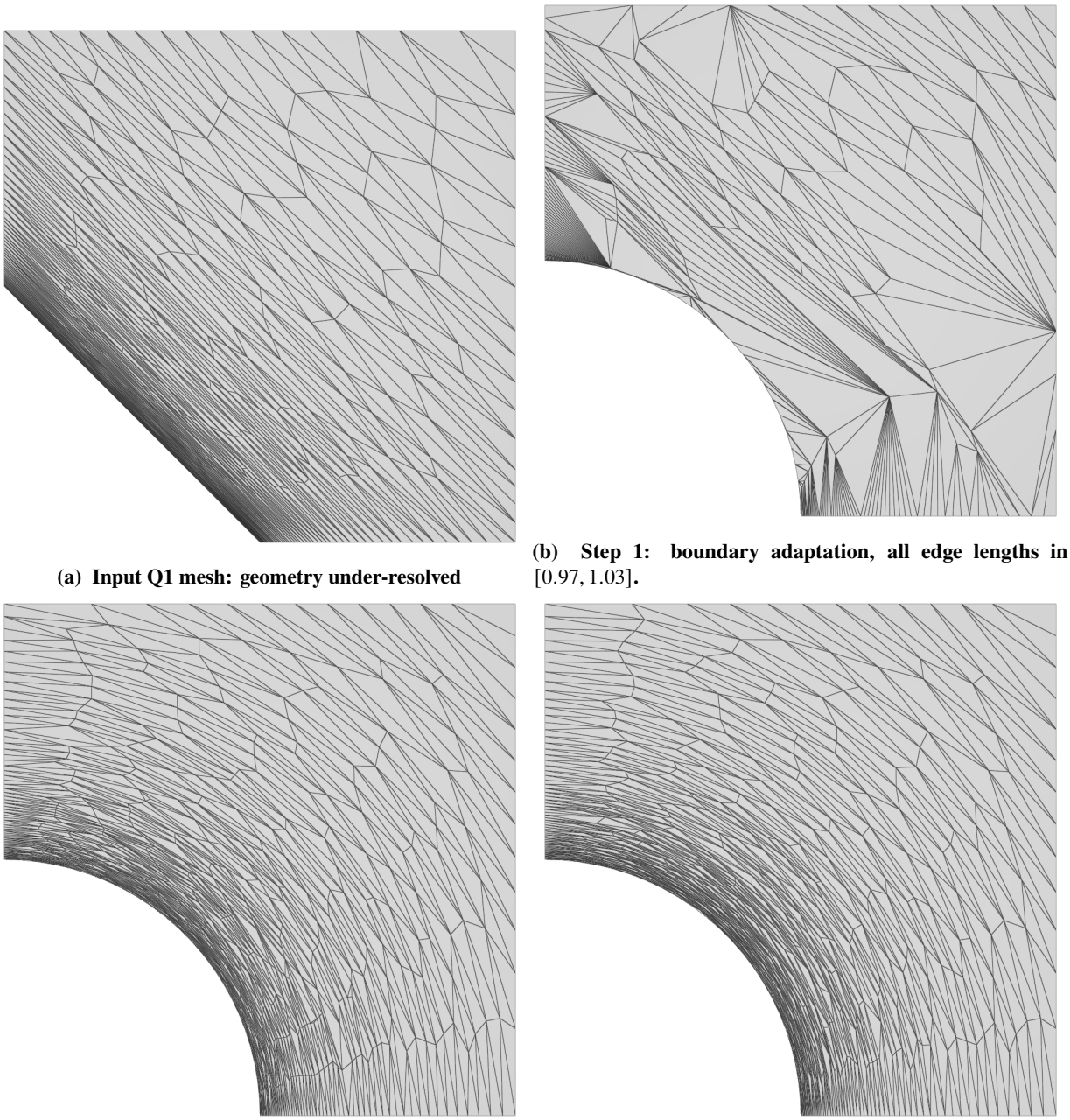
$$\forall \xi \in \widehat{K}, \exists \lambda(\xi), \exists R(\xi),\ \mathcal{J}_K(\xi) = \lambda(\xi)\mathcal{M}^{-1/2}(F_K(\xi))R(\xi)\mathcal{J}_0 \tag{23}$$

This is weaker than the unit definition which stipulates a constant-per-element R and $\lambda = 1$. Thus, this function may not be suitable to bring great changes to the mesh, but only to add finishing touches. It may already not be desirable in the Q1 case to rely too much on quality smoothing as it tends to be expensive. A metric conformity error more suitable to high-order meshes is something to look into in the future.

Illustration We provide an illustration of the adaptation steps in Figure 2. The top left is the initial mesh, it is comprised of a boundary layer sitting on a very under-resolved boundary (the geometry is curved). The top right is the mesh after the step `adaptBoundary()` in Algorithm 1. By re-generating the boundary with growing cavities, this method is capable of producing a unit boundary mesh. Moreover, all edges are with lengths in the range $[0.97, 1.03]$ which sets a good starting point for the next step. This case would have been particularly difficult to mesh the boundary of if not for this re-generation. Some Steiner points are visible along normals to the curved boundary, which are necessary for some insertions to be possible. The mesh is kept valid at each step of this boundary remeshing process, and naturally at the end of it. The result of the adaptation loop is show in the bottom left. Although some smoothing is used within the main adaptation loop, its purpose is mainly to unlock new adaptation operations, thus optimization iterations are kept to a minimum to reduce cost (in this case, up to 5). Lastly, the bottom right figure is the mesh after final smoothing and swapping, corresponding to the `optimizeMesh()` step. In this stage, considerably more iterations are carried out, up to 20 in this case.

C. Fast metric-based curving using Bézier offsets

If we assume the underlying linear mesh is unit, we can derive optimal positions of the high order nodes analytically using the unit mesh definition Eq. (19), provided the metric field is restricted to those where the relationship can hold



(a) Input Q1 mesh: geometry under-resolved

(b) Step 1: boundary adaptation, all edge lengths in $[0.97, 1.03]$**.**

(c) Step 2: interior Q1 adaptation.

(d) Step 3: smoothing and swapping (optimization).

Fig. 2 Illustration of major steps in 2D adaptation.

exactly. More specifically, $\mathcal{M}^{-1/2} \circ F_K$ should be linear, if the mesh is quadratic. This is analogous to the fact that — in the linear case with a single metric —, when an edge is set, the last point can be chosen to always produce a unit triangle. We begin by defining Bézier offsets, and then derive edge-intrinsic curvature (optimal offsets) relying on the work within [38].

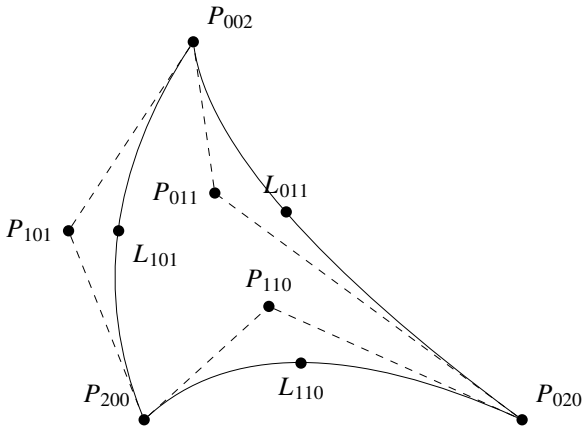


Fig. 3 Q2 Bézier triangle: control points $(P_\alpha)_\alpha$ and Lagrange nodes $(L_\alpha)_\alpha$. Control net in dashed lines.

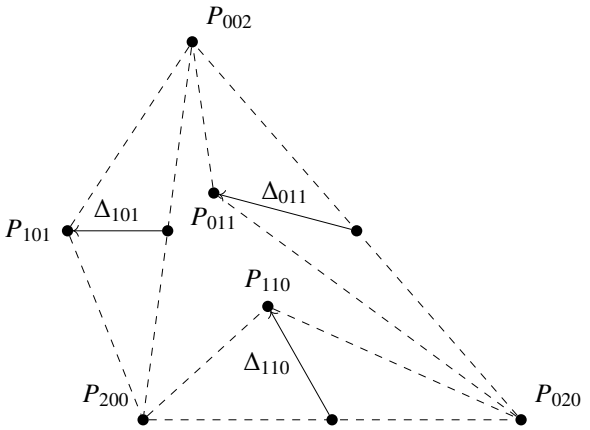


Fig. 4 Q2 Bézier triangle: offsets $(\Delta_\alpha)_\alpha$. Control net and subjacent linear element in dashed lines.

Bézier offsets Let us introduce Bézier offsets $\Delta_\alpha = P_\alpha - \frac{\alpha_i}{d} P_{de_i}$. For instance, $\Delta_{110} = P_{110} - \frac{1}{2}(P_{200} + P_{020})$ or $\Delta_{121} = P_{121} - \frac{1}{4}(P_{200} + 2P_{020} + P_{002})$. Then [32]:

$$F_K(\xi) = \sum_{i=1}^{k+1} \xi_i P_{de_i} + \sum_{\alpha \in \tilde{K}_k^d} \Delta_\alpha B_\alpha(\xi). \tag{24}$$

The P_{de_i} are the vertices of K, and the offsets at vertices (for some $\alpha = de_j$) vanish. Thus the sum written, for convenience, over all possible indices, only concerns non-vertex control points (or rather their offsets). Let us illustrate in the quadratic case. The vertex indices are 200, 020, 002 and the edge indices are 011, 101 and 110. The offsets at the vertices Δ_{200}, Δ_{020} and Δ_{002} vanish. Thus any quadratic triangle mapping can be written as

$$F_K(\xi) = \xi_1 P_{200} + \xi_2 P_{020} + \xi_3 P_{002} + 2\xi_1\xi_2\Delta_{110} + 2\xi_2\xi_3\Delta_{011} + 2\xi_3\xi_1\Delta_{101}. \tag{25}$$

Notice that the factors against vertices are linear regardless of element degree. Thus element non-linearity is entirely governed by these offsets between control points and the regularly-spaced nodes (such as $\frac{1}{2}(P_{200} + P_{020})$).

Figure 3 illustrates a Q2 triangle with control points and Lagrange nodes. The dashed lines are the Bézier hull or control polygon. Figure 4 illustrates the same triangle in terms of its Bézier offsets. The control polygon is shown together with the underlying linear triangle.

Optimal edge offsets Optimal offsets per element are derived in [38]. We introduce here an edge-intrinsic (rather than element-bound) offset formula. We illustrate in 2D as the generalization to 3D is immediate.

Let P_1, P_2 be two points, the edge $\ell = P_2 - P_1$ assumed unit in $\mathcal{M}$. Let P_3 be a point such that $K = P_1, P_2, P_3$ is a positively oriented triangle, and P_1, P_2, P_3 is unit in $\mathcal{M}$. Its Jacobian matrix is $(\ell \;\; P_3 - P_1)$. According to [38], the Bézier offset of the unit quadratic element is, for this edge,

$$\Delta_{110}^k = -\frac{1}{2} \sum_{st} N_{s1} N_{t1} T_{skt}^{\mathcal{M}}$$

with $N = R\mathcal{J}_0$ and $T^{\mathcal{M}}$ the third order tensor defined by:

$$T_{ijk}^{\mathcal{M}}(\xi_0) = \sum_l \partial_l \mathcal{M}_{jk}^{-1/2}(F_K(\xi_0)) \mathcal{M}_{il}^{-1/2}(F_K(\xi_0)) \tag{26}$$

with ξ_0 some metric collocation barycentric coordinate. In this case, we take ξ_0 to be the middle of the edge of interest.

Now, notice $N_{s1} = \mathcal{M}^{1/2}\ell_s$. Thus the offset

$$\Delta_\ell^k = \Delta_{110}^k = -\frac{1}{2} \sum_{st} (\mathcal{M}^{1/2}\ell)_s (\mathcal{M}^{1/2}\ell)_t T_{skt}^{\mathcal{M}}$$

does not depend on the particular element via R. It is intrinsic to the edge.

Computing these unit edge Bézier offsets is clearly very simple and cheap. The complexity and cost are mainly present in differentiating the metric field, which is also a requirement for derivative-based optimization of anisotropic quantities. Metric field derivatives in the context of log-Euclidean interpolation were obtained in [32] and [65]. It is also possible — as is done within Metris — to combine differentiation of basis functions at the log-interpolation step followed by automatic differentiation of the matrix exponential. Note that these are physical (not barycentric space) derivatives, hence the final step is a left multiplication by the element Jacobian matrix inverse.

The mesh curved using these offsets is not guaranteed to be valid. In the rest of this Subsection, we describe techniques for correcting high-order meshes based on Linear Programs (LP). Two methods are presented, one deals only with validity, the other applies the optimal Bézier offsets and boundary curvature while keeping validity.

High-order mesh correction We begin by describing the LP-based Jacobian correction method introduced in [35]. This is a general method to recast the problem of maximizing the minimum Jacobian determinant over a mesh as a Linear Program. It relies on variable separation.

Recall from Section III.B the Jacobian determinant J_K of a degree d dimension k simplex is a degree $d' = k(d-1)$ polynomial and may thus be written in the Bernstein basis with coefficients N_α as

$$J_K = \sum_{\alpha \in \widehat{K}_k^{d'}} N_\alpha B_\alpha \geq \min_\alpha N_\alpha. \tag{27}$$

Positivity of all control coefficients is a sufficient condition for the validity of K. Denoting P_i the mesh vertices and N_j the control coefficients of all mesh elements, solving:

$$\max_{P_i \in \mathbb{R}^2} \min_j N_j(P_i). \tag{28}$$

until the minimum is positive, guarantees validity of K.

Control coefficients are linear combinations of determinants of matrices whose columns are vertices P_i. Hence they are quadratic (2D) or cubic (3D) functions of the vertices. Thus the minimum of the control coefficients is piecewise cubic, which is not differentiable, hence unusable by classic optimization schema such as Newton's method. However, with variable separation, this may be recast to a Linear Program. Determinants are not general quadratic or cubic functions, they are bi- or tri-linear. Hence can be viewed as linear functions if: i) *only one* control point is allowed to move or ii) all control points are allowed to move along *one dimension only* at a time.

We'll refer to this, respectively, as separation by point and by coordinate. Although not within Metris, the separation by point has already been implemented and shown to provide very fast mesh correction [35], even in difficult 3D cases. The control point separation requires a sophisticated smoothing strategy, whereas the strategy for coordinate separation is trivially to loop over the coordinate rank 1 through 3 (in 3D). Moreover, all inter-dependencies between control points are accounted for by the optimization problem itself. The coordinate-separation validity LP is implemented in Metris, where we solve a tractable sequence of sparse linear programs, decoupled into coordinate-wise optimizations.

From now on, take X to be a global coordinate vector as the optimization variables and N_j all the control coefficients in the mesh. The control coefficients in their affine form in X become $N_j(X) = a_j + b_j^T X$, where $b_j^T = (\frac{\partial N_j}{\partial X_1}, \frac{\partial N_j}{\partial X_2}, ..., \frac{\partial N_j}{\partial X_m})$. Hence, the constraints are linear and by introducing an auxiliary variable t, we can solve

$$\max_{X \in \mathbb{R}^m} \min_j N_j(X), \tag{29}$$

in its equivalent LP formulation:

$$\begin{aligned} \max \quad & t \\ \text{s.t.} \quad & t \leq N_j(X), \ \forall j. \end{aligned} \tag{30}$$

We view the constraints as block matrices, with b_j^T as the rows of B, namely

$$-a \leq \begin{pmatrix} B & -1 \end{pmatrix} \begin{pmatrix} X \\ t \end{pmatrix}. \tag{31}$$

This LP directly maximizes the minimum element Jacobian determinant across the mesh and is solved with either the interior point method or the simplex method using open source libraries ALGLIB [69] and Coin-OR CLP [70]. The

currently supported optimization variables are either the full mesh or only high-order nodes. Although derivatives of control coefficients are only implemented for Q2 triangles and tetrahedra, the LP formulation is prepared to handle arbitrary polynomial degrees and remains applicable to other element types.

Bézier offsets LP Generalizing the pure validity LP, we propose a variant that simultaneously attempts to improve mesh quality by enforcing displacements of control points. Namely, the Bézier offset LP routine moves control points towards their target positions given by the Bézier offsets described in the beginning of this Subsection, while maintaining high-order validity through Jacobian positivity constraints — precisely as in the pure validity LP.

We minimize the norm of the distance between the current and target point positions. Norms L^1 and L^∞ can be implemented as LPs. Note that only an LP can solve for the L^∞ norm natively. First, for the L^∞ norm, we solve

$$\min_{X\in\mathbb{R}^m} \max_i |P_i - P_i^*|, \tag{32}$$

where the index i is over all mesh vertices, of which we take the fixed coordinate that the global coordinate vector X is considering. Similar to the pure validity case, we introduce a virtual variable t that we minimize such that $t \geq |P_i - P_i^*|$. The absolute value still remains nonlinear, and we introduce variables to represent the positive and negative parts of the expression. The LP writes

$$\begin{aligned} \min \quad & t \\ \text{s.t.} \quad & D_i - X_i \geq P_i - P_i^* - |P_i - P_i^*|, \\ & D_i + X_i \geq P_i^* - P_i - |P_i - P_i^*|, \\ & b_j^T X \geq j_0 V_j - a_j, \\ & t \geq D_i - |P_i - P_i^*|. \end{aligned} \tag{33}$$

The constraints may be equivalently expressed in the block matrix form

$$\begin{pmatrix} P - P^* - |P - P^*| \\ P^* - P - |P - P^*| \\ j_0 V - a \\ 0 \end{pmatrix} \leq \begin{pmatrix} -I & I & 0 \\ I & I & 0 \\ B & 0 & 0 \\ 0 & -I & \mathbf{1} \end{pmatrix} \begin{pmatrix} X \\ D \\ t \end{pmatrix}, \tag{34}$$

where I is the identity matrix of size m, $\mathbf{1}$ is a vector of 1s, and B has row dimension equal to the total number of control coefficients. In the L^1 norm mode, we solve

$$\min_{X\in\mathbb{R}^m} \sum_i |P_i - P_i^*|, \tag{35}$$

with the explicit LP as

$$\begin{aligned} \min \quad & \textstyle\sum_i D_i \\ \text{s.t.} \quad & \begin{pmatrix} P - P^* - |P - P^*| \\ P^* - P - |P - P^*| \\ j_0 V - a \end{pmatrix} \leq \begin{pmatrix} -I & I \\ I & I \\ B & 0 \end{pmatrix} \begin{pmatrix} X \\ D \end{pmatrix}. \end{aligned} \tag{36}$$

Local Variants The methods developed both admit local variants that may offer advantages in performance and adaptability to localized features of the mesh. We aim to validate curved elements and conform them to a prescribed offset position throughout the adaptation process, rather than solely as a final post-processing step. Even in the global coordinate separation, there's still the matter of selecting points to optimize as it may not be necessary to move the entire mesh if only few elements are invalid. This is particularly the case as meshes are refined: both boundary and metric-induced curvature become lesser. Future work will include generalizing the scheme to arbitrary mesh subsets, and current development includes using the Bezier Offset LP to correct the final cavity within the cavity operator.

Examples We illustrate a posteriori curving in Figure 5. The linear mesh is adapted (left) to an analytical metric

$$\mathcal{M}(r,\theta) = P(\theta)DP(\theta)^T \quad \text{with} \quad P(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \quad \text{and} \quad D = \begin{pmatrix} 1/h_1^2 & 0 \\ 0 & 1/h_2^2 \end{pmatrix} \tag{37}$$

with $h_1 = 0.1$ and $h_2 = 0.5$. This mesh is curved using quality optimization alone. The minimum scaled Jacobian determinant is positive at around 4×10^{-4}. A higher threshold `X` can be set with the CLI option `-jtol X`.

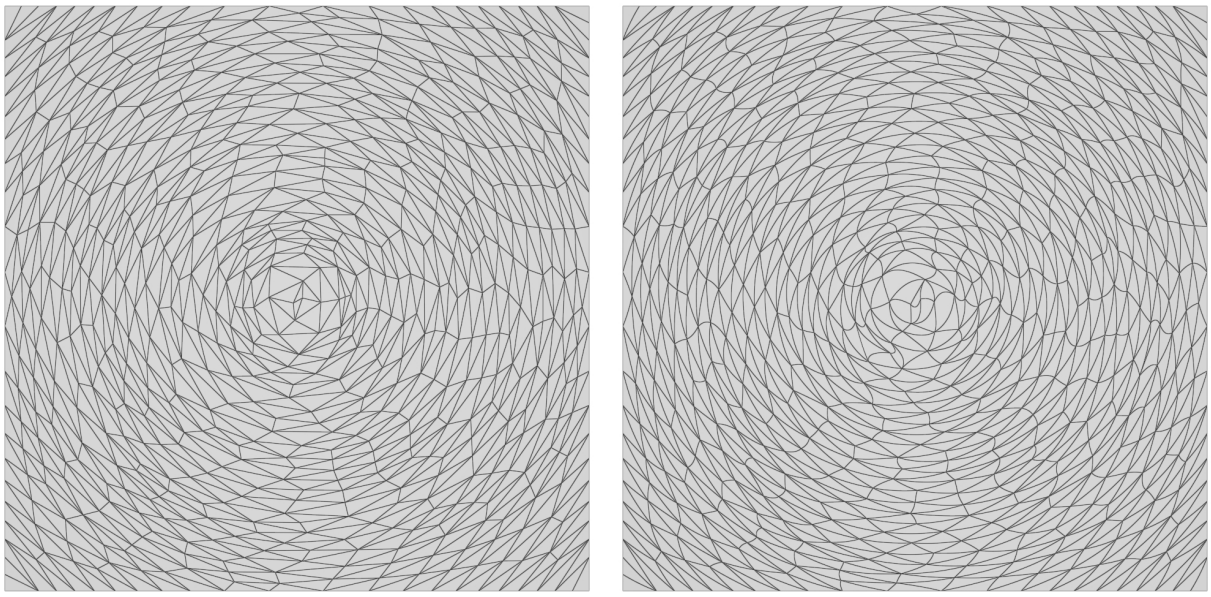
The initial Q1 mesh had a conformity error L^2 norm of 0.027 (normalized by number of elements). A very simple uniform quadrature is used at the nodes of the elements. Hence, for a degree 1 mesh, it is computed using the conformity error at the vertices and, for a degree 2 mesh, at the vertices and control points. Once elevated in degree but before any curving, the conformity error is recomputed (now as a Q2 mesh) and found to be 0.127. After curving, the conformity error norm drops to 0.015. Hence, we manage to produce curved meshes as unit as their Q1 counterparts.

Regarding edge length, curving tends to minimize edge length in this case. The initial Q1 mesh has an average edge length of 1.036 and the curved mesh of 0.978. This is not contradictory with the previous point, as conformity error is invariant to scale, it only controls shape. This could indicate the relevance of carrying out adaptation on Q2 meshes from the onset, instead of adding curvature only at the end, to better take into account required number of degrees of freedom (which the adaptation loop — but not optimization — does by insertions and collapses).

We now illustrate the LP approach. For this we now consider the analytical metric

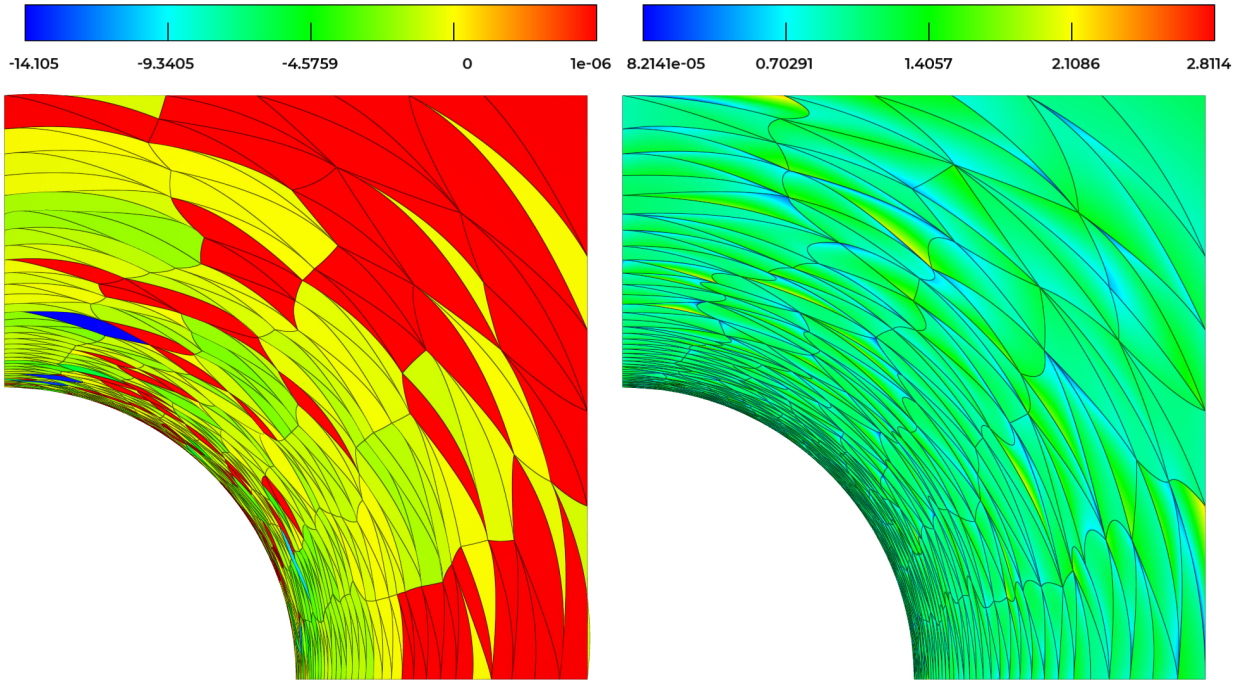
$$\mathcal{M}'(r,\theta) = P(\theta)D'P(\theta)^T \quad \text{with} \quad P(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \quad \text{and} \quad D'(r) = \begin{pmatrix} 1/h_1^2 & 0 \\ 0 & 1/h_2(r)^2 \end{pmatrix} \tag{38}$$

with $h_1 = 0.5$, $h_2(r + r_0) = r h_{hi} + (1 - r)h_{lo}$ with $h_{lo} = 10^{-3}$ and $h_{hi} = 0.1$. This metric field is radially invariant with anisotropy that increases as we approach $r = r_0$, with $r_0 = 0.5$ in this case. We consider the same "reentrant circle" geometry as in Figure 2. Nodes on the boundary are evaluated to the geometry and, in the interior, their optimal Bézier offsets computed. The resulting mesh is illustrated in Figure 6a. The minimum control coefficient per element is show as a solution field: only the elements in red are valid (above the user-set 10^{-6} threshold). Hence, a majority of the mesh is invalid. The mesh is then reverted to straight and the desired node positions passed to the offsets-based LP, which curves the mesh while keeping it valid. The resulting mesh is illustrated in Figure 6b.



(a) Initial Q1 mesh **(b) Q2 mesh curved to metric**

Fig. 5 Illustration of degree elevation to Q2 and a posteriori curving using quality optimization and swaps.



(a) Q2 mesh with nodes projected on boundary and moved by optimal offset positions in the interior. Only the red elements are valid.

(b) Q2 mesh after offset-based LP curving. All elements have scaled Jacobian determinant above the threshold 10^{-6}: the mesh is valid.

Fig. 6 Illustration of degree elevation to Q2 and a posteriori curving by setting boundary nodes on geometry and using optimal Bézier offsets in the interior, followed by offsets-based LP curving with strict validity constraints.

V. Numerical results

The solver SANS (Solution Adaptive Numerical Simulator) [45] is used throughout this Section. It can solve a wide range of PDEs using Continuous Galerkin (CG) and Discontinuous Galerkin (DG) discretizations, including synthetic problems such as L^2 projection of an analytical function. An important feature of SANS is its ability to carry out error estimation using the MOESS method [3–6]. In essence, MOESS proceeds in two phases. It first fits a metric-based error model to observed (error,anisotropy) pairs. Elements are split in configurations of differing anisotropy (hence associated local metric). Local solves are then carried out on these anisotropic configurations, and the error to the original numerical solution computed (or more sophisticated schemes such as DWR), providing the (error,anisotropy) pairs. These are finally fit to a generic metric-based error model. Secondly, the error model — which is a function of the mesh-intrinsic metric — is minimized, providing an optimal metric for the next adaptation iteration.

A. L^2 error minimization of L^2 projection of analytical solutions

We begin by considering a controlled setting where the solver is an L^2 projection of an analytical solution and the error of interest is the L^2 norm of error. The solver is SANS [45] and we compare results with freely available avro [42] and refine [40]. We consider the following analytical functions:

- The corner singularity (CS) function

$$f_{CS}(r,\theta) = (r + r_0)^{\alpha} \sin(\alpha(\theta + \theta_0)) \tag{39}$$

with the parameter values $\alpha = 2/3$, $\theta_0 = \pi/2$ and $r_0 = 0$. This is an isotropic function. It is studied in [71] where the optimal mesh sizing is shown to follow:

$$h_{opt}(r) = r^{1-\frac{\alpha+1}{p+2}} \tag{40}$$

where p is the polynomial degree of the scheme.

- The double boundary-layer (2BL):

$$f_{2BL}(x, y) = \lambda f_a(x) f_b(y) + c \tag{41}$$

with:

$$f_\alpha(z) = \frac{1 - \exp(-\alpha(\delta_e - x\lambda_e)/\nu)}{1 - \exp(-\alpha/\nu)} \tag{42}$$

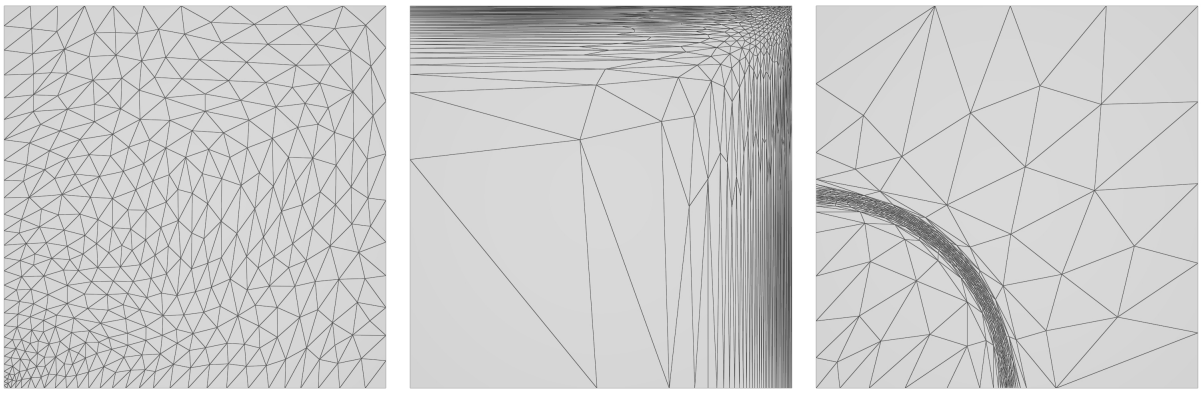
with the following parameter values: $a = 3/5$, $b = 4/5$, $\nu = 1/50$, $\delta_e = 1$, $\lambda_e = -1$. This is an anisotropic function with almost uniform anisotropy.

- The polar Gauss (PG) function

$$f_{PG}(r, \theta) = \lambda \exp\left(-a(r - r_0)^2\right) + ar^{2/3}/4 \tag{43}$$

with the parameter values: $a = 10^4$, $r_0 = 0.5$. This function is anisotropic with anisotropy directions being radially invariant. Thus, it is anisotropic and exhibits curvature, rendering it more difficult to adapt to.

These functions are illustrated in 7 via meshes adapted to minimize their L^2 error.



(a) Function CS **(b) Function 2BL** **(c) Function PG**

Fig. 7 Analytical function illustrations: intermediate meshes (≃ 2000 points) adapted by Metris and P1 solver.

Methodology We consider the L^2 projection solver using a Continuous Galerkin (CG) discretization of orders 1, 2 and 3. MOESS is set to minimize the L^2 error. We compare results between uniform Cartesian grids, Metris, avro [42], and refine [40]. The interfaces for the latter two had already been used within SANS [42, 72] and the Metris API was used for interfacing with SANS. Adaptation is run with target complexities 250 through 16000 doubling at each step. Each step is itself a loop of 50 solver-adaptation iterations. The uniform grids naturally don't involve any iteration per complexity, and the target complexities are in terms of elements of the Cartesian grid. This illustrates the effect of p-refinement as the meshes have the same number of elements regardless of degree.

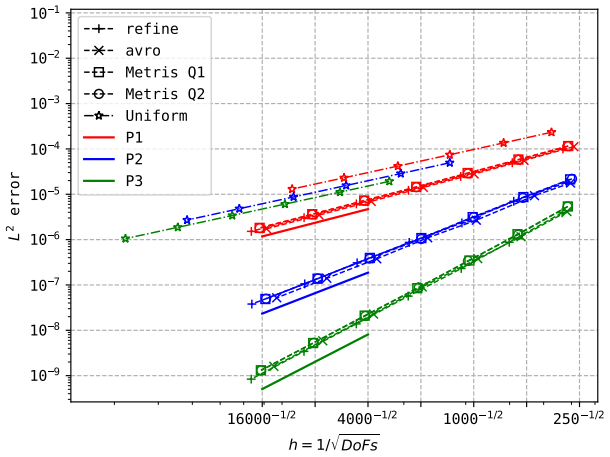
Metris is run in Q1 and Q2 mode, meaning providing linear and degree 2 meshes. Although the solver does exploit the mesh high-orderness, the error estimator (MOESS) does not. Hence, the metrics do not take into account mesh curvature explicitly, although the solver is influenced by it. Mesh curvature is brought in after adaptation. The input mesh is linear, adapted, and then the mesh is elevated to degree 2 and curved to the metric using Bézier gaps backtracked to validity to kickstart curving followed by conformity error minimization (mesh smoothing). We opted for backtracking instead of using the LP solvers as, in the absence of localized versions of the LP solvers, it remains considerably faster to backtrack and, in the absence of a boundary to curve to (as the domain is square), any lost curvature is rather easily recoverable through quality optimization.

Errors at each complexity are obtained by taking the average error of the last 10 adaptation-solver iterations at that complexity. Figures 8a, 9a, and 9b report error convergence for function CS, 2BL, respectively PG. The short full lines represent the expected order of convergence $p + 1$. Results for all three solver degrees are presented.

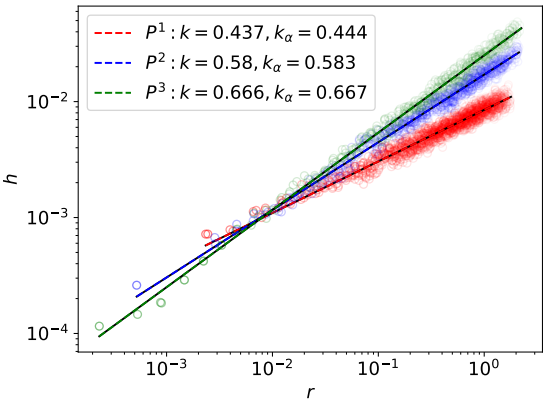
In all cases, uniform gris result in three to four orders of magnitude higher error. In the case of the corner singularity, the convergence speed on uniform meshes is also reduced. Figure 8b illustrates a scatter of element sizes with respect to r computed as the square root of the element area on an intermediate adapted mesh. The optimal element size as

a function of r (in polar coordinates) has been derived in [71], and this sizing distribution is necessary for optimal convergence speed. The coefficient k of Eq. (40) is computed from a regression (reported as k) and compared to the analytical value k_α, reported in Fig. 8b. Very good agreement — to two decimal places — is found, indicating MOESS automatically adapted to the optimal size repartition using Metris.

The adapted meshes all converge as fast as expected. Overall, there is very little difference in results between the meshers used. In particular, Q1 and Q2 meshes produced by Metris behave very similarly, their error curves being indistinguishable. On the one hand, this is good illustration that meshes adapted to an embedded family of metric fields are conforming in the terminology of [20], i.e. with curvature decreasing as $O(h^2)$ (for sizes $O(h)$) which preserves convergence speed despite fears that random curvature could, in this case, have halved convergence speed [27, 73]. On the other, we do not recover accelerated convergence speed on Q2 meshes as found in [32] for interpolation error or [31, 74] for L^2 projection error. Convergence could have been accelerated by between half and order and a full order, including for P1 projection or interpolation on Q2 meshes. Note however that such work is based either on direct optimization of the interpolation or projection error, or on modified MOESS methods that take into account mesh curvature explicitly [74], which is not the case here. We illustrate Q2 meshes adapted to the P2 solver at intermediate complexity DoFs $\simeq 4000$ in Figure 10.

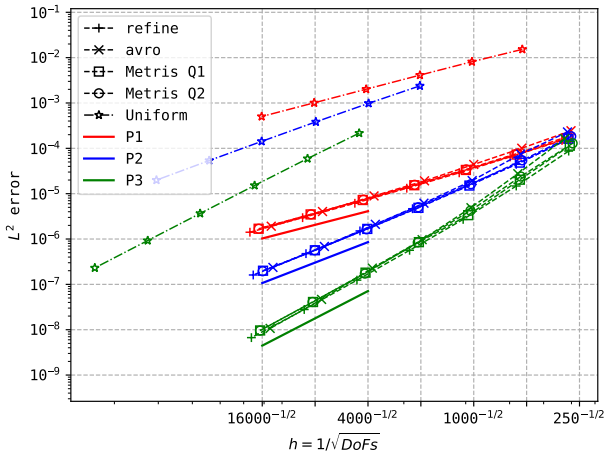


(a) Convergence of L^2 error.

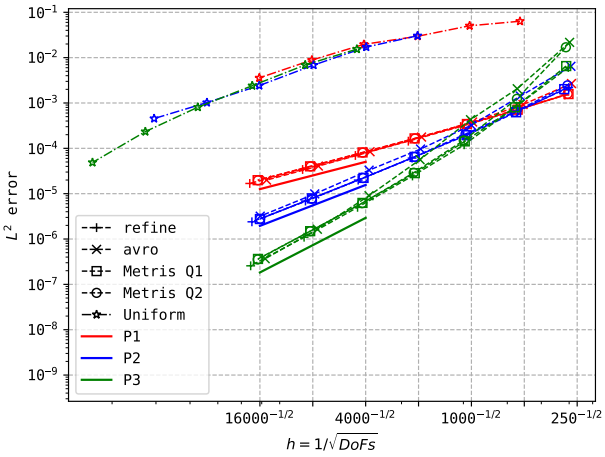


(b) Element size as function of distance to origin on intermediate mesh. Expected k_α and observed k slopes agree.

Fig. 8 Convergence and mesh statistics for function Corner

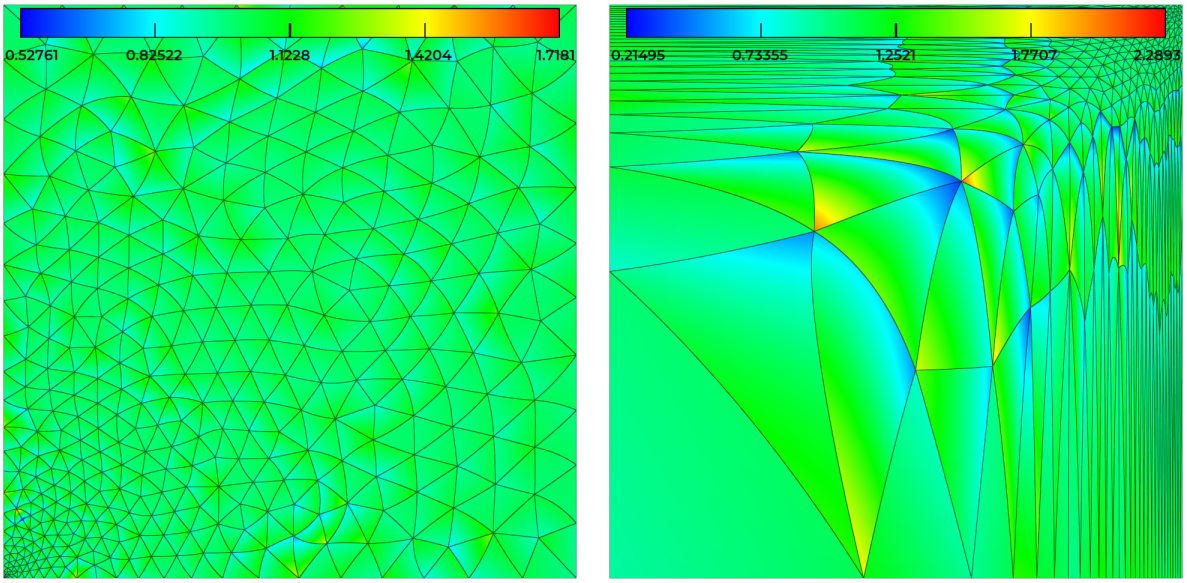


(a) Convergence of L^2 error of function Double BL.

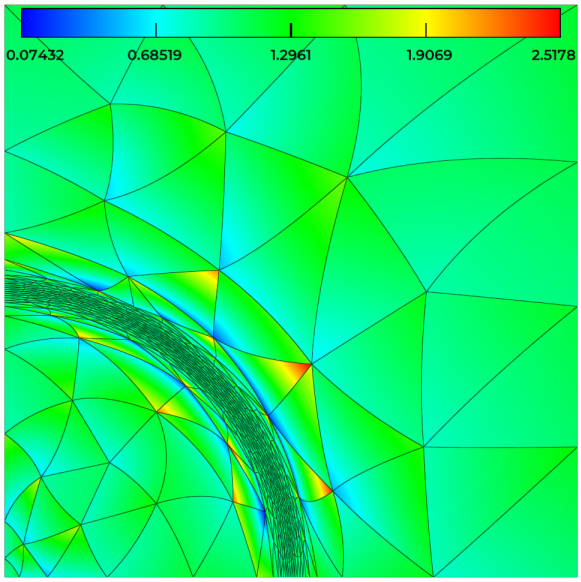


(b) Convergence of L^2 error of function Polar Gauss.

Fig. 9 Convergence of L^2 error for functions 2BL and PG.



(a) Function CS **(b) Function 2BL**



(c) Function PG

Fig. 10 Q2 Meshes adapted to L^2 projection error using MOESS and SANS as the solver with around 4000 **degrees of freedom adapted and curved with Metris. Represented field is scaled control coefficient.**

B. Transonic simulation on the ONERA OAT15A airfoil

We now consider a transonic RANS simulation at Mach number 0.73 with the SA-neg [75] turbulence model of a 2D ONERA OAT15A airfoil [76] with no-slip boundary conditions on the airfoil and far-field boundary conditions on the bounding box. The high-order VMSD-BR2 method [77] is used with MOESS adapting to the drag coefficient. Angles of attack $\alpha \in \{1.36, 1.5, 2.5, 3.0\}$ in degrees were chosen. Metris results are compared against the state-of-the-art adaptation software refine [40]. 40 adaptation iterations are carried out per target mesh complexity starting at 2,000 points and doubling until 64,000 in the case of refine and 128,000 in the case of Metris.

As this case lacks an analytical solution, convergence of the drag and lift coefficients is compared between meshers used, in Figure 11. Full lines represent lift (C_L) and dashed lines drag (C_D). Only the first and last angles of attack are

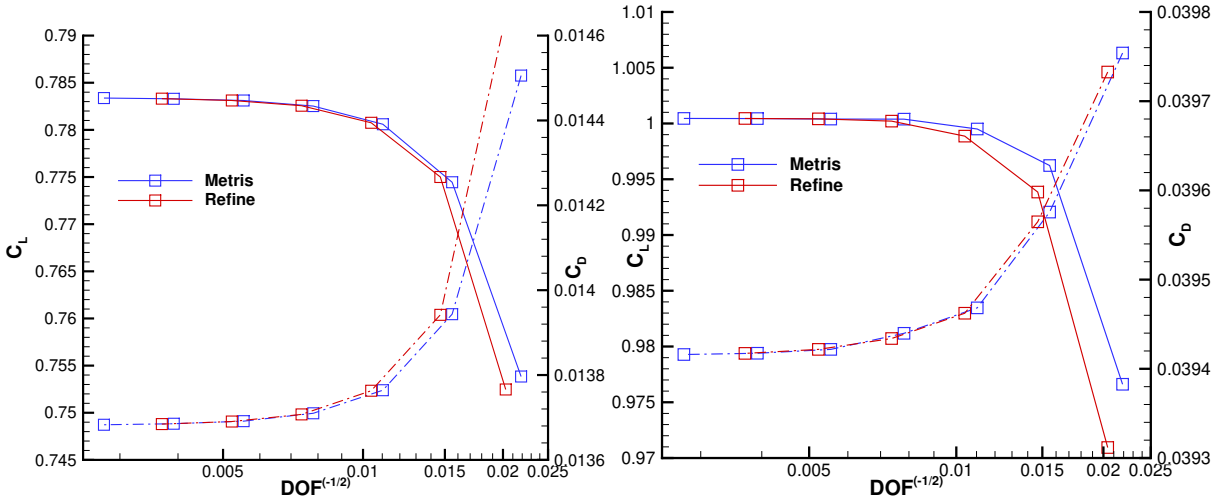
illustrated here for brevity. MOESS is driving adaptation using drag tangential to the flow as the quantity of interest, hence success of the overall adapted solve is measured in how accurately this coefficient is captured. We lack accurate reference values, but Metris is found to consistently help converge faster to the limit values than refine, across all angles of attack, and for both lift and drag.

We illustrate the adapted meshes using the $\alpha = 3°$ case and a before-last complexity mesh at 65080 vertices and 129096 triangles. Figures 12 and 13 present overviews of the mesh from afar and close to the geometry. Both the wake and flow leading to the geometry require high resolution. From closer, a lambda-shock is visible near the middle of the airfoil, as well as a large lambda-shaped adjoint feature, and boundary layers. Figure 14 offers a close-up of the lambda-shock blending into the boundary layer. The shock is not overly anisotropic at this resolution, anisotropic ratios in the other of 100 are observed from metric field eigenvalues. Figure 15 illustrates smaller scale adjoint features mirroring the large lambda-shaped feature. These are a necessity for the convergence of the error estimate. Figure 16 illustrates the blunt trailing edge. Much as in the classic cases of the reentrant corner, or the corner singularity illustrated in the previous Subsection, additional resolution is needed at the sharp corners, which is automatically captured by MOESS. Lastly Figure 17 illustrates the leading edge (mesh rotated by 90°) and boundary layer formation slightly away from the wall, as had previously been observed [78].

Final Metris adapted mesh quality metrics are reported in Table 3 for each considered angle of attack α. MOESS proceeds iteratively and relies on modifications of the current mesh-implied metric, hence it requires that very high percent unit edges be achieved, as close to 100% as possible. For this reason, Metris is run with considerable smoothing and adaptation iterations when coupled with SANS. In this case, maximum 50 adaptation iterations at the last run of a given complexity and 20 mesh optimization (smoothing and swapping) iterations. In all cases, percent quasi-unit edges achieved is at or higher than 99.64%. A slight bias to long edges is observed, as average lengths are around 1.01 in all cases. This is mostly due to insertion rejections on the basis of new low height or low quality (high conformity error) elements, but the bias remains very weak. Low conformity error (or high quality) is observed on all meshes, both in average and in maximum quality. We recall that conformity error and quality relate as: $E = (1 - Q)^p$ with the quality function $Q \in [0, 1]$. We used here $p = 2$ and, at mesh degree 1, the error is an average of the quality at each element's nodes. It is common to consider instead the quality function $1/Q \in [1, +\infty[$. In that case, the values corresponding to 0.31, 0.25, 0.27 and 0.29 conformity error are qualities of, respectively, 2.26, 2.00, 2.08, and 2.17, setting aside the fact they are averages of three values.

Table 3 Statistics of final Metris adapted meshes of ONERA OAT15A for each angle of attack α: percent quasi-unit edges, average edge length in the metric, average metric conformity error (from 0 to 1) per element, maximum element conformity error.

α	% unit	avg. len.	avg. $E_{\mathcal{M}}(K)$	max. $E_{\mathcal{M}}(K)$
1.36	99.65	1.012	0.0085	0.31
1.50	99.65	1.010	0.0086	0.25
2.50	99.64	1.011	0.0084	0.27
3.00	99.67	1.012	0.0083	0.29



(a) $\alpha = 1.36°$ **(b)** $\alpha = 3.0°$

Fig. 11 Lift C_L (full) and drag C_D (dashed) coefficients convergence on the ONERA OAT15A transonic case for angles of attack $\alpha \in \{1.36, 3.1\}$ with Metris and refine for adaptation.

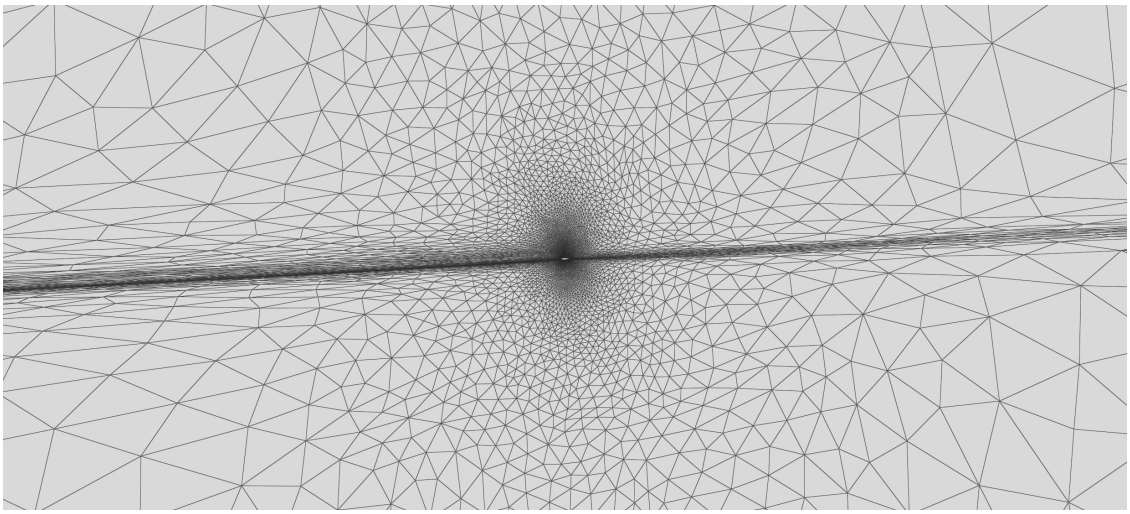


Fig. 12 Final transonic ONERA OAT15A mesh for $\alpha = 3.0°$: overview.

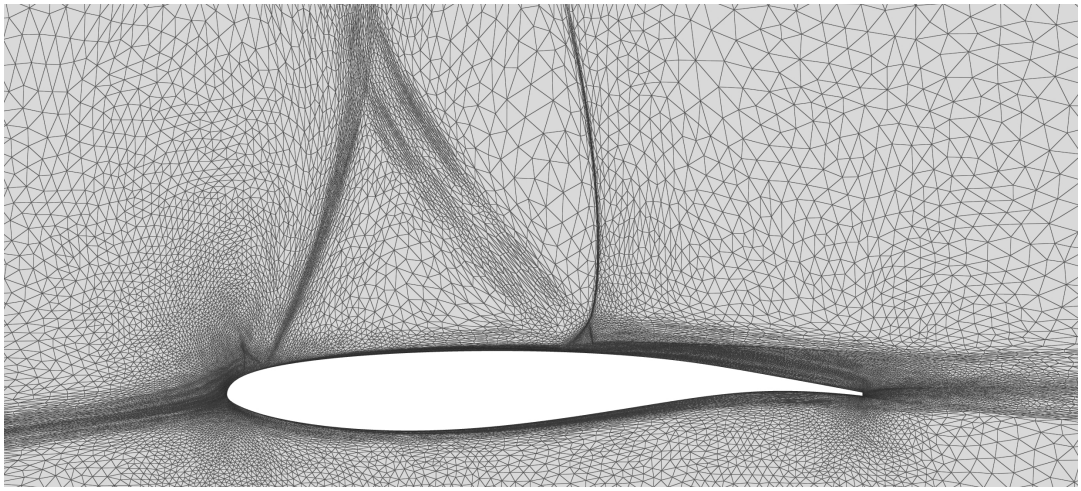


Fig. 13 Final transonic ONERA OAT15A mesh for $\alpha = 3.0°$: close-up.

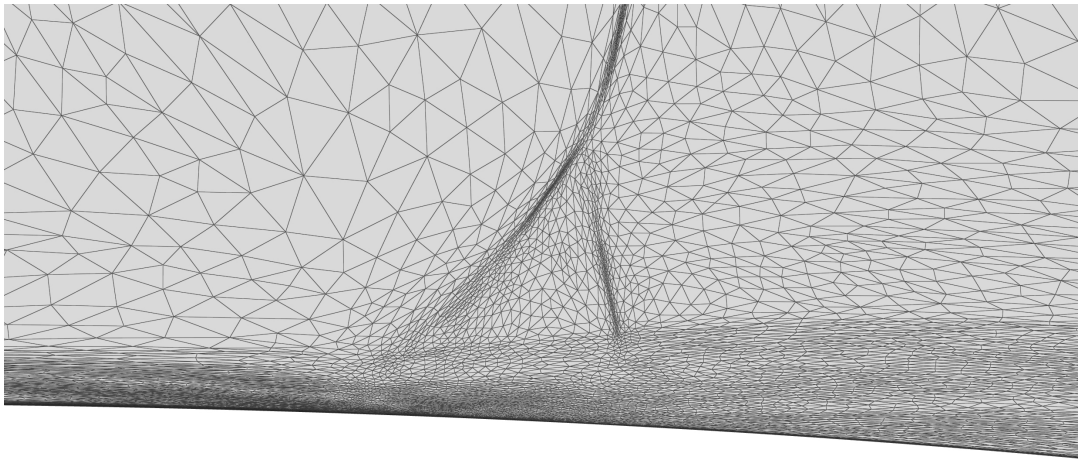


Fig. 14 Final transonic ONERA OAT15A mesh for $\alpha = 3.0°$: rear lambda shock.

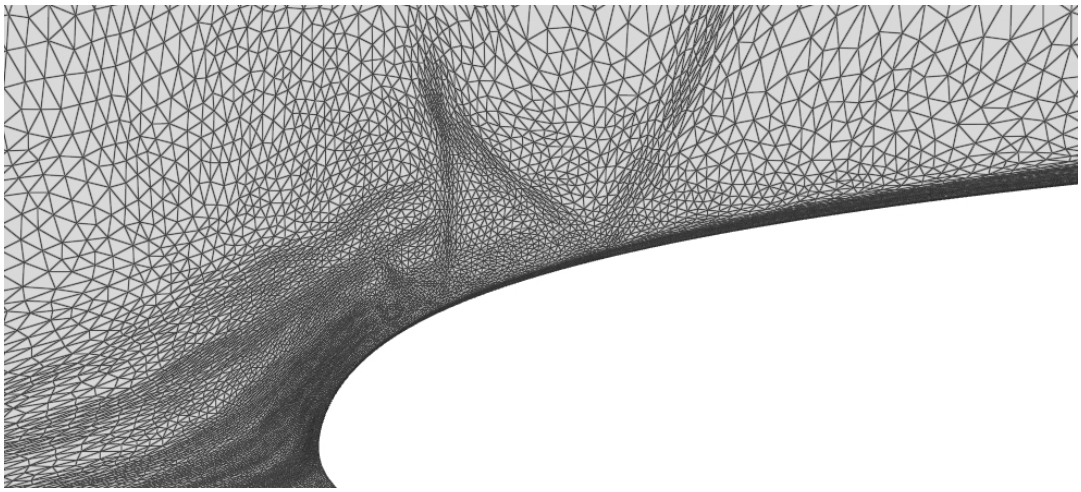


Fig. 15 Final transonic ONERA OAT15A mesh for $\alpha = 3.0°$: adjoint features.

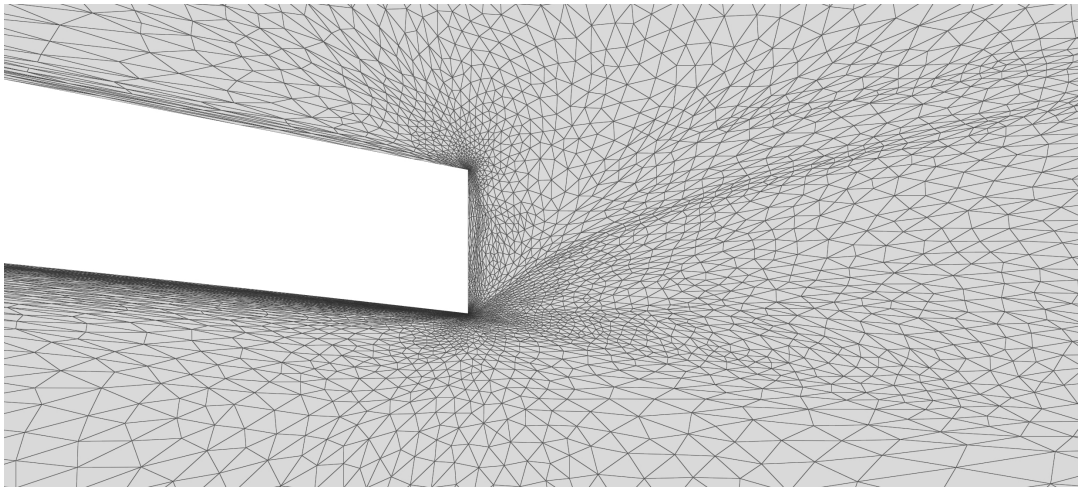


Fig. 16 Final transonic ONERA OAT15A mesh for $\alpha = 3.0°$: sharp trailing edge with corner singularity resolution.

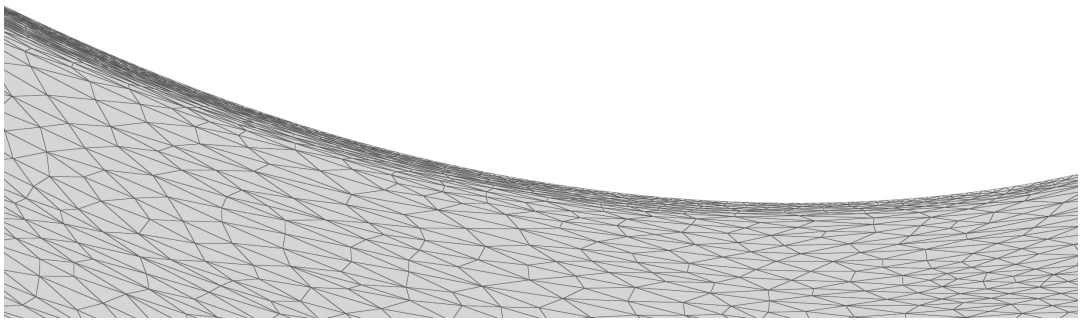


Fig. 17 Final transonic ONERA OAT15A mesh for $\alpha = 3.0°$: leading edge. Highest anisotropy starts slightly away from wall.

VI. Conclusion

This paper introduced Metris, a new metric-based mesh adaptation software that handles high-order meshes. Metric fields prescribe sizes and anisotropy as in other similar software, but also mesh curvature, which presents a novelty. Examples of adapted and curved meshes are demonstrated in 2D. Metris performance was shown to be comparable on a synthetic case (L^2 error minimization of L^2 projection of an analytical function) to similar freely-available software, namely refine and avro. Metris was also compared to refine on a transonic RANS simulation of the ONERA OAT15A airfoil where the mesh is adapted to minimize numerical error in the drag coefficient using the solver SANS and MOESS algorithm. Faster convergence of drag and lift was obtained using Metris than refine, in this case. Final Metris meshes are shown to be of very high quality, with at least 99.64% edges unit, and quality (in range $[1, +\infty[$) under 2.26 for the worst elements. This code is freely available on GitHub [§] and currently implements 2D adaptation and curving of P^2 meshes, which development for 3D underway.

Future work All topological changes in Metris rely on the cavity operator. The most urgent to support 3D adaptation is to implement tetrahedral cavity reconnection in this operator. 3D meshes are already partially supported although surface operations have not been extensively tested yet.

On degree 2 meshes, the cavity operator can currently verify validity rigorously using control coefficients but cannot yet curve the final cavities beyond placing new nodes on CAD geometry. To this end, it will be necessary to localize the

[§] `https://github.com/LucienRochery/Metris`

LP-based correction and curving approaches and possibly implement a tailored solver to reach maximum performance. In another context, control coefficient maximization of the tetrahedral shell of an edge has been shown possible in roughly as long time as it takes to evaluate the control coefficients 7 times [35]. Currently, the LP implementation, relying on external generic sparse solvers, does not meet that performance. The cavity operator is certainly the most called non-trivial function within an adaptation code, hence it needs to be as fast as possible.

Regardless of how fast Bézier offsets setting via LPs can be, there remains a use for differentiable optimization in curving meshes. The Bézier offsets approach relies on a coarse discretization of the unit relationship that relates an element's Jacobian matrix to the metric field. Essentially, it requires an order 1 (for a degree 2 mesh) Taylor expansion of $\mathcal{M}^{-1/2} \circ F_K$, with F_K the element mapping. The resulting fixed-point problem has been shown to converge very rapidly [38] and is not the most fundamental issue. This is rather that such an approach is only good for approximate unitness, but cannot reach the standards of mesh conformity that continuous optimization offers. It remains uncertain whether, on fine meshes, this approximate curved unitness will prove enough for any error gains. At any rate, coarse meshes will continue to benefit from mesh smoothing.

The currently used quality (or metric conformity error) function presents some theoretical issues. The high-order unit relationship relates an element's Jacobian matrix $\mathcal{J}_K$ and the metric field $\mathcal{M}^{-1/2}$ by a pointwise relationship: for all ξ, $\mathcal{J}_K(\xi) = \mathcal{M}^{-1/2}(F_K(\xi))R\mathcal{J}_0$, where R is a rotation matrix and $\mathcal{J}_0$ the ideal element Jacobian matrix. At the optimum, the quality function only guarantees the following relationship: for all ξ, there exist $\lambda(\xi)$ and $R(\xi)$ such that $\mathcal{J}_K(\xi) = \lambda(\xi)\mathcal{M}^{-1/2}(F_K(\xi))R(\xi)\mathcal{J}_0$. The invariance by scaling is not the issue (this is shared with linear meshing), the problem is the non-uniformity of the rotation matrix. Hence these functions need to be modified to either be not invariant to rotation, or invariant only to constant rotation per element.

Element curvature on the boundary is still very rudimentary: the t coordinate of a high-order node on the geometry is the average of its edge's endpoints, and the vertex is then evaluated on the geometry. It is urgent to implement node placement on geometry that takes into account geometry error. Not only will this avoid robustness issues, proper node placement on geometry can achieve super-convergence of geometry-related errors [29]. Similarly, even for linear meshes, it might be desirable to implement geometric error control via metric field intersection and grading using metric fields for boundary approximation [12] which exist also for high-order meshes [13].

Lastly, super-convergence of P^2 interpolation error or L^2 projection error on P^2 bases has been observed [30–32], but we currently lack either optimal metrics derived from a priori estimates, or from a posteriori estimation as in MOESS, that take into account mesh curvature. Hence an important topic of work to fully utilize high-order meshing remains error estimation.

Acknowledgments

This work was funded by the EnCAPS project, AFRL Contract FA8650–20–2–2002: "EnCAPS: Enhanced Computational Aircraft Prototype Syntheses", with Dr. Joshua Deslich as the Technical Monitor. Funding was also provided by Oak Ridge National Laboratories under DOE Contract 4000192102 with Dr. Ryan Glasby as the Technical Monitor.

References

[1] Alauzet, F., and Frazza, L., "Feature-based and goal-oriented anisotropic mesh adaptation for RANS applications in aeronautics and aerospace," *Journal of Computational Physics*, Vol. 439, 2021, p. 110340. https://doi.org/10.1016/j.jcp.2021.110340.

[2] 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. https://doi.org/10.1137/090754078.

[3] Yano, M., and Darmofal, D. L., "An Optimization-Based Framework for Anisotropic Simplex Mesh Adaptation," Vol. 231, No. 22, 2012, pp. 7626–7649. https://doi.org/10.1016/j.jcp.2012.06.040.

[4] Carson, H. A., Huang, A. C., Galbraith, M. C., Allmaras, S. R., and Darmofal, D. L., "Anisotropic mesh adaptation for continuous finite element discretization through mesh optimization via error sampling and synthesis," Vol. 420, 2020, p. 109620. https://doi.org/10.1016/j.jcp.2020.109620.

[5] Carson, H. A., Huang, A. C., Galbraith, M. C., Allmaras, S. R., and Darmofal, D. L., "Mesh optimization via error sampling and synthesis: An update," AIAA 2020–0087, January 2020. https://doi.org/10.2514/6.2020-0087.

[6] Carson, H. A., "Provably Convergent Anisotropic Output-Based Adaptation for Continuous Finite Element Discretizations," PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Nov. 2019. URL https://dspace.mit.edu/handle/1721.1/129891.

[7] Alauzet, F., and Loseille, A., "High-order sonic boom modeling based on adaptive methods," *Journal of Computational Physics*, Vol. 229, No. 3, 2010, pp. 561–593. https://doi.org/10.1016/j.jcp.2009.09.020.

[8] 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. https://doi.org/10.1038/nature24671.

[9] Frey, P.-J., and Alauzet, F., "Anisotropic mesh adaptation for CFD computations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 194, No. 48-49, 2005, pp. 5068–5082. https://doi.org/10.1016/j.cma.2004.11.025.

[10] Carson, H. A., Huang, A. C., Galbraith, M. C., Allmaras, S. R., and Darmofal, D. L., "Mesh Optimization via Error Sampling and Synthesis: An Update," *AIAA SciTech 2020 Forum*, American Institute of Aeronautics and Astronautics, Orlando, FL, 2020. https://doi.org/10.2514/6.2020-0087, URL https://arc.aiaa.org/doi/10.2514/6.2020-0087.

[11] 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/10.1016/j.jcp.2009.12.021.

[12] Frey, P., "About Surface Remeshing," *9th International Meshing Roundtable*, Sandia National Laboratories, 2000.

[13] 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.

[14] 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.

[15] Loseille, A., Menier, V., and Alauzet, F., "Parallel generation of large-size adapted meshes," *Procedia Engineering*, Vol. 124, 2015, pp. 57–69.

[16] Michal, T., Krakos, J., Kamenetskiy, D., Galbraith, M., Ursachi, C.-I., Park, M. A., Anderson, W. K., Alauzet, F., and Loseille, A., "Comparing Unstructured Adaptive Mesh Solutions for the High Lift Common Research Airfoil," *AIAA Journal*, Vol. 59, No. 9, 2021, pp. 3566–3584.

[17] Alauzet, F., Loseille, A., Marcum, D., and Michal, T. R., "Assessment of anisotropic mesh adaptation for high-lift prediction of the HL-CRM configuration," *23rd AIAA Computational Fluid Dynamics Conference*, 2017, p. 3300. https://doi.org/10.2514/6.2017-3300.

[18] Alauzet, F., Clerici, F., Loseille, A., Maunoury, M., Rochery, L., Tarsia-Morisco, C., Tenkes, L.-M., and Vanharen, J., "4th AIAA CFD High Lift Prediction Workshop results using metric-based anisotropic mesh adaptation," *AIAA Aviation 2022 Forum*, 2022, p. 3521. https://doi.org/10.2514/6.2022-3521.

[19] Ergatoudis, I., Irons, B. M., and Zienkiewicz, O. C., "Curved, isoparametric, 'quadrilateral' elements for finite element analysis," *International Journal of Solids and Structures*, Vol. 4, No. 1, 1968, pp. 31–42. https://doi.org/10.1016/0020-7683(68)90031-0, URL https://www.sciencedirect.com/science/article/pii/0020768368900310.

[20] Ciarlet, P. G., "Isoparametric finite elements and Applications to second order problems over curved domains," *The finite element method for elliptic problems*, No. v. 4 in Studies in mathematics and its applications, North-Holland Pub. Co. ; sole distributors for the U.S.A. and Canada, Elsevier North-Holland, Amsterdam ; New York : New York, 1978, pp. 224–286.

[21] 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. https://doi.org/10.1002/nme.4547.

[22] Wang, Z., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H., Kroll, N., May, G., Persson, P.-O., van Leer, B., and Visbal, M., "High-order CFD methods: current status and perspective," *International Journal for Numerical Methods in Fluids*, Vol. 72, No. 8, 2013, pp. 811–845. https://doi.org/10.1002/fld.3767, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.3767, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.3767.

[23] 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. https://doi.org/10.1006/jcph.1997.5454.

[24] Ciarlet, P., 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. https://doi.org/10.1016/B978-0-12-068650-6.50020-4.

[25] 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. https://doi.org/10.1137/0723036.

[26] 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.

[27] 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.

[28] 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.

[29] Docampo-Sánchez, J., Ruiz-Gironés, E., and Roca, X., "An Efficient Solver to Approximate CAD Curves with Super-Convergent Rates," 2022. https://doi.org/https://doi.org/10.5281/zenodo.6562447.

[30] Sanjaya, D. P., and Fidkowski, K., "High-Order Node Movement Discretization Error Control in Shape Optimization," *AIAA SCITECH 2023 Forum*, 2023, p. 2367.

[31] Bawin, A., "Metric-based mesh adaptation with curvilinear triangles," Ph.D. thesis, UCL-Université Catholique de Louvain, 2024.

[32] Rochery, L., "Adaptation courbe de maillages sous champ de métriques anisotropes," PhD Thesis, Inria Saclay - Université Paris-Saclay, 2023.

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

[34] Bawin, A., Garon, A., and Remacle, J.-F., "Optimally convergent isoparametric P2 mesh generation," *International Meshing Roundtable*, Springer, 2018.

[35] Rochery, L., and Loseille, A., "Fast High-Order Mesh Correction for Metric-Based Cavity Remeshing and a Posteriori Curving of P2 Tetrahedral Meshes," *Computer-Aided Design*, Vol. 163, 2023, p. 103575. https://doi.org/https://doi.org/10.1016/j.cad.2023.103575, URL https://www.sciencedirect.com/science/article/pii/S0010448523001070.

[36] Aparicio-Estrems, G., Gargallo-Peiró, A., and Roca, X., "High-Order Metric Interpolation for Curved R-Adaption by Distortion Minimization," *SIAM International Meshing Roundtable*, Zenodo, 2022. https://doi.org/10.5281/zenodo.6562456, URL 10.5281/zenodo.6562456.

[37] Sanjaya, D., "Towards Automated, Metric-Conforming, Mesh Optimization for High-Order, Finite-Element Methods," PhD Thesis, 2019.

[38] Rochery, L., Galbraith, M. C., Darmofal, D. L., and Allmaras, S., "A Generalized Continuous Mesh Framework for Explicit Mesh Curving," *AIAA SCITECH 2024 Forum*, 2024, p. 0787.

[39] Ciarlet, P., and Raviart, P.-A., "Interpolation theory over curved elements, with applications to finite element methods," *Computer Methods in Applied Mechanics and Engineering*, Vol. 1, No. 2, 1972, pp. 217–249.

[40] Park, M. A., "Anisotropic output-based adaptation with tetrahedral cut cells for compressible flows," Ph.D. thesis, Massachusetts Institute of Technology, 2008.

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

[42] Caplan, P. C., Haimes, R., Darmofal, D. L., and Galbraith, M. C., "Extension of local cavity operators to $3d + t$ spacetime mesh adaptation," AIAA 2019-1992, January 2019. https://doi.org/10.2514/6.2019-1992.

[43] Michal, T., and Krakos, J., "Anisotropic mesh adaptation through edge primitive operations," *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2012, p. 159.

[44] George, P.-L., 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.

[45] Galbraith, M. C., Allmaras, S. R., and Darmofal, D. L., "A verification driven process for rapid development of CFD software," AIAA 2015-0818, January 2015. https://doi.org/10.2514/6.2015-0818.

[46] 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. https://doi.org/10.1137/10078654X.

[47] George, P. L., Borouchaki, H., Alauzet, F., Laug, P., Loseille, A., and Marechal, L., *Meshing, Geometric Modeling and Numerical Simulation, Volume 2: Metrics, Meshes and Mesh Adaptation*, John Wiley & Sons, 2019.

[48] Dompierre, J., Vallet, M.-G., Bourgault, Y., Fortin, M., and Habashi, W. G., "Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part III. Unstructured meshes," Vol. 39, No. 8, 2002, pp. 675–702. https://doi.org/10.1002/fld.357, URL http://dx.doi.org/10.1002/fld.357.

[49] Coupez, T., Jannoun, G., Veysset, J., and Hachem, E., *Edge-Based Anisotropic Mesh Adaptation for CFD Applications*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 567–583. https://doi.org/10.1007/978-3-642-33573-0_33, URL http://dx.doi.org/10.1007/978-3-642-33573-0_33.

[50] Loseille, A., *Recent Improvements on Cavity-Based Operators for RANS Mesh Adaptation*, 2018. https://doi.org/10.2514/6.2018-0922, URL https://arc.aiaa.org/doi/abs/10.2514/6.2018-0922.

[51] Michal, T., and Krakos, J., "Anisotropic mesh adaptation through edge primitive operations," AIAA 2012–159, January 2012. https://doi.org/10.2514/6.2012-159.

[52] Dompierre, J., Vallet, M. G., Labbé, P., and Guibault, F., "An analysis of simplex shape measures for anisotropic meshes," *Computer Methods in Applied Mechanics and Engineering*, Vol. 194, No. 48, 2005, pp. 4895–4914. https://doi.org/10.1016/j.cma.2004.11.018, URL https://www.sciencedirect.com/science/article/pii/S0045782505000721.

[53] Chen, L., "Mesh Smoothing Schemes Based On Optimal Delaunay Triangulations," *13th International Meshing Roundtable*, 2004, pp. 109–120.

[54] 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.

[55] Feuillet, R., "Embedded and high-order meshes : two alternatives to linear body-fitted meshes," Ph.D. thesis, 2019. URL http://www.theses.fr/2019SACLY010/document, thèse de doctorat dirigée par Ciarlet, Patrick et Alauzet, Frédéric.

[56] Cirrottola, L., and Froehly, A., "Parallel unstructured mesh adaptation based on iterative remershing and repartitioning," *WCCM-Eccomas 2020-14th World Congress on Computational Mechanic*, 2021.

[57] Gorman, G. J., Rokos, G., Southern, J., and Kelly, P. H., "Thread-parallel anisotropic mesh adaptation," *New challenges in grid generation and adaptivity for scientific computing*, 2015, pp. 113–137.

[58] Alauzet, F., and Marcum, D., "Metric-aligned and metric-orthogonal strategies in AFLR," *23rd AIAA Computational Fluid Dynamics Conference*, 2017, p. 3108.

[59] Borouchaki, H., George, P.-L., Hecht, F., Laug, P., and Saltel, E., "Mailleur bidimensionnel de Delaunay gouverné par une carte de métriques. Partie I: Algorithmes," Ph.D. thesis, INRIA, 1995.

[60] Loseille, A., and Rochery, L., "Developments on the P2 cavity operator and Bézier Jacobian correction using the simplex algorithm." *AIAA SciTech 2022 Forum*, 2022, p. 0389. https://doi.org/10.2514/6.2022-0389.

[61] Galbraith, M. C., Allmaras, S. R., and Haimes, R., "Full potential revisited: A medium fidelty aerodynamic analysis tool," *55th AIAA Aerospace Sciences Meeting*, 2017, p. 0290.

[62] Haimes, R., and Dannenhoffer, J., "EGADSlite: A Lightweight Geometry Kernel for HPC," *2018 AIAA Aerospace Sciences Meeting*, 2018, p. 1401. https://doi.org/10.2514/6.2018-1401.

[63] Maréchal, L., "libMeshb," https://github.com/LoicMarechal/libMeshb, 2017.

[64] 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/10.1002/mrm.20965.

[65] Aparicio-Estrems, G., Gargallo-Peiró, A., and Roca, X., "Combining High-Order Metric Interpolation and Geometry Implicitization for Curved r-Adaption," *Computer-Aided Design*, Vol. 157, 2023, p. 103478. https://doi.org/https://doi.org/10.1016/j.cad.2023.103478, URL https://www.sciencedirect.com/science/article/pii/S0010448523000106.

[66] Schoof, L. A., and Yarberry, V. R., "EXODUS II: a finite element data model," Tech. rep., Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 1994.

[67] Boost, "Boost C++ Libraries," http://www.boost.org/, 2015. Last accessed 2015-06-30.

[68] Coppeans, A., Fidkowski, K., and Martins, J. R., "Anisotropic Mesh Adaptation for High-Order Meshes in Two Dimensions," *AIAA SCITECH 2024 Forum*, 2024, p. 1020.

[69] Bochkanov, S., and Bystritsky, V., "Alglib," *Available from: www alglib net*, Vol. 59, 2013, p. 833.

[70] Forrest, J., Vigerske, S., Ralphs, T., Forrest, J., Hafer, L., jpfasano, Santos, H. G., Jan-Willem, Saltzman, M., a andre, Kristjansson, B., h-i gassmann, King, A., Arevall, Mart, B., Bonami, P., Luies, R., Brito, S., and to st, "coin-or/Clp: Release releases/1.17.10," , Aug. 2024. https://doi.org/10.5281/zenodo.13347196, URL https://doi.org/10.5281/zenodo.13347196.

[71] Yano, M., "An Optimization Framework for Adaptive Higher-Order Discretizations of Partial Differential Equations on Anisotropic Simplex Meshes," PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Jun. 2012. URL http://hdl.handle.net/1721.1/76090.

[72] Caplan, P. C., "Four-dimensional Anisotropic Mesh Adaptation for Spacetime Numerical Simulations," PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Jun. 2019. URL https://hdl.handle.net/1721.1/122367.

[73] Botti, L., and Di Pietro, D. A., "Assessment of hybrid high-order methods on curved meshes and comparison with discontinuous Galerkin methods," *Journal of Computational Physics*, Vol. 370, 2018, pp. 58–84. https://doi.org/10.1016/j.jcp.2018.05.017.

[74] Sanjaya, D. P., Fidkowski, K., and Murman, S. M., "Comparison of Algorithms for High-Order, Metric-Based Mesh Optimization," *AIAA SciTech 2020 Forum*, 2020, p. 1141.

[75] Allmaras, S. R., Johnson, F. T., and Spalart, P. R., "Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model," Seventh International Conference on Computational Fluid Dynamics ICCFD7-1902, Big Island, Hawaii, 2012. URL https://www.iccfd.org/iccfd7/assets/pdf/papers/ICCFD7-1902_paper.pdf.

[76] Jacquin, L., Molton, P., Deck, S., Maury, B., and Soulevant, D., "Experimental study of shock oscillation over a transonic supercritical profile," *AIAA journal*, Vol. 47, No. 9, 2009, pp. 1985–1994.

[77] Huang, A. C., Carson, H. A., Allmaras, S. R., Galbraith, M. C., Darmofal, D. L., and Kamenetskiy, D. S., "A variational multiscale method with discontinuous subscales for output-based adaptation of aerodynamic flows," *AIAA Scitech 2020 Forum*, 2020, p. 1563.

[78] Hu, Y., Wagner, C., Allmaras, S. R., Galbraith, M. C., and Darmofal, D. L., "Application of higher–order adaptive method to Reynolds–Averaged Navier–Stokes test cases," *AIAA Journal*, Vol. 54, No. 9, 2016, pp. 2626–2644. https://doi.org/10.2514/1.J054558.