P^3 BÉZIER CAD SURROGATES FOR ANISOTROPIC MESH ADAPTATION

A. Loseille[1] L. Rochery[2]

[1] *Inria Saclay, 91120 Palaiseau, France, adrien.loseille@inria.fr*
[2] *Inria Saclay, 91120 Palaiseau, France, lucien.rochery@inria.fr*

ABSTRACT

Mesh generation and adaptation rely heavily on BREPs created by proprietary CAD software, piecewise parametric descriptions of geometry from which numerous problems arise: model continuity is only enforced up to a tolerance often higher than required mesh sizes, projection is costly and prone to error, derivatives — thus normals and curvature metrics — may not be well defined, unintended small features driving unnecessary mesh complexity may be present... unlike discrete surface meshes, of which high-order ones offer advantageous convergence speed over degree of freedom ratio relative to P^1 meshes. P^3 meshes, in particular, are the first degree for which $\mathcal{G}^1$ continuity at the vertices may be enforced. In this paper, we compare two methods to construct P^3 meshes from a CAD model. The resulting P^3 meshes are then used instead of the CAD model in a full converging adaptation loop on a complex geometry, the HL-CRM wing with flaps with a highly anisotropic metric field.

Keywords: BREP, CAD surrogate, P^3 Bézier triangles, anisotropic mesh adaptation, surface mesh generation

1. INTRODUCTION

Cost-effective numerical resolution of PDEs — namely of hyperbolic ones such as Navier-Stokes — is enabled by anisotropic mesh adaptation. Using either generic [1, 2] or PDE-tailored [3] error estimates, meshes are locally modified [4, 5] or the degree of interpolation is locally elevated (p-adaptation) [6, 7], sometimes both (hp-adaptation) [8, 9], to match local features of the solution and thus maximize the precision over degrees of freedom (computational cost) ratio [10, 11]. This places mesh generation and adaptation at the heart of simulation, which becomes a loop that converges to an optimal mesh-solution couple (Fig. 1).

Domain geometry is described in a continuous fashion using a CAD file. The BREP (Boundary REPresentation) model with rational Bézier patches and curves [12, 13], in particular, is widely used and the focus of this paper. In this framework, a surface is described as a collection of connected trimmed patches. Global

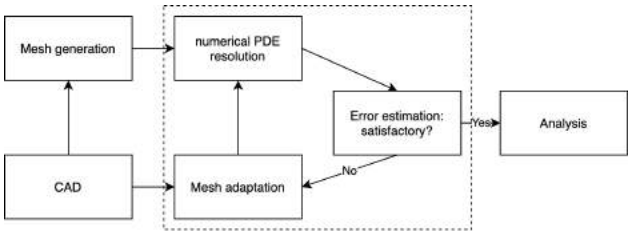

Figure 1: Mesh adaptation loop

BREP topology is illustrated in Fig. 2. This description using Bézier and rational Bézier curves and surfaces is quite flexible and can represent a wide variety of shapes with complex features, as well as represent exactly a number of frequently used geometric primitives such as sections of spheres, cylinders, cones... Some of the more frequent operations on these objects include evaluating a point on the surface given its parametric coordinates, projection of $3D$ points

onto CAD edges and faces and computing derivatives (up to the second order, most frequently) at given points on the surface. These operations are intensively used both in initial mesh generation as well as in subsequent adaptation steps, as illustrated in Fig. 1. BREP models are a tool specialized in defining shapes rather than manipulating them. For this reason, it is not infrequent for CAD models to pose a number of difficulties, such as by not being watertight, having face lines that degenerate into single points, autointersecting faces, interpenetrating neighbouring faces, ill-conditioned parameterizations, unintended small features, etc... [14, 15] go over the reasons for these features in detail. Furthermore, these errors are often bound by tolerances too high for the purposes of mesh adaptation where smaller edges may be required close to or on the surface [16].

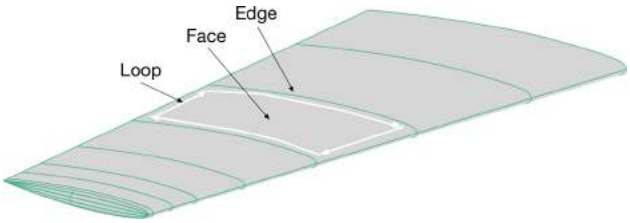

Figure 2: Trimmed BREP topology

Strategies to combat these issues operate at essentially two levels: by correcting the CAD before it is used in meshing, and by devising robust tessellation algorithms. By tessellation, we designate any surface mesh which will not necessarily be used as the support for any volume mesh, but rather as a discrete surrogate for the CAD geometry. Strategies of the first type include the use of virtual topology [17] and various CAD correction procedures [15].

In this paper, we present our approach of the second type, and illustrate the ability of P^3 surface meshes to drive mesh adaptation as well as the CAD itself on cases with high anisotropy close to the boundary. A P^1 tessellation of the parametric surface is first generated, and then elevated to the third order. A discrete tessellation offers fast inverse evaluation through simple algorithms, it can be seen as a first order Taylor expansion of the surface. Therefore, projection on a P^1 mesh is similar to gradient descent on the original surface with derivatives precomputed (Jacobians of surface elements). In fewer words, it is much simpler and stable, much of the burden being shifted to the tessellation step. This is only possible at the expense of precision, a cost that can be reduced by replacing linear elements by higher-order elements such as P^3 Bézier triangles. Through simple benchmarks, the speed and precision of projection on these meshes are exhibited. They are then used as the geometric support for a full adaptation loop using a highly anisotropic analytical boundary layer metric.

1.1 The BREP

Let us go over, in little detail, the elements that constitute a BREP-based CAD model. Further details can be found in the reference [13]. A model is given by a set of faces, loops, edges and corners. CAD edges are mapped from $1D$ domains and grouped into loops. These loops trim CAD faces which are mapped from $2D$ domains. In turn, faces sharing a loop portion are neighbours. Edges are defined using a $1D$ domain $D_1 \subset \mathbb{R}$ and a parameterization $\tau_3 : D_1 \mapsto \mathbb{R}^3$. The parameterization τ_3 is, typically, a rational Bézier curve although it is not infrequent for CAD systems to distinguish subcases such as particular conics and circle arcs. Faces are defined much in the same way, from a $2D$ domain $D_2 \subset \mathbb{R}^2$ and a parameterization $\sigma : D_2 \mapsto \mathbb{R}^3$. Likewise, it is typical for σ to be a rational Bézier surface, but particular cases such as sections of spheres and cylinders are sometimes distinguished. The face trimming is then defined using another parameterized curve defined on $D_1' \subset \mathbb{R}$ with a mapping $\tau_2 : D_1' \mapsto D_2$ that draws a $2D$ curve in the face's parameter domain.

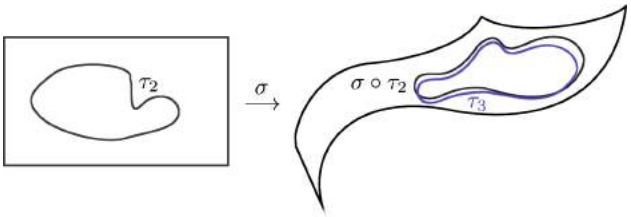

Figure 3: Trimming curve definition tolerances: τ_2 defined in surface patch parameter space is mapped to the black curve in $\mathbb{R}^3$ by σ, τ_3 maps a segment to the blue curve. These curves differ by a tolerance set by the CAD design tool (visible here as the exaggerated gap).

This curve is a representation in face parameter space of the physical face-face intersection curve. Unfortunately, one cannot assume that $\sigma \circ \tau_2 = \tau_3$. This is due to the fact that these trimming curves are computed from rational Bézier surface intersections which, in the general case, are not rational Bézier curves. This means that τ_3 is, typically, already an approximation of the desired curve. On top of this, τ_2 is, again, an approximation of the projection of τ_3 onto the face parameter space D_2. As such, CAD software only guarantees correspondence between the face-local and global curves up to a tolerance, typically much higher than desired mesh sizes. This means that despite the fact that parametric surfaces and curves are abritrarily precise, they can be very inaccurate in some regions (junctions), where the geometry is actually ill-defined

under a threshold. Concretely, this means that there can actually be gaps between faces meant to share an edge, or that neighbouring faces may be interpenetrating at scales that cannot be neglected by the meshing algorithm. The paper [16] goes over this issue in great detail, illustrating the impact on mesh adaptation for CFD problems. For instance, it is frequent for wing and fuselage intersections to be given with a tolerance several orders of magnitude higher than the smallest prescribed mesh size in this area by the end of adaptation. Furthermore, derivatives of the parameterizations are used to compute metric fields for surface error approximation [18, 19] or surface normals and tangent planes. Once more, the CAD description may pose problems such as by having faces map portions of edges onto points (degeneracy) or having unwanted local features such as folds under the tolerance (another issue pointed out in [16]). In particular, derivatives are not defined at these points and unstable in their vicinity. This also means that these situations call for robust optimization algorithms when projecting points close to these regions and that projection is slow (relative to on a discrete mesh) and prone to error.

2. THE PARAMETRIC P^3 MESHER

In this section we turn to the parametric meshing algorithm. It takes a CAD object as input and outputs a P^1 mesh adapted with regards to a geometric approximation metric. The P^3 mesh is then constructed by elevating the degree of these P^1 elements and projecting control nodes.

2.1 Building the initial tessellation

In this section, we give a rough description of a parametric surface mesher. Let us consider a trimmed CAD face F. It is defined by a rectangular parametric domain $D \subset \mathbb{R}^2$, a rational Bézier function $\sigma : D \mapsto \mathbb{R}^3$, and a set of connected rational Bézier edges $E_i \subset \mathbb{R}^3$ forming a closed loop. These edges E_i are constructed by the CAD system on path intersections and are an approximation of the actual intersection. Indeed, two rational surfaces need not intersect at a rational curve. The projection of these edges on D is also given by the CAD system as a two-dimensional rational Bézier cuve lying in D. If we denote $\tau_i : [0,1] \mapsto \mathbb{R}^3$ the parametrization of E_i in physical space and $\tau_i^{(2)} : [0,1] \mapsto D$ the parametrization of the given projection of E_i onto D, the identity

$$\sigma_i \circ \tau_i^{(2)} = \tau_i$$

does not hold in the general case.

The first step of our method is to produce a P^1 mesh of the trimming loop $\bigcup E_i$ in physical space. This step proceeds on a per-edge basis and computes an edge approximation error metric [18] to construct a mesh with quasi-uniform geometric error under the prescribed tolerance (user input). This mesh gives a set of vertices P_i and edges $e_i = [P_i, P_{i+1}]$ in $\mathbb{R}^3$. These vertices are then projected onto D, giving $P_i^{(2)}$ s.t. $P_i = \sigma(P_i^{(2)})$. The edges between these projected points form a closed loop in D. We use this as the trimming curve instead of $\tau_i^{(2)}$. In doing so we guarantee that, if a given CAD edge is part of the trimming loops of two edge faces, it will be mapped by both parametrizations to the same physical line mesh.

A constrained Delaunay mesher in parametric domain D is then called with this boundary as input. This first tessellation is used to compute the surface approximation error metric in low [18] or in high order [19]. When constructing the P^1 mesh as support for the P^3 surface mesh, a P^3 geometric approximation metric is used to adapt the surface mesh. The trimmed patch tessellation can then be adapted to this metric field [20, 21].

2.2 The P^3 surface mesh

P^3 Bézier elements are defined by a set of control nodes, either the Lagrange nodes which we denote P^ℓ_{ijk} or the Bézier nodes P_{ijk}, with $(i,j,k) \in \widehat{K}^d = \{(i,j,k) \in \mathbb{N}^3, i+j+k = 3\}$. In the latter case, the mapping from the reference triangle $\widehat{K}$ is given by

$$F_K(\xi) = \sum_{\alpha \in \widehat{K}^d} B_\alpha(\xi) P_\alpha,$$

whereas in the former, F_K is interpolated exactly by the degree three Lagrange basis $(\phi_\alpha)_{\alpha \in \widehat{K}^d}$,

$$F_K(\xi) = \sum_{\alpha \in \widehat{K}^d} \phi_\alpha(\xi) F_K(\widehat{P}_\alpha) = \sum_{\alpha \in \widehat{K}^d} \phi_\alpha(\xi) P^\ell_\alpha$$

where the $\widehat{P}_\alpha$ are the control nodes of the reference element, *i.e.* $\widehat{P}_{ijk} = (i/3, j/3, k/3)$. This also provides the definition of the Lagrange control nodes. The B_α are the Bernstein polynomials, defined by

$$B_{i,j,k}(u,v,w) = \binom{3}{i}\binom{3-i}{j} u^i v^j w^k$$

for every $(i,j,k) \in \widehat{K}^d$ and $(u,v,w) \in \widehat{K}$. When choosing the control points at the thirds, *i.e.*

$$P_{ijk} = \frac{i}{3} P_{300} + \frac{j}{3} P_{030} + \frac{k}{3} P_{003},$$

the Lagrange and Bézier control nodes coincide and the mapping F_K degenerates to become linear. This is what we'll naturally refer to as the straight P^3 element. Figure 4 illustrates a P^3 triangle with its Lagrange control nodes.

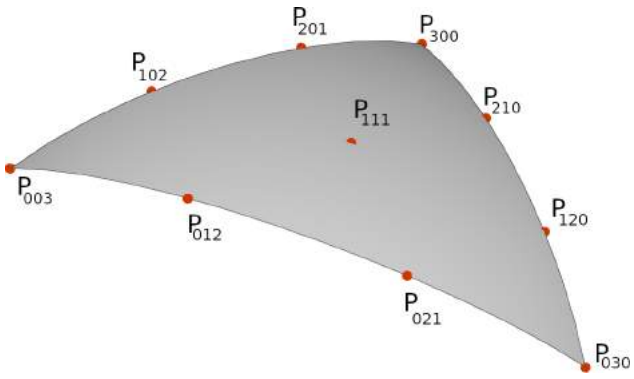

Figure 4: P^3 triangle with Lagrange nodes in red.

The P^1 mesh must now be brought to the third degree. To do this, new edge and face high-order nodes must be created for each triangle and have their positions set. There are two approaches to this. The first is to initialize the Lagrange nodes of each element at the straight position in physical space. These points are then projected onto the surface using the CAD model. The second approach evaluates the Lagrange nodes on CAD faces directly. This can be done easily since, for every vertex P of the P^1 mesh, its coordinates in the parametric domain of the host face are known from the P^1 meshing step. Taking an edge of extremities P_{30}, P_{03} on a face with mapping σ and with ξ_{30} and ξ_{03} known such that $\sigma(\xi_i) = P_i$, the Lagrange nodes for the direct approach are given by

$$P_{21}^{dir} = \sigma\left(\frac{2}{3}\xi_{30} + \frac{1}{3}\xi_{03}\right) \quad \text{and} \quad P_{12}^{dir} = \sigma\left(\frac{1}{3}\xi_{30} + \frac{2}{3}\xi_{03}\right),$$

whereas, denoting by Π_σ the surface projection operator, the Lagrange nodes for the inverse approach are

$$P_{21}^{proj} = \Pi_\sigma\left(\frac{2}{3}P_{30} + \frac{1}{3}P_{03}\right), \; P_{12}^{proj} = \Pi_\sigma\left(\frac{1}{3}P_{30} + \frac{2}{3}P_{03}\right).$$

The direct approach is faster than the inverse approach due to CAD projections. However, we will see that the indirect approach is more robust, since the initial position of the Lagrange nodes is not too far from the straight element, which is valid in the sense that it does not self-intersect and remains well-conditioned for optimization (projection). In either case, boundary edges are first brought to the high order and stored in a hash table. Triangles are then looped over and new control points created or recovered from the hash table. At this stage, the mesh is fully P^3 with all control points on the geometry. The last optional step is to apply a Lagrange-to-Bézier transformation since the Bézier representation is a more convenient and generalizable one. Using the definition of the Lagrange control nodes, we have the following relations:

$$\sum B_{ijk}^3(i_0/3, j_0/3, k_0/3)P_{ijw} = P_{i_0 j_0 k_0}^\ell$$

for every $(i_0, j_0, k_0) \in \widehat{K}^d$. These equations degenerate for any triplet where one of the indices is 3 (principal vertices of the triangle), leaving 7 non-trivial equations which correspond to two per edge and one for the face control node. Equations relating to edge control nodes are treated in pairs, yielding 6 of the Bézier control nodes. The face Bézier node is finally computed using these values. Taking as example the edge $\{w = 0\}$,

$$12P_{210} + 6P_{120} = 27P_{210}^\ell - 8P_{300} - P_{030}$$
$$6P_{210} + 12P_{120} = 27P_{120}^\ell - 8P_{030} - P_{300}.$$

This leads to

$$-6P_{210} = 9P_{120}^\ell - 18P_{210}^\ell - 2P_{030} + 5P_{300}$$
$$-6P_{120} = 9P_{210}^\ell - 18P_{120}^\ell - 2P_{300} + 5P_{030}.$$

As for the face node, the case $i = j = k = 1$,

$$6P_{111} = 27P_{111}^\ell - P_{300} - 3P_{210} - 3P_{120} - P_{030} - 3P_{021} - 3P_{012} - P_{003} - 3P_{102} - 3P_{201},$$

which we compute using the previous.

Let us now compare the two approaches to constructing the Lagrange nodes: the direct evaluation and the projection. Fig. 5 illustrates the typical case of a half sphere mapped to from a rectangle in parametric space. It has two poles where quad edges have degenerated into a single point. This leads to points close to each other on the surface to have very distant parametric coordinates. In particular, edges of triangles close to these poles can be seen to be very curved (top figure) when the Lagrange control nodes were evaluated directly. Fortunately, this does not affect the Lagrange nodes which were projected from the straight positions as much. Indeed, this is less of a problem of ill-definition than one of strong variations: this is a region where points close in parametric space are sent to positions far from each other in physical space. This is not, however, truly a problem for projection, especially for edge nodes that are well in the interior of the face patch. However, in both cases but especially in the second, it is now possible to compute a surface normal and curvatures where the CAD previously did not allow.

There is room for improvement in the choice of the control points. For now, these are placed at the straight positions ($P_{ijk}^\ell = \frac{i}{3}P_{300} + \frac{j}{3}P_{030} + \frac{k}{3}P_{003}$) in parametric space. It could be beneficial to optimize this initial placement with regards to geometric deviation, $\mathcal{G}^1$ continuity or edge length in metric space with the P^3 surface approximation metric in parametric space. Note that this P^3 mesh is not guaranteed to be $\mathcal{G}^1$ continuous contrarily to those constructed using the tangent plane method [22]. However, as the benchmarks in the following subsection show, this is of little consequence in practice.

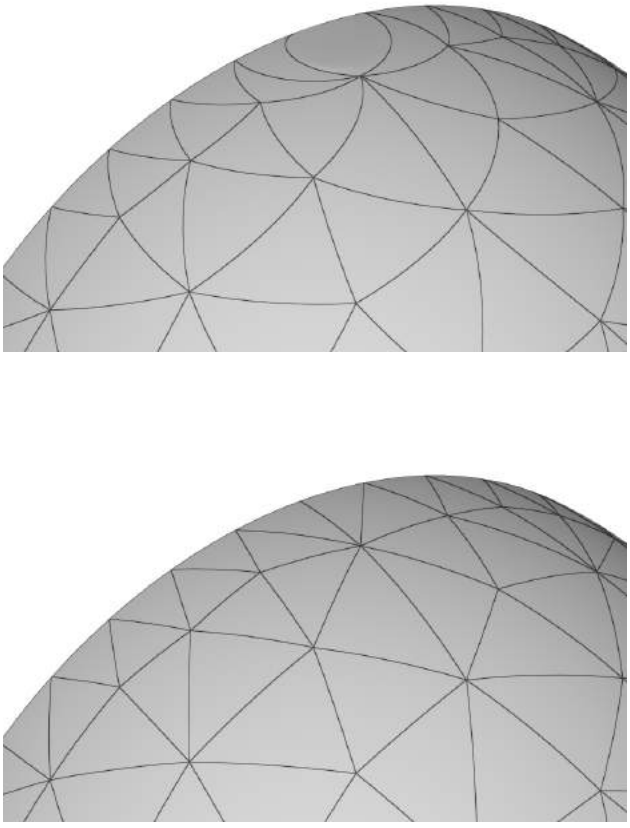

Figure 5: Comparison of P^3 meshes with Lagrange control nodes created using the direct evaluation approach (top) and the projection of the straight element approach (bottom) in the vicinity of the pole of a sphere.

2.3 Geometric primitives on the P^3 CAD surrogate

This mesh has been constructed with the objective of providing fast and robust projection on the surface as well as evaluation of derivatives. Indeed, when adapting the computational surface mesh, new vertices are created which must lie on the geometry. Likewise, surface optimization may call for normals or curvatures which are linked to, respectively, first and second derivatives of the surface parameterization. These two steps commonly fail on CAD systems. One common cause of failure for projection is the high tolerances present between patches: too close to patch intersections, the geometry is ill-defined. Another problem comes from the fact that it is carried out by gradient-based optimization — typically, Newton — in parameters space at the patch level. This means that, to project point P on the face with mapping σ over parametric domain D, it is the couple $(u, v) = \min_D\{||\sigma - P||\}$ that is sought and the (up to second) derivatives of σ are used. Therefore, projections may fail because of degeneracies of the mapping (such as σ mapping a boundary edge of D to a single point as for a cone) which lead to undefined behaviour of the derivatives.

Given a P^3 element K and barycentric coordinates of some point on the triangle, computing derivatives is trivial due to the polynomial nature K. Projection becomes slightly more algorithmic, and proceeds in two steps. The lower level step consists in computing, for a given triangle K, the barycentric coordinates of the projected point onto K. The higher level strategy uses this step to move around elements according to the sign of these barycentric coordinates. Finding the barycentric coordinates of a point on a P^3 element is relatively costly, since it requires several steps of a gradient-based optimization algorithm. P^1 triangles, on the contrary, give exact barycentric coordinates in a single step using ratios of triangle areas. Denoting by $bary_{deg}(P, K)$ a function returning the barycentric coordinates of point P in triangle K seen as a degree deg triangle, the projection algorithm can be summarized as follows:

Input: point P, initial guess K

While not found:

For $deg = (1, 3)$:

While not found:

$\xi \leftarrow bary_{deg}(P, K)$

If $\xi_1 > 0, \xi_2 > 0, \xi_3 > 0$:

break

Else:

$K \leftarrow i$-th neighbour of K s.t. $\xi_i < 0$

The first iteration of the outer loop closes in on the correct element, using only cheaper projections on linear elements, so that very few (often one) steps are left to identify the correct P^3 triangle in the second iteration. Some details were omitted for simplicity, such as usual search logic (marking elements so as to avoid repetition, stacking or sorting candidates when two barycentric coordinates are negative) and the fact that a guess for the normal is supplied to avoid finding a local minimum on the wrong side of a thinly folded surface and to distinguish the cases where the point lies outside of an open surface.

3. NUMERICAL EXAMPLES

3.1 Benchmarks

We now turn to simple benchmarks of the P^3 projection operator and of the P^3 mesh construction. The geometry used in this section is the High-Lift Common Research wing Model (HL-CRM) with flaps used in the 4-th AIAA CFD High Lift Prediction Workshop. Figure 6 illustrates this geometry. The model contains 262 CAD faces and 700 CAD edges. This geometry is relatively complex with a number of degenerate patches: any triangular patches were originally mapped from a rectangular domain (green, orange domains in bottom figure).

3.1.1 Projection speed and accuracy

For the first test, we loop over elements and generate N random points on each P^3 triangle. We then call the projection algorithm with no knowledge of the solution and compare its return value to the expected point. Euclidean norm of the difference is used to compute ℓ^2 and ℓ^∞ norms of the projection error. This is compared to a similar test on the CAD system, wherein points are generated on the geometry and then projected using a reasonable first guess (closest point in mesh). CAD projections are carried out using EGADSlite [23] built-in tools. Two meshes of 48 and 92 thousand triangles obtained by tessellating with surface error metrics for a geometric tolerance of 0.1 and 0.05 respectively are used. The only influence mesh coarseness should have on the results is in making the P^3 triangles slightly flatter. Indeed, we are not measuring deviation from the P^3 mesh to the geometry but rather the ability of the projection algorithm to recover a point that is exactly on the surface. Table 1 offers a summary of the results on these two meshes created from the same high-lift geometry. Both for the CAD and P^3 tests and on either mesh, $1M$ test points were generated to project. Projection errors are normalized for wing length. The first thing that stands out is that the P^3 projection is in the order of 3 times faster than the projection on the CAD using EGADSlite. Projection quality is better using the CAD on average, with ℓ^2 errors for the P^3 projection in the order of $1e-8$ and in the order of $1e-11$ for the CAD projection. EGADSlite CAD projections therefore tend to yield results in the order of 3 orders of magnitude times better than the P^3 projection in its current state. However, looking at ℓ^∞ errors, it appears that CAD projection is capable of more catastrophic failures, with a highest error of roughly 10% of the model's size on at least one point. This is not an isolated result, as similar behaviour has been observed on other cases. We believe the high average quality of CAD projections is due to the fact that NURBS are mostly regular, except on patch boundaries when singularities exist. A number of these singularities exist on this model, such as on the bounding box (a half sphere with two degenerate edges at the poles) and on some triangular patches seen Fig. 6 with one degenerate edge. The P^3 mesh, on the other hand, is regular and unbothered by these singularities, except potentially on construction. This explains that ℓ^∞ errors on P^3 projection remain more controlled. As for its worse ℓ^2 accuracy, the algorithm implemented to carry out P^3 projection is a very rudimentary Newton descent with a fixed iteration count, no line search nor preconditioning. Using a more sophisticated algorithm would yield better results, as the optimization problem at hand is relatively simple.

# Triangles	48k	92k
P^3 err. ℓ^2	$4.2e{-8}$	$2.1e{-8}$
P^3 err. ℓ^∞	$1.15e-2$	$8.3e{-3}$
P^3 CPU	1.15M p/sec	1.08M p/sec
CAD err. ℓ^2	$6.7e{-11}$	$4.1e{-11}$
CAD err. ℓ^∞	$1.1e{-1}$	$1.1e-1$
CAD CPU	366k p/sec	390k p/sec

Table 1: Results for the first benchmark: projection of random points using the CAD system (EGADSlite) and the P^3 CAD surrogate. CPU times given in projections per second (p/sec). 10^6 points were generated in both cases.

3.1.2 Surface approximation error

Now that the P^3 projection operator has been verified, we can move onto the second test involving geometric approximation. Here, we seek to evaluate the gap between the actual surface (CAD model) and the surface meshes (P^1 and P^3). There are two approaches to this. The first involves generating points on the P^3 and P^1 geometries and then projecting them onto the CAD model. The second does the opposite: points are generated on the CAD model and then projected onto the P^1 and P^3 surfaces. We chose the second approach because it is stabler, in that the CAD model is only used for evaluation and the mesh projection has been validated by that point.

To follow this approach, we proceed on a per-face basis and require a tessellation of the trimmed parametric domain of each CAD face. This is accomplished easily at this stage. Indeed, the surface mesher keeps records of which CAD faces and edges created which mesh vertices and at which parameter values. In other words, for any vertex P_i of the surface mesh, it is trivial to retrieve the σ and ξ_i for which $\sigma(\xi_i) = P_i$. Now, given the set of triangles and vertices (P_i) that discretize a given CAD face, the desired tessellation of the trimmed parameters domain is given by the mesh with the same connectivity and vertices ξ_i. We denote

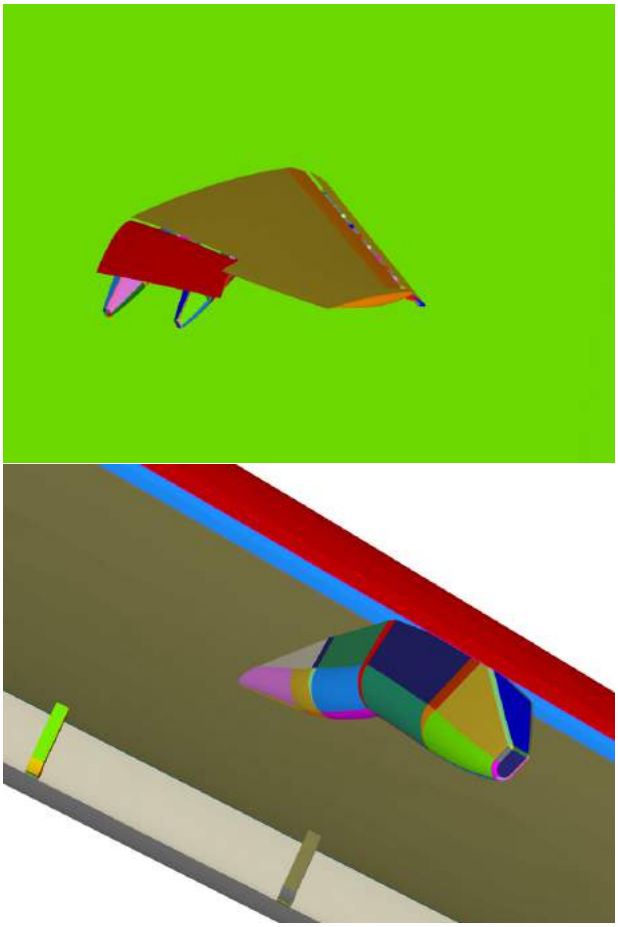

Figure 6: The high-lift wing model with flaps.

$(K_i)_{1\leq i\leq N}$ the N triangles of this tessellation and Π^k the projection operator on the degree k mesh. We seek to evaluate the quantities

$$E_p^k = ||\sigma - \Pi^k\sigma||_{L^p(\bigcup_{1=1}^N K_i)} \tag{1}$$

for p either 2 or ∞. In the case of $p = 2$,

$$\begin{aligned} E_2^k &= \sum_{i=1}^N \int_{K_i} ||\sigma(\xi) - \Pi^k(\sigma(\xi))||d\xi \\ &= \sum_{i=1}^N |J_{K_i}| \int_{\widehat{K}} ||\sigma(F_K(\widehat{\xi})) - \Pi^k(\sigma(F_K(\widehat{\xi})))||d\widehat{\xi} \end{aligned} \tag{2}$$

where $F_K : \widehat{K} \mapsto K$ is the reference-to-physical mapping of the triangle K and J_K the determinant of its Jacobian. Note that, since we are dealing with elements defined in parametric space, this quantity is well-defined since elements are in $\mathbb{R}^2$. Furthermore, even in the case of a P^3 triangle, the Jacobian is constant because we have constructed the P^3 triangles to be straight in parametric space (their mapping degenerates into the linear one) and it is only in physical space that they are curved. These individual integrals are then evaluated by simple uniform quadrature. The reason for this choice is that we seek to evaluate this integral with a great degree of precision and such a scheme is very simple to converge, though at a slower pace. 45 quadrature points were used. As for the L^∞ error, we simply take the maximum absolute value at the quadrature points, needing no further computations. Greater accuracy on the L^∞ error is another advantage of a quadrature scheme using more points. We also define the errors on the normal directions

$$\partial E_p^k = ||\mathbf{n}_\sigma(u,v) - \mathbf{n}^k(\Pi^k\sigma)||_{2,L^p(\bigcup_{1=1}^N K_i)} \tag{3}$$

where $\mathbf{n}_\sigma(u,v) = (\partial_u\sigma \wedge \partial_v\sigma)/\|\partial_u\sigma \wedge \partial_v\sigma\|$ is the normalized normal vector computed using the CAD patch and $\mathbf{n}^k(X)$ is the normalized normal at point X on the P^k mesh. Furthermore, errors are the normalized $\widetilde{E}_p^k = E_p^k C^k$ where C^k is the mesh complexity for a given degree given by

$$\begin{aligned} C^1 &= 3N_T + 3\rho N_P^1 \\ C^3 &= 10N_T + 3\rho(N_T + N_P^1 + 2N_E) \end{aligned} \tag{4}$$

where N_T denotes the number of triangles in the mesh, N_P^1 the number of vertices in the P^1 mesh, N_E the number of edges. ρ is the real to integer storage cost ratio, with $\rho = 2$ in our case given that 32 bit integers and 64 bit reals (double precision) were used. The same goes for the errors on normals $\partial\widetilde{E}_p^k$. Finally, we normalize by the total area of the surface, estimated from the P^1 mesh.

Results are summarized in Tab. 2. Complexities for the P^1 and P^3 meshes show that, in both cases, the P^3 mesh was upwards of 6 times heavier in memory than the P^1 mesh for the same number of triangles. However, looking at the ratios of errors, the P^1 mesh is clearly much worse than than the P^3 mesh for the same number of degrees of freedom. On the coarser mesh, the P^1 mesh was 72 times worse in L^2 norm and 7.5 times worse in L^∞ norm. These figures jump to a 1900, resp. 83, times worse P^1 mesh at the same number of degrees of freedom with the finer mesh. This alone heavily favours the P^3 CAD surrogate for geometric projection. But looking at the errors on normals, it comes with no surprise that the P^3 mesh is thousands of times more accurate than the P^1 mesh at a given number of degrees of freedom. Even more so on the coarser mesh, where the piece-wise constant normals given by the P^1 mesh are no match for the smooth normals of the P^3 mesh (factor $\times 20000$ more accurate on average and in the worst case).

$N_T/C^1/C^3$	48k / 290k / 1.8M	92k / 550k / 3.4M
$\widetilde{E}_2^1$	5.7e2	41
$\widetilde{E}_\infty^1$	5.7e-1	4.5e-2
$\widetilde{E}_2^3$	7.9	2.2e-2
$\widetilde{E}_\infty^3$	7.7e-2	5.4e-4
$\widetilde{E}_2^1/\widetilde{E}_2^3$	72	1.9e3
$\widetilde{E}_\infty^1/\widetilde{E}_\infty^3$	7.5	83
$\partial\widetilde{E}_2^1$	9.0	2.2
$\partial\widetilde{E}_\infty^1$	5.9e-3	3.1e-3
$\partial\widetilde{E}_2^3$	4.7e-4	4.5e-4
$\partial\widetilde{E}_\infty^3$	3.7e-7	3.2e-7
$\partial\widetilde{E}_2^1/\partial\widetilde{E}_2^3$	1.9e4	4.9e3
$\partial\widetilde{E}_\infty^1/\partial\widetilde{E}_\infty^3$	1.6e4	9.8e3

Table 2: Normalized errors for the P^1 and P^3 meshes. Errors are given for two coarseness levels: 48000 and 92000 triangles.

3.1.3 P^2 volume meshes

Finally, we present a minor (with regards to adaptation as a whole) use of these P^3 CAD surrogate meshes, namely as a geometric support to curve P^2 meshes. Acknowledgement of the importance of curved volume meshes dates back to the 70s with the proof that optimal convergence of high-order methods is only possible with a curved boundary in the case of elliptic problems [24, 25] and later for hyperbolic problems, where physical features are lost on P^1 boundaries [26]. Unlike CAD surrogate meshes, these curved surface meshes are constrained by the validity of inciding tetrahedra. Indeed, they are comprised of triangles that are faces of volume elements whose Jacobian determinants must remain positive. A volume mesh curving technique based on minimizing edge lengths in the metric field [27] could be extended to surface meshes by parameterizing Lagrange node position of P^2 edges as barycentric coordinates on P^3 CAD surrogate mesh elements. This takes the constraint that the Lagrange node must lie on the surface into account at a lower level than by letting it be any point in space that is later projected on the surface. Not only does this reduce the dimension of the edge length minimization problem from 3 to 2 variables, it also makes for more accurate derivatives of the cost function. This would be much more complicated to do on the CAD, where the parameterizations are more sophisticated, not to mention the problems already cited before.

Figure 7 illustrates results on the 3rd High-Lift CRM. This is a whole-body model with fewer features. The P^1 mesh was obtained as part of the high-lift drag prediction workshop. It is the result of adaptation using AMG/feflo.a [28] for mesh modifications and the solver Wolf [29]. The P^3 surface mesh was then created and used to project Lagrange nodes of the P^2 surface mesh. P^2 volume edges were then curved using the metric field by minimizing edge lengths. Without too much surprise, the P^2 surface mesh inherits the good properties of the P^3 CAD surrogate.

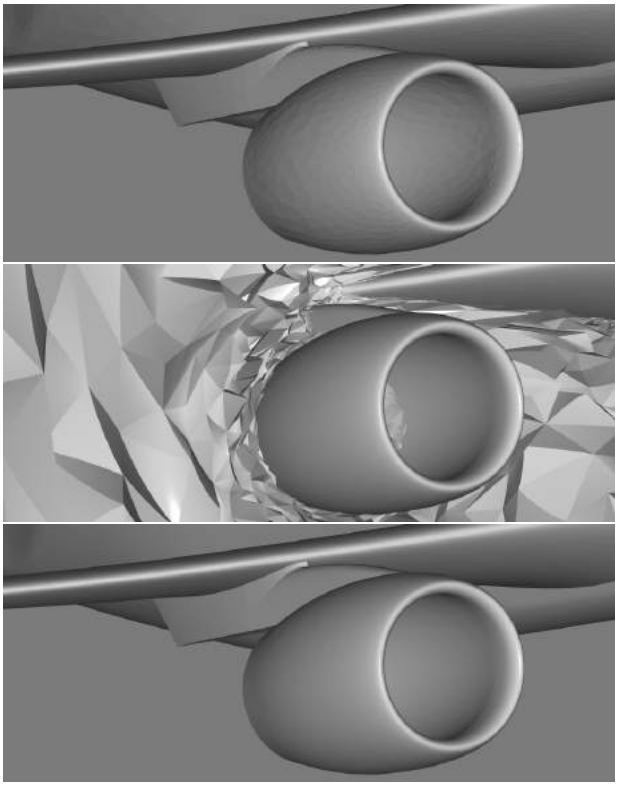

Figure 7: High-Lift CRM of the 3rd AIAA CFD High Lift Prediction Workshop meshes of degrees 1, 2 and 3 from top to bottom. The P^1 mesh (top) is the result of adaptation. The P^3 mesh (bottom) was elevated from it using the method presented here. The P^2 mesh (middle) was elevated from the P^1 mesh using projection on the P^3 mesh rather than the CAD. Volume elements (visible in the cut plane) have positive Jacobian determinants despite the clearly curved surface.

3.2 Adaptation convergence

In the previous section, we have shown that the P^3 CAD surrogate mesh is accurate as desired and that the geometric primitives are fast and robust. We now present a more pragmatic test, based on exhibiting convergence of the remeshing algorithm with an analytical boundary-layer metric while using the P^3 CAD surrogate for point projections. This metric is of the form:

$$\mathcal{M}(P) = \mathcal{R}\Lambda\mathcal{R}^T$$

with

$$\Lambda = diag(h_M^{-2}, h_M^{-2}, \min(h_m^{-2}, \max(\exp(1/\|P - \Pi P\|), h_M^{-2})))$$

$$\text{and } \mathcal{R} = \begin{pmatrix} \tau_1(\Pi P) & \tau_2(\Pi P) & \mathbf{n}(\Pi P) \end{pmatrix}$$

where ΠP denotes the projection of P onto the surface, τ_i are unit tangent vectors and $\mathbf{n}$ a unit normal at the surface. Finally, h_m and h_M are, respectively, the minimum and maximum metric sizes.This metric field is anisotropic with the smallest size along the normal direction to the surface. This size converges to the minimum admissible size as one goes closer to the surface, leading to a mesh with that prescribed size extruding from the surface. This metric field is illustrated in Fig. 8.

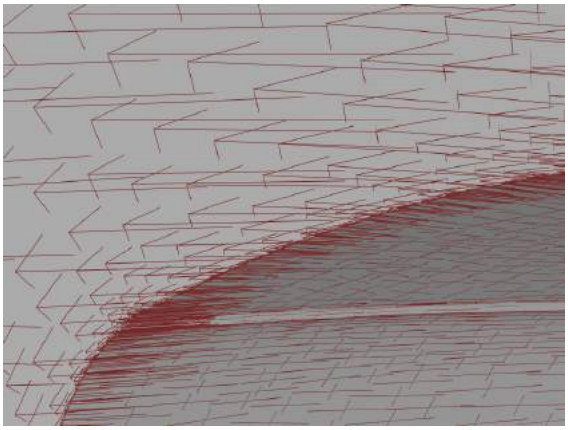

Figure 8: Analytical metric field used. Metrics are represented by their eigenvectors pondered by the square root of the inverse of their eigenvalues (characteristic sizes).

This is a challenging case in that very small edges are liable to appear in the vicinity of patch junctions, where the geometry is ill-defined by the CAD system. The converged adapted meshes must display the proper anisotropy even close to the surface. Furthermore, geometric approximation error must not increase during adaptation. Otherwise, this would mean that the surface mesh is converging to the wrong geometry. We will compare two adaptation runs: the first uses the P^3 CAD surrogate, the second the CAD model. Adaptation is carried out by the automatic metric-based remesher AMG/feflo.a [28]. The overall adaptation proceeds as follows:

While target mesh complexity not reached

Compute metric field with increased complexity target

Adapt mesh to metric field

This is necessary because, despite the fact that the chosen metric is analytical, the metric field is only known at the vertices of the input mesh and is therefore an approximation of the desired metric field. As such, even starting at the correct complexity, several iterations would be necessary. Starting from a lower complexity target affords faster convergence since individual executions of the remeshing algorithm are faster to execute. In our case, 13 iterations were carried out.

	P^3 CAD surrogate	CAD model
CPU time	9m18s	9m16s
Unit Edge %	91.4	91.5
Q_{max}	74	97
Final Geo. err.	3.9e-6	5.5e-6

Table 3: Adaptation metrics: CPU time for the entire process, overall mesh quality (unit edges), maximum surface quality, final geometric approximation error of the adapted mesh.

Fig. 9 illustrates the adapted results in both cases. CPU times and mesh quality metrics are given in Tab. 3. Unit edges are those for which their length in the metric field lies in $[1/\sqrt{2}, \sqrt{2}]$. The quality Q of a triangle K is

$$Q(K) = \frac{1}{2\sqrt{3}} \frac{\sum l_i^2}{\mathcal{A}(K)}$$

where the l_i are the lengths of the edges, $\mathcal{A}(K)$ is the area of K, and both quantities are computed in the metric field. The scaling factor is such that the minimum of Q over all possible triangles is 1. This corresponds to unit triangles in the metric, whereas as Q goes to infinity, triangles are flatter and less unit in the metric. We then define Q_{max} as the maximum quality over all triangles in the mesh. CPU times are extremely close. This is simply a consequence of the fact that surface projection is not very significant in overall adaptation CPU time. Indeed, the majority of mesh vertices lie in the volume and even for surface points the projection times represent at most one tenth of insertion time. As an order of comparison, our algorithm inserts between 25 and 40 thousand points per second depending on compilation flags, whereas point projection proceeded at a rate of 300 thousand per second even for the slowest case, direct CAD projection. Proportion of unit edges in the mesh and maximum surface element quality are within reasonable values. This in itself is not enough to conclude that P^3 projection is sufficient. Indeed, we must now compare geometric approximation errors and ensure that the adapted surface mesh is not degraded with regards to the initial mesh. Geometric error is reported in L^∞ norm as computed in (2). In both cases, the final mesh exhibits an error under the prescribed geometric error of $2e - 3$ by a large margin.

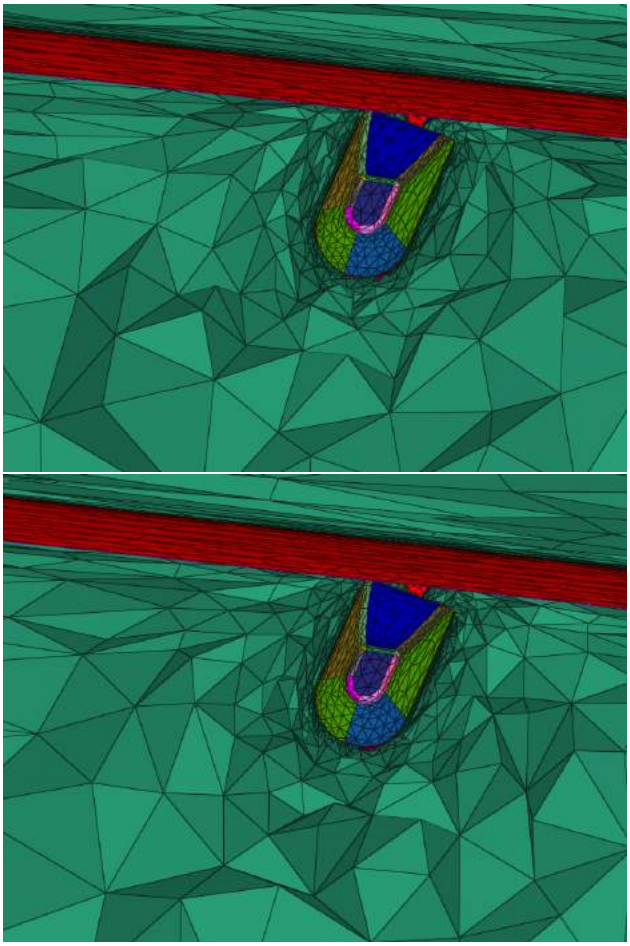

Figure 9: Detail of resulting adapted meshes using the P^3 CAD surrogate (top) and the CAD model (bottom) for the analytical boundary layer metric.

4. CONCLUSION

In this paper, we have exhibited a very simple extension to a parametric meshing algorithm that produces polynomial P^3 meshes from CAD objects, *i.e.* continuous representations riddled with singularities, degenerating features and large tolerances at intersections. These meshes are constructed in a preprocessing step and are then taken as input instead of the CAD file on each adaptation call. Tolerances at junctions between NURBS patches were managed by treating the problem on a discrete level. To do this, physical CAD edges are first discretized. This yields a set of vertices and edges of $\mathbb{R}^3$ which are then projected onto the parametric patches of neighbouring NURBS faces. This set of edges is then used to trim the NURBS patch instead of the CAD supplied parametric trimming curve, eliminating mismatches between points and edges generated by each NURBS face.

Two approaches for creating P^3 meshes from an initial P^1 surface and a CAD model were described. The first only involved CAD evaluations, by setting the parametric coordinates of the new control points at the thirds of those of edge extremities (and triangle vertices, in the case of face control nodes). The second required CAD projections, since P^3 triangles were first made straight in physical space and then projected onto parametric space. In the absence of further optimization, the second approach is more stable, as the first method easily produces bad high-order nodes close to CAD discontinuities such as the poles of a sphere. Resulting P^3 meshes are lightweight and allow for fast projection on the surface. They improve greatly on geometric deviation over P^1 meshes as well as derivative computations which can be in the order of 10^4 times more accurate for the same storage cost. Furthermore, derivatives are always defined on a P^3 Bézier mesh, which they are not necessarily on trimmed NURBS. Point projections are in the order of 3 times faster on P^3 meshes as well, by using the implicit P^1 mesh as an accelerator. Finally, the fact that the CAD model is only used in the preprocessing step reduces the frequency at which CAD errors may occur in the adaptation process. One common occurrence is a shock that spans parts of the geometry where NURBS meet. At these junctions, the CAD object has large tolerances, essentially meaning these interfaces are fuzzy under some scale. This scale tends to be much larger than that of desired element size at shocks, limiting potential anisotropy and robustness of adaptation. This cannot possibly happen on the P^3 mesh which is watertight by construction.

These virtues of the P^3 geometric representation were put to the test through a full adaptation run on the High-Lift model. Comparing between adaptation using CAD projection and using P^3 projection, we have shown that the P^3 surface mesh is sufficiently accurate to carry out mesh adaptation (including surface adaptation) on complex geometries with a strongly anisotropic metric field. Another advantage of using P^3 meshes for surface adaptation is that the CAD is not necessary. Although the techniques shown here involve a CAD, P^3 meshes can be constructed from a straight mesh by estimating normals at the vertices and enforcing $\mathcal{G}^1$ continuity [22].

The fidelity of the CAD surrogate meshes can be improved in at least two ways. The first is by optimizing the Lagrange node placement of P^3 triangles so as to minimize surface approximation error. This has not been done at all, Lagrange nodes are simply placed on the straight element at the thirds and then projected onto the geometry. There is a direct link between Bézier nodes and tangent planes of P^3 triangles (the tangent plane at P_{300} is the span of the vectors P_{210} and P_{120}, for instance) which could help devise

a procedure to fit the derivatives of the surface mapping up to the third order with little cost. An approach using the geometric approximation metric field to seek geodesics in parameter space to avoid the phenomenom illustrated in figure 5 is another possibility, albeit perhaps more costly. Since the P^3 mesh creation step is accomplished once per full adaptation, we expect such optimizations would not prove too costly in the grand scheme of things, while possibly enabling coarser P^3 meshes to approximate the surface under the tolerance. Finally, these meshes could be improved by increasing their degree. The main difficulty with higher order meshes would be in extending the aforementionned optimization strategies. The second way these CAD surrogate meshes could be improved involves replacing parts of the mesh with pieces of the CAD, such as rational Bézier triangles. These could be extracted from the CAD's NURBs patches in such a way that they would be exact representations of the original parameterization, while being an intermediary step towards a discrete mesh helping projection. This could provide a discrete-exact description of the geometry, with CAD faces being exactly represented in their interior, and approximated close to the trimming curve by a discrete P^3 mesh. Likewise, it is possible that some singularities arising from degenerating CAD edges could be fixed this way.

References

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

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

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

[4] Loseille A., Menier V. "Serial and parallel mesh modification through a unique cavity-based primitive." *Proceedings of the 22nd International Meshing Roundtable*, pp. 541–558. Springer, 2014

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

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

[7] Kompenhans M., Rubio G., Ferrer E., Valero E. "Comparisons of p-adaptation strategies based on truncation-and discretisation-errors for high order discontinuous Galerkin methods." *Computers & Fluids*, vol. 139, 36–46, 2016

[8] Ceze M., Fidkowski K.J. "Anisotropic hp-adaptation framework for functional prediction." *AIAA Journal*, vol. 51, no. 2, 492–509, 2013

[9] Dolejsi V., Ern A., Vohralík M. "hp-adaptation driven by polynomial-degree-robust a posteriori error estimates for elliptic problems." *SIAM Journal on Scientific Computing*, vol. 38, no. 5, A3220–A3246, 2016

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

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

[12] Farin G.E., Farin G. *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann, 2002

[13] Piegl L., Tiller W. *The NURBS book*. Springer Science & Business Media, 1996

[14] Marussig B., Hughes T.J. "A review of trimming in isogeometric analysis: Challenges, data exchange and simulation aspects." *Archives of Computational Methods in Engineering*, vol. 25, no. 4, 1059–1127, 2018

[15] Beall M.W., Walsh J., Shephard M.S. "Accessing CAD Geometry for Mesh Generation." *Imr*, pp. 33–42. 2003

[16] Park M.A., Haimes R., Wyman N.J., Baker P.A., Loseille A. "Boundary Representation Tolerance Impacts on Mesh Generation and Adaptation." *AIAA AVIATION 2021 FORUM*, p. 2992. 2021

[17] Sheffer A., Bercovier M., BLACKER T., Clements J. "Virtual topology operators for meshing." *International Journal of Computational Geometry & Applications*, vol. 10, no. 03, 309–331, 2000

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

[19] Feuillet R., Coulaud O., Loseille A. "Anisotropic error estimate for high-order parametric surface mesh generation."

[20] Loseille A., Alauzet F. "Optimal 3D highly anisotropic mesh adaptation based on the continuous mesh framework." *Proceedings of the 18th International Meshing Roundtable*, pp. 575–594. Springer, 2009

[21] Frey P.J., Alauzet F. "Anisotropic mesh adaptation for CFD computations." *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 48-49, 5068–5082, 2005

[22] Vlachos A., Peters J., Boyd C., Mitchell J.L. "Curved PN triangles." *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pp. 159–166. 2001

[23] Haimes R., Dannenhoffer J. "EGADSlite: A Lightweight Geometry Kernel for HPC." *2018 AIAA Aerospace Sciences Meeting*, p. 1401. 2018

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

[25] Lenoir M. "Optimal isoparametric finite elements and error estimates for domains involving curved boundaries." *SIAM Journal on Numerical Analysis*, vol. 23, no. 3, 562–580, 1986

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

[27] Loseille A., Rochery L. "Developments on the Pˆ 2 cavity operator and Bézier Jacobian correction using the simplex algorithm." *AIAA SCITECH 2022 Forum*, p. 0389. 2022

[28] Loseille A. "Chapter 10 - Unstructured Mesh Generation and Adaptation." *Handbook of Numerical Methods for Hyperbolic Problems*, vol. 18 of *Handbook of Numerical Analysis*, pp. 263 – 302. Elsevier, 2017

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