Robust untangling of curvilinear meshes

Thomas Toulorge[1], Christophe Geuzaine[2], Jean-François Remacle[1] and Jonathan Lambrechts[1,3]

[1] *Université catholique de Louvain, Institute of Mechanics, Materials and Civil Engineering (iMMC), Bâtiment Euler, Avenue Georges Lemaître 4, 1348 Louvain-la-Neuve, Belgium*

[2] *Université de Liège, Department of Electrical Engineering and Computer Science, Montefiore Institute B28, Grande Traverse 10, 4000 Liège, Belgium*

[3] *Fonds National de la Recherche Scientifique, rue d'Egmond, Bruxelles.*

Abstract

This paper presents a technique that allows to untangle high order/curvilinear meshes. The technique makes use of unconstrained optimization where element Jacobians are constrained to lie in a prescribed range through moving log-barriers. The untangling procedure starts from a possibly invalid curvilinear mesh and moves mesh vertices with the objective of producing elements that all have bounded Jacobians. Bounds on Jacobians are computed using results of papers [1, 2]. The technique is applicable to any kind of elements, both for surface, volume, hybrid or boundary layer meshes. A series of examples demonstrate both the robustness and the efficiency of the technique. The final example, involving a time explicit computation, shows that it is possible to control the stable time step of the computation for curvilinear meshes through an alternative element deformation measure.

Keywords: High order methods, finite elements, mesh generation

1. Introduction

There is a growing consensus in the computational mechanics community that state of the art solver technology requires, and will continue to require too extensive computational resources to provide the necessary resolution for a broad range of demanding applications, even at the rate that computational

Email address: thomas.toulorge@uclouvain.be (Thomas Toulorge[1], Christophe Geuzaine[2], Jean-François Remacle[1] and Jonathan Lambrechts[1,3])

power increases. The requirement for high resolution naturally leads us to consider methods which have a higher order of grid convergence than the classical (formal) 2^{nd} order provided by most industrial grade codes. This indicates that higher-order discretization methods will replace at some point the current finite volume and finite element solvers, at least for part of their applications.

The development of high-order numerical technologies for engineering analysis has been underway for many years now. For example, Discontinuous Galerkin methods (DGM) have been largely studied in the literature, initially in a theoretical context [3], and now from the application point of view [4]. In many contributions, it is shown that the accuracy of the method strongly depends on the accuracy of the geometrical discretization [5–7]. Consequently, it is necessary to address the problem of generating the high-order meshes that are needed to fully benefit from high-order methods.

Modern mesh generation procedures take as input CAD[1] models composed of *model entities*. Each model entity G_i^d has a geometry (or shape) [8, 9], that depends on the solid modeler for its underlying representation. Solid modelers usually provide a parametrization of the shapes, that is, a mapping $\boldsymbol{\xi} \in R^d \mapsto \boldsymbol{x} \in R^3$. The geometry of a model vertex G_i^0 is simply its 3D location, $\boldsymbol{x}_i = (x_i, y_i, z_i)$. The geometry of a model edge G_i^1 is its underlying curve with its parametrization $\boldsymbol{x}(t)$, $t \in [t_1, t_2]$. The geometry of a model face G_i^2 is its underlying surface with its parametrization $\boldsymbol{x}(u, v)$, $(u, v) \in \mathcal{S} \subset R^2$. The geometry associated with a model region G_i^3 is R^3. There are also four kind of mesh entities: mesh vertices M_i^0, mesh edges M_i^1, mesh faces M_i^2 and mesh regions M_i^3 that are said to be classified on model entities[2]. Each mesh entity is classified on the model entity of the smallest dimension that contains it.

The way of building a high order mesh is to first generate a straight sided mesh. Then, mesh entities that are classified on the curved boundaries of the domain are curved accordingly (see Figure 1). For mesh edges that are classified on model edges (for example $M_2^1 \sqsubset G_1^1$ on Figure 1), additional high order mesh points are added on the geometry of the model edge. Then, high order points are added on mesh edges that are classified on model faces (for example $M_1^1 \sqsubset G_1^2$ on Figure 1). Finally, high order points may be added on mesh faces that are classified on model faces. The position of the high order points can be chosen in such a way that the geometrical error,

[1] Computer Aided Design.

[2] We use the symbol $\sqsubset$ for indicating that a mesh entity is classified on a model entity.

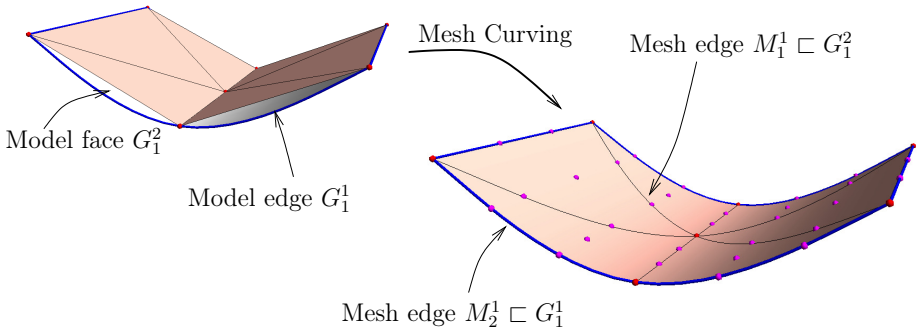


Figure 1: Straight sided mesh (left) and curvilinear (cubic) mesh (right).

i.e. the distance between the CAD model and the mesh, is minimized. In this paper however, the high order points are simply defined by orthogonal projection from the straight mesh entity onto the curved geometry defined by the CAD model.

The naive curving procedure described above does not ensure that all the elements of the final curved mesh are valid. Figure 2 gives an illustration of this important issue. Some of the curved triangles are tangled: they self-intersect after having been curved. It is important to note that this problem is not related to the accuracy of the geometrical discretization: in Figure 2, the mesh would not be valid even if the curved edge was assigned the exact geometry (blue curve). Invalid elements may be detected by exploiting

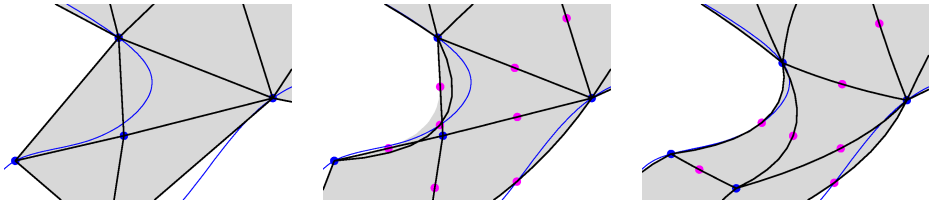


Figure 2: Straight sided mesh (left) basic curvilinear (quadratic) mesh with tangled elements (center) and untangled mesh (right).

specific properties of the Jacobian of the transformations that map a fixed reference element onto each element in the physical domain, as shown in Figure 3. In particular, a change of sign in the Jacobian over an element indicates that the corresponding map is not injective, so the numerical meth-

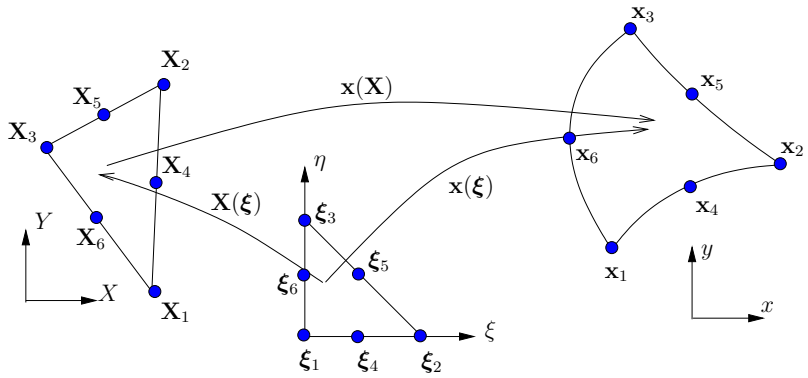


Figure 3: Reference unit triangle in local coordinates $\boldsymbol{\xi} = (\xi, \eta)$ and the mappings $\mathbf{x}(\boldsymbol{\xi})$, $\mathbf{X}(\boldsymbol{\xi})$ and $\mathbf{X}(\mathbf{x})$.

ods that rely on integration over elements cannot be used. This is the case for self-intersecting elements. In two recent papers [1, 2], a general formulation has been developed for computing robust estimates of the geometrical validity of a curvilinear element. Provable bounds on element Jacobians can be computed for high order triangles, quads, tetrahedra, hexahedra and prisms.

Figure 2 indicates that, without refining the mesh, the only way of generating a valid high order mesh is to curve not only mesh entities classified on curved model entities, but also those that are initially straight-sided inside the computational domain. It is thus necessary to somehow propagate the curvature inside the domain. Several smoothing schemes have been proposed in the literature to this effect: linear smoothing techniques such as Laplacian smoothing [10], Winslow smoothing [11] or linear elasticity with varying stiffness [10, 12]. Even though such simple techniques may often lead to interesting results, there is no guarantee whatsoever that applying such a linear smoother will result in an untangled mesh. Nonlinear smoothing techniques have also been investigated, in particular using a nonlinear elasticity analogy [13], which results in a valid curvilinear mesh. Yet, computing a nonlinear mechanics problem including large deformations on a high-order and highly stretched mesh is numerically very complex (more than solving the Navier-Stokes equations on the same grid, for instance). The computational cost may then be excessive, given that the meshing time is expected to be

a fraction of the overall CPU time. Besides linear and nonlinear smoothing techniques, other authors [14–16] make use of mesh adaptation techniques by eliminating invalid elements by a combination of local mesh refinements, edge and face swaps, and node relocations.

In this paper, we propose a robust smoothing scheme that makes it possible to build a curvilinear mesh for which the validity of every element is guaranteed. This new untangling procedure relies on an optimization algorithm rather than a mechanical analogy or local mesh refinement: it specifically targets element Jacobians and modifies node locations in such a way that Jacobian values lie in a predefined range. This approach is loosely related to the optimization schemes proposed by several authors for untangling straight-sided meshes [17–20], but with a focus on arbitrarily high-order elements.

The paper is organized as follows. In Section 2, we briefly recall the results of paper [1, 2] on Jacobian bounds. Section 3 is dedicated to the practical computation of both Jacobian bounds and their derivatives with respect to the motion of mesh vertices. Section 4 is the kernel of the paper. An objective function that specifically targets invalid Jacobians is described in Section 4.1. Constraints on Jacobian positivity are imposed through log-barriers, allowing the use of unconstrained optimization procedures. The optimization strategy is described in Section 4.2. The optimization starts with an invalid mesh and the asymptote in the log barrier is progressively moved into the valid region. Some examples of mesh untangling are presented in Section 5 where both the robustness and the efficiency of the new methodology are demonstrated. Some 3D examples are presented in Section 5.1 with timings and statistics. In Section 5.2, the example of a high order boundary layer mesh is presented. A new term is added to the objective function with the aim of controlling the maximum Jacobian as well. Finally, Section 6 presents a simple idea for controlling the impact of mesh curvature on the explicit time step of computations.

2. Validity estimates of curvilinear meshes

Let us introduce the following notations. We call n_e and n_v the number of elements and vertices of the mesh. Each element e of the mesh contains N_p vertices. Note that in this context, the word "vertex" denotes any node of the mesh, whether it is a geometrical vertex of a straight-sided element or a high-order node of a curved element.

We note $\boldsymbol{X}_i^e$ the position of the i^{th} vertex of the element e in the straight-sided configuration and $\boldsymbol{x}_i^e$ the location of the same vertex, yet in some

deformed configuration.

The shape of an element e is defined geometrically through its vertices $\boldsymbol{x}_i^e$, $i = 1 \dots N_p$ that are nodes associated to a set of Lagrange shape functions $\mathcal{L}_i^{(p)}(\boldsymbol{\xi})$, $i = 1 \dots N_p$ at order p. This allows us to map a reference element to the real one:

$$\boldsymbol{x}(\boldsymbol{\xi}) = \sum_{i=1}^{N_p} \mathcal{L}_i^{(p)}(\boldsymbol{\xi})\ \boldsymbol{x}_i^e. \tag{1}$$

Consider now the transformation $\boldsymbol{x}(\boldsymbol{X})$ that maps straight sided elements onto curvilinear elements (see Figure 3). The mapping $\boldsymbol{x}(\boldsymbol{X})$ should be bijective, i.e. it should admit an inverse. This implies that the determinant of the Jacobian $\det \boldsymbol{x}_{,\boldsymbol{X}}$ has to be non-zero, for every value of $\boldsymbol{\xi}$ in the reference element. In practice, this condition reduces to strict positivity for 3D elements, but an ambiguity remains for 2D elements, in which the Jacobian is defined with respect to a local normal. The orientation of planar 2D elements can be chosen in such a way that the direction of the constant normal ensures the positivity of the Jacobian. For curved surfaces in 3D however, the definition of the Jacobian is more complex, so that the condition of strict positivity that we retain in this paper may be overly restrictive.

It is possible to write this determinant in terms of the $\boldsymbol{\xi}$ coordinates as:

$$\det \boldsymbol{x}_{,\boldsymbol{X}} = \frac{\det \boldsymbol{x}_{,\boldsymbol{\xi}}}{\det \boldsymbol{X}_{,\boldsymbol{\xi}}} = \frac{J(\boldsymbol{\xi})}{J_0^e},$$

where J_0^e is the strictly positive and constant[3] Jacobian of the straight sided element. The function $\boldsymbol{x}(\boldsymbol{X})$ is called the distortion mapping. Its determinant $\det \boldsymbol{x}_{,\boldsymbol{X}}$, that we call the *scaled Jacobian*, should be as close to 1 as possible in order not to degrade the quality of the straight sided element e.

In [1, 2] it is shown that it is possible to reliably detect invalid elements. In other words, it is possible to find reliable bounds to $J_{\min} = \min_{\boldsymbol{\xi}} J$ and to $J_{\max} = \max_{\boldsymbol{\xi}} J$ over the whole element. The Jacobian J is a polynomial in $\boldsymbol{\xi}$. It can then be interpolated exactly as a linear combination of Bézier polynomials $\mathcal{B}_i^{(q)}$ at a certain order $q \geq p$ over the element. Provable bounds for $J_{\min}$ and $J_{\max}$ are then computed using an interesting property of the Bézier polynomials, namely that their value is contained within the range of

[3] Straight sided element Jacobians are constant for simplical elements only, i.e. triangles in 2D and tetrahedra in 3D.

their coefficients. Assuming that J is a polynomial of order q in $\boldsymbol{\xi}$, we write

$$J(\boldsymbol{\xi}) = \sum_{i=1}^{N_q} \mathcal{B}_i^{(q)}(\boldsymbol{\xi}) B_i \tag{2}$$

and bounds can be computed as

$$\min_{\boldsymbol{\xi}} J(\boldsymbol{\xi}) \geq \min_i B_i \quad \text{and} \quad \max_{\boldsymbol{\xi}} J(\boldsymbol{\xi}) \leq \max_i B_i.$$

Although they are reliable, the bounds obtained in this manner may not be accurate: $\min_i B_i$ may be much lower than $\min_{\boldsymbol{\xi}} J(\boldsymbol{\xi})$ and $\max_i B_i$ may be much greater than $\max_{\boldsymbol{\xi}} J(\boldsymbol{\xi})$. The adaptive procedure described in [1, 2] overcomes this problem, but it implies a higher computational cost, as well as additional complexity in the formulation. Therefore, we favor the direct evaluation provided by Eq. (2) in the present work, even though it yields an overly restrictive estimation of the Jacobian bounds.

The following section is dedicated to the practical computation of the coefficients B_i as well as their derivatives with respect to the vertex coordinates $\boldsymbol{x}_i^e$.

3. Computation of Bézier coefficients and their derivatives

The aim of our method is to untangle both surfacic and volume meshes. For that, we assume that a point $\boldsymbol{x}$ has always 3 coordinates $\boldsymbol{x} = \{x, y, z\}$. Local coordinates $\boldsymbol{\xi} = \{\xi, \eta, \zeta\}$ are also supposed to be three dimensional. Yet, for surface meshes, we assume that vector

$$\mathbf{n} = \left\{ \frac{\partial x}{\partial \zeta}, \frac{\partial y}{\partial \zeta}, \frac{\partial z}{\partial \zeta} \right\}$$

is the constant unit normal vector to the straight sided element. With that hypothesis, it is possible to compute the determinant of the Jacobian at every Lagrange node $\boldsymbol{\xi}_k = (\xi_k, \eta_k, \zeta_k)$ at order q:

$$\begin{aligned} J_k = J(\boldsymbol{\xi}_k) \quad = \quad & \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \zeta} + \frac{\partial z}{\partial \xi}\frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \zeta} + \frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \eta}\frac{\partial x}{\partial \zeta} - \\ & \frac{\partial z}{\partial \xi}\frac{\partial y}{\partial \eta}\frac{\partial x}{\partial \zeta} - \frac{\partial x}{\partial \xi}\frac{\partial z}{\partial \eta}\frac{\partial y}{\partial \zeta} - \frac{\partial y}{\partial \xi}\frac{\partial x}{\partial \eta}\frac{\partial z}{\partial \zeta}. \end{aligned} \tag{3}$$

Considering that

$$x = \sum_{i=1}^{N_p} x_i^e \mathcal{L}_i^{(p)}(\boldsymbol{\xi}_k),$$

it is possible to compute the sensitivity of the Jacobian at point k with respect to the x coordinate of the vertex i:

$$\begin{aligned}\frac{\partial J_k}{\partial x_i^e} &= \frac{\partial \mathcal{L}_i^{(p)}}{\partial \xi}\frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \zeta} + \frac{\partial z}{\partial \xi}\frac{\partial \mathcal{L}_i^{(p)}}{\partial \eta}\frac{\partial y}{\partial \zeta} + \frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \eta}\frac{\partial \mathcal{L}_i^{(p)}}{\partial \zeta} - \\ &\quad \frac{\partial z}{\partial \xi}\frac{\partial y}{\partial \eta}\frac{\partial \mathcal{L}_i^{(p)}}{\partial \zeta} - \frac{\partial \mathcal{L}_i^{(p)}}{\partial \xi}\frac{\partial z}{\partial \eta}\frac{\partial y}{\partial \zeta} - \frac{\partial y}{\partial \xi}\frac{\partial \mathcal{L}_i^{(p)}}{\partial \eta}\frac{\partial z}{\partial \zeta}. \end{aligned} \tag{4}$$

The same computation can be done for $\frac{\partial J_k}{\partial y_i^e}$ and $\frac{\partial J_k}{\partial z_i^e}$. In practice, the following matrix $\mathbf{J}$ of size $N_q \times (3N_p + 1)$ is computed for every element e:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial J_1}{\partial x_1^e} & \cdots & \frac{\partial J_1}{\partial x_{N_p}^e} & \frac{\partial J_1}{\partial y_1^e} & \cdots & \frac{\partial J_1}{\partial y_{N_p}^e} & \frac{\partial J_1}{\partial z_1^e} & \cdots & \frac{\partial J_1}{\partial z_{N_p}^e} & J_1 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\ \frac{\partial J_{N_q}}{\partial x_1^e} & \cdots & \frac{\partial J_{N_q}}{\partial x_{N_p}^e} & \frac{\partial J_{N_q}}{\partial y_1^e} & \cdots & \frac{\partial J_{N_q}}{\partial y_{N_p}^e} & \frac{\partial J_{N_q}}{\partial z_1^e} & \cdots & \frac{\partial J_{N_q}}{\partial z_{N_p}^e} & J_{N_q} \end{bmatrix}$$

Assuming that $T_{lk}^q = \mathcal{B}_l^{(q)}(\boldsymbol{\xi}_k)$ is the transformation matrix that enables to compute Bézier coefficients B_l using Lagrange coefficients J_k, the matrix

$$\mathbf{B} = \begin{bmatrix} \frac{\partial B_1}{\partial x_1^e} & \cdots & \frac{\partial B_1}{\partial x_{N_p}^e} & \frac{\partial B_1}{\partial y_1^e} & \cdots & \frac{\partial B_1}{\partial y_{N_p}^e} & \frac{\partial B_1}{\partial z_1^e} & \cdots & \frac{\partial B_1}{\partial z_{N_p}^e} & B_1 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\ \frac{\partial B_{N_q}}{\partial x_1^e} & \cdots & \frac{\partial B_{N_q}}{\partial x_{N_p}^e} & \frac{\partial B_{N_q}}{\partial y_1^e} & \cdots & \frac{\partial B_{N_q}}{\partial y_{N_p}^e} & \frac{\partial B_{N_q}}{\partial z_1^e} & \cdots & \frac{\partial B_{N_q}}{\partial z_{N_p}^e} & B_{N_q} \end{bmatrix}. \tag{5}$$

that contains both the Bézier coefficients B_l as well as their gradients with respect to the position of nodes of element e is calculated through a single matrix-matrix product: $B_{lj} = T_{lk}^q J_{kj}$.

It is then possible to use the B_i's and their gradients in a gradient-based optimization procedure. In what follows, an objective function that contains the coefficients B_i is constructed in order to control the quality of elements.

4. Curvilinear mesh untangling

4.1. An objective function

This section describes the objective function $f(\boldsymbol{x}_i^e)$ that is used to untangle invalid curved elements through an unconstrained optimization procedure. We design a function

$$f = \mathcal{E} + \mathcal{F}$$

that is composed of two parts $\mathcal{E}$ and $\mathcal{F}$.

Our assumption is that the method is provided with a straight-sided mesh of high quality. This mesh has potentially been defined to satisfy multiple criteria, such as a predetermined size field, or anisotropic adaptation. When curving such meshes, it is desirable to preserve as much as possible all these features, which implies keeping the nodes as close as possible to their initial positions in the straight sided mesh.

To this end, we want to introduce some kind of *energy* $\mathcal{E}$ associated with the displacement of the nodes $\boldsymbol{x}_i^e - \boldsymbol{X}_i^e$, i.e. a positive quadratic form that is a measure of the distance between the straight sided nodes $\boldsymbol{X}_i^e$ and their position $\boldsymbol{x}_i^e$ in the curved mesh:

$$\mathcal{E}(\boldsymbol{x}_i, \mathbf{K}) = \frac{1}{2} \sum_e \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} (\boldsymbol{x}_i^e - \boldsymbol{X}_i^e) \mathbf{K}_{ij}^e (\boldsymbol{x}_j^e - \boldsymbol{X}_j^e) \geq 0 \tag{6}$$

where $\mathbf{K}$ is a symmetric positive matrix of size $3n_v \times 3n_v$ and $\mathbf{K}_{ij}^e$ is of size 3×3. In this paper, we choose $\mathbf{K}$ as the diagonal matrix with entries w_i^e / L^2, where w_i^e are user-defined weights used to set the balance between the two parts $\mathcal{E}$ and $\mathcal{F}$ of the objective function f, and L is a length scale representative of the problem. We choose L as the maximum distance, among all vertices of the initial tangled mesh, between a node and its counterpart in the straight-sided mesh. As for the non-dimensional weights w_i^e, we usually set $w_i^e = 10^5$ if the node i of element e lies on the boundary, and $w_i^e = 100$ otherwise. However, we observe in practice that the value of w_i^e has little influence on the convergence of the optimization method. The presence of $\mathcal{E}$ prevents the problem from being under-determined, and it orients the optimization procedure towards a solution that tends to preserve the straight-sided mesh, but the term $\mathcal{F}$ dominates for the most problematic (tangled) elements that drive the mesh deformation.

The second part $\mathcal{F}$ of the functional deals with Jacobian positivity. We use a log barrier [21] in order to avoid Jacobians that are too small, and a quadratic function to penalize Jacobians that are too high:

$$\mathcal{F}(\boldsymbol{x}_i, \epsilon) = \sum_{e=1}^{n_e} \sum_{l=1}^{N_q} F_l^e(\boldsymbol{x}_i^e, \epsilon)$$

with

$$F_l^e(\boldsymbol{x}_i^e, \epsilon) = \left[\log \left(\frac{B_l^e(\boldsymbol{x}_i^e) - \epsilon J_0^e}{J_0^e - \epsilon J_0^e} \right) \right]^2 + \left(\frac{B_l^e(\boldsymbol{x}_i^e)}{J_0^e} - 1 \right)^2, \tag{7}$$

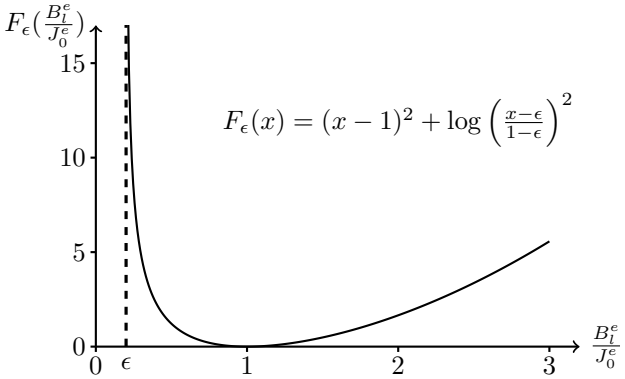


Figure 4: Plot of the barrier function $F(J_l^e)$ for $\epsilon = 0.2$.

that is defined in such a way that $\mathcal{F}$ blows up when $B_l^e = \epsilon J_0^e$, but still vanishes whenever $B_l^e = J_0^e$. Barrier methods are among the most powerful class of algorithms available for attacking general nonlinear optimization problems. These techniques converge to at least a local minimum in most cases, regardless of the convexity characteristics of the objective function and constraints [22]. Figure 4 shows a plot of the barrier function in Equation (7) for $\epsilon = 0.2$.

The objective function f being smooth, it is possible to compute its gradient ∇f with respect to the positions of the element vertices $\boldsymbol{x}_i^e$. This gradient is used in a gradient-based optimization procedure.

Mesh vertices can be classified on mesh entities of various dimensions.

Some of the mesh vertices $M_i^0 \sqsubset G_j^1$ are classified on model edges. Such a vertex can only be moved along G_j^1, i.e. its position only depends on one curve parameter t. We have therefore

$$\frac{df}{dt} = \frac{\partial f}{\partial \boldsymbol{x}_i^e} \cdot \frac{d\boldsymbol{x}_i^e}{dt}$$

with $\frac{d\boldsymbol{x}_i^e}{dt}$ the tangent vector to the curve at point t, that is obtained from the CAD model.

Other vertices, that are classified on model faces $M_i^0 \sqsubset G_j^2$, can only be moved along the surface. Two parameters u and v are associated to those vertices. We have therefore

$$\frac{\partial f}{\partial u} = \frac{\partial f}{\partial \boldsymbol{x}_i^e} \cdot \frac{\partial \boldsymbol{x}_i^e}{\partial u} \quad \text{and} \quad \frac{\partial f}{\partial v} = \frac{\partial f}{\partial \boldsymbol{x}_i^e} \cdot \frac{\partial \boldsymbol{x}_i^e}{\partial v}$$

with $\frac{\partial \boldsymbol{x}_i^e}{\partial u}$ and $\frac{\partial \boldsymbol{x}_i^e}{\partial v}$ the two tangent vectors to the surface at point (u, v). Those can be computed using the CAD model.

Vertices that are classified on model regions have a complete freedom to move in every direction of the 3D space, in which case the physical coordinates $\{x, y, z\}$ are used. Finally, mesh vertices that are classified on model vertices have no freedom to move, and are excluded from the optimization problem.

4.2. Optimization Strategy

The problem of untangling curvilinear meshes is defined as

$$\min_{\boldsymbol{x}_i} f(\boldsymbol{x}_i, \mathbf{K}, \epsilon), \quad i = 1, \ldots, n_v.$$

There is a variety of methods that can be used to solve unconstrained minimization problems. We have tested several alternatives: interior point methods implemented in the software package IPOPT [23], as well as L-BFGS [24] and conjugate gradients [25] algorithms provided by ALGLIB [25]. In the end, the use of conjugate gradients seemed to be the best choice in terms of computational efficiency.

The most important part of the optimization strategy is indeed to define the right sequence of optimization problems. Some preliminary remarks can be made at this point:

- The optimization should not be applied to the whole mesh but locally. Blobs of elements that surround an invalid element are constructed. Mesh vertices that lie on the boundary of the blob are fixed.

- It is necessary to apply preconditioning to the optimization problem, because the scale of parametric or physical coordinates of different mesh vertices can differ by orders of magnitude, depending on the model entity on which they are classified. We found that a simple diagonal preconditioner, based on the norm of the tangent vectors $\frac{d\boldsymbol{x}_i^e}{dt}$ for vertices classified on model edges (respectively $\frac{\partial \boldsymbol{x}_i^e}{\partial u}$ and $\frac{\partial \boldsymbol{x}_i^e}{\partial v}$ for vertices classified on model faces), and unity for vertices classified on model regions, yields fast and robust convergence.

- In order for the variables to remain in the domain of definition of the barrier function $\mathcal{F}$, the value of the barrier ϵ must be lower than the worst scaled Jacobian of all elements in one given blob. In particular, ϵ has to be negative for an initially tangled mesh. Therefore, we compute a sequence of optimization problems with "moving barriers": ϵ is

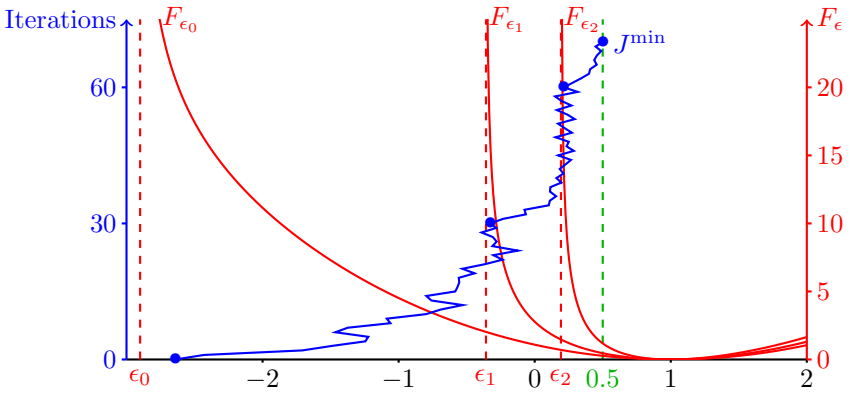


Figure 5: Optimization process: three successive series of (maximum) 30 conjugate gradient iterations, with their respective log barriers.

increased between each optimization problem, until the desired barrier value $\bar{\epsilon}$ is reached. This procedure is illustrated in Figure 5.

The optimization strategy is described in Algorithm 1.

Algorithm 1: Optimization strategy

```
Compute non overlapping element blobs B_k, k = 1 ... N_B ;
for k = 1 to N_B do
    repeat
        compute κ = min_e min_l B_l^e / J_0^e ,   e ∈ B_i,  l ∈ [1, N_q];
        if κ < ε̄ then
            set ε = κ − 0.1 |κ|;
            solve min_xi f(x_i, K, ε) for all elements of blob B_k;
            recompute κ = min_e min_l B_l^e / J_0^e ,   e ∈ B_i,  l ∈ [1, N_q];
    until κ ≥ ε̄;
```

As an example, consider a coarse 3D tetrahedral mesh of a sphere, as presented in Figure 6. The surface of the sphere is described in the CAD system as one single patch that covers the whole range of spherical coordinates. In order to challenge our optimization strategy, high order nodes classified on the surface are added along lines in the parameter plane. The resulting mesh that is presented in the middle image of Figure 6 is clearly

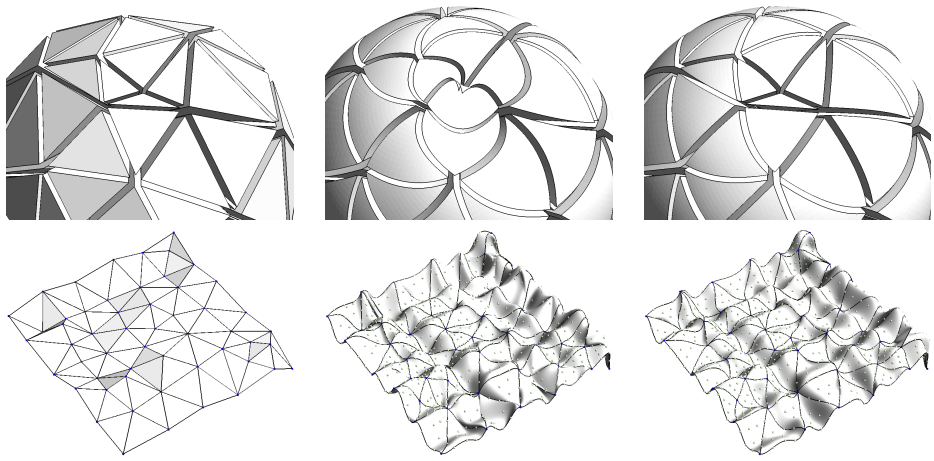


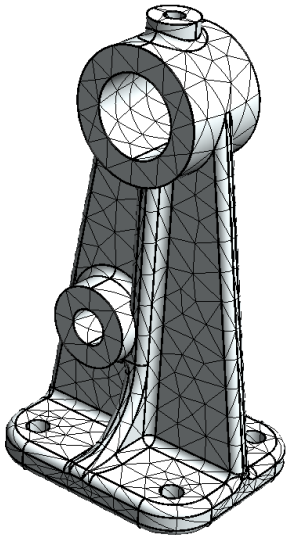
Figure 6: Examples of mesh untangling with mesh vertices motions on manifolds. Upper figures describe the untangling of a 3D coarse mesh of a sphere. The straight sided mesh (left) is made quadratic (center) and is subsequently untangled (right).

wrong. Our untangling strategy is then successfully applied to the invalid mesh: the final valid mesh that is presented in Figure 6 has all elements with scaled Jacobians in the range $[0.9, 1.1]$. Less than one second is required for converging. The other example in Figure 6 is a quartic surface mesh that is untangled using the same procedure.

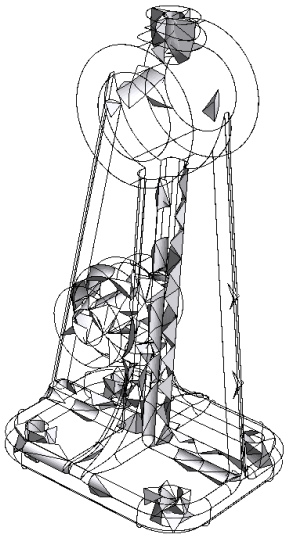
5. Examples of mesh untangling

5.1. 3D Mechanical parts

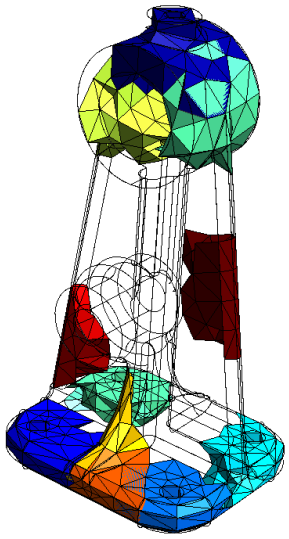
Figure 7 shows images of the mesh of a mechanical part before and after the untangling process. The mesh is composed of 5,605 quadratic tetrahedra of which 215 have a $J^e_{\min}/J^e_0 < 0.3$. The untangling process was performed considering 13 blobs of elements. The process for constructing one blob can be described as follows. One invalid element is initially inserted into the blob. N layers of elements around the invalid element are inserted in the blob. Vertices that lie on the boundary of the blob are fixed in order not to take any risk of invalidating elements outside the blob in the optimization process. In this process, blobs may of course overlap. A value of $N = 2$ is usually sufficient for ensuring the convergence of the optimization process. Note that this way of building the blobs is not adequate for untangling highly stretched meshes. This issue will be addressed in the next subsection.



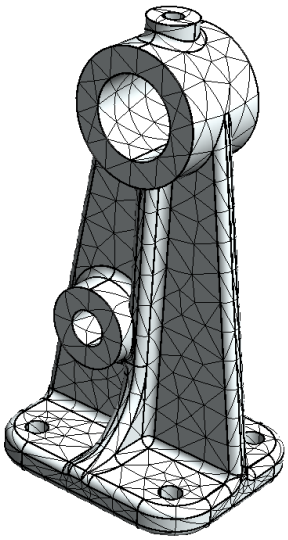
Initial quadratic mesh



Invalid tetrahedra



Some blobs of elements



Final untangled mesh

Figure 7: Mesh of a mechanical part that is composed of 5,605 quadratic tetrahedra. Figures show a surface view of the initial quadratic mesh, a volume view of the 215 invalid tetrahedra, a view of 10 among 13 blobs of elements used in the optimization process and the final untangled mesh.

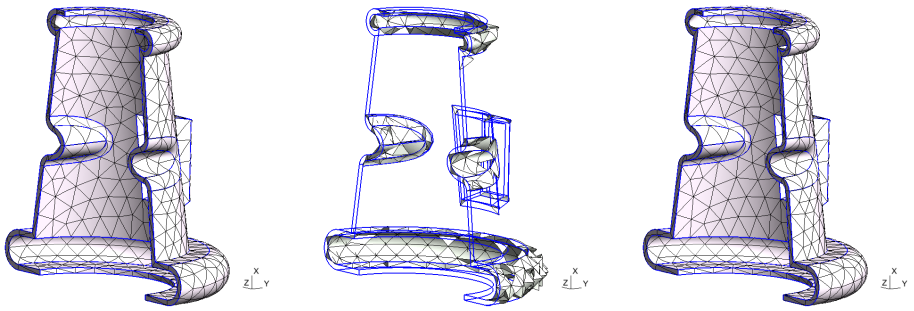


Figure 8: Mesh of a mechanical part that is composed of 2,988 quadratic tetrahedra. Figures show a surface view of the initial quadratic mesh, a volume view of the 131 invalid tetrahedra and the final untangled mesh.

In the example of Figure 7, we take the value $N = 2$ and the total CPU time for untangling that mesh is 18 seconds on a standard laptop. A global optimization procedure involving one single blob of 5,605 elements takes 60 seconds to converge.

Another example is presented on Figure 8. The same parameters as in the first example is used. The total time needed to untangle the 3 blobs of elements is now 23 seconds.

5.2. High-lift airfoil

We consider the high-lift airfoil shown in Figure 12. Two straight-sided meshes $\mathcal{M}_1$ and $\mathcal{M}_2$, featuring boundary layer-type stretching around every surface of the airfoil, are generated. $\mathcal{M}_1$ is made out of 9,814 triangles, while $\mathcal{M}_2$ contains 3,178 triangles and 3,318 quads. Both meshes are converted to order 2 to 5 and optimized.

The objective function described in Section 4.1 is designed to untangle invalid elements that are created by the conversion of straight-sided meshes to higher order, which happens mainly near convex parts of the geometry. However, we note that the conversion can also generate elements of lower quality close to concave parts, as seen in Figure 9. These elements are not strictly invalid, because their Jacobian is clearly positive everywhere, but they do not maintain the boundary layer-type size progression imposed in the straight-sided mesh. In order to remedy this inconvenience, we supplement the procedure described in Section 4 with a second optimization pass that controls the maximum Jacobian in the mesh. In the first pass, the objective function f introduced in Section 4.1 is used in the "moving barrier" procedure

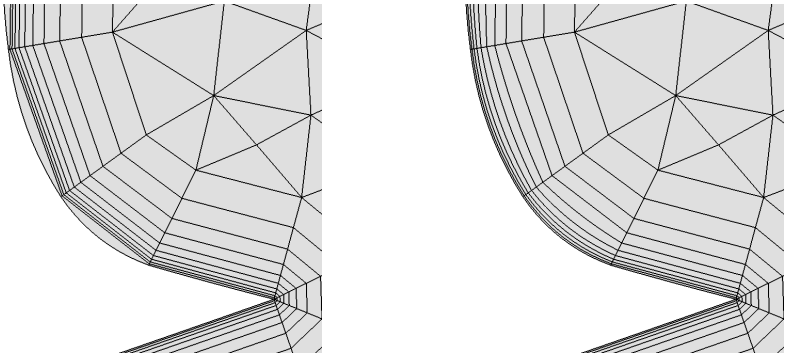


Figure 9: Detail of the mesh on the suction side of the slat in the high-lift airfoil case: primary second-order mesh (left), optimized mesh (right).

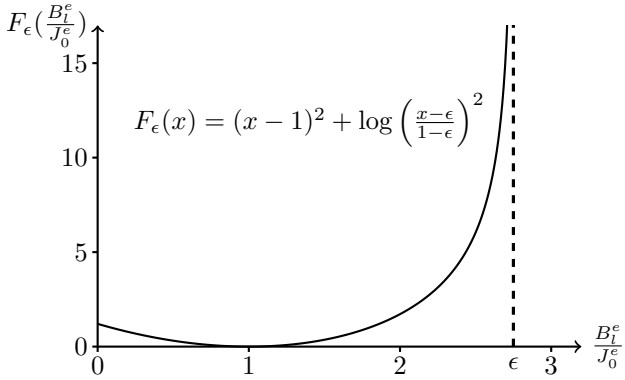


Figure 10: Plot of the barrier function $F(J_l^e)$ for $\epsilon = 2.75$.

mentioned in Section 4.2 to reach the desired minimum Jacobian $\bar{\epsilon}_{min}$. In the second pass, we switch to a different objective function f_1:

$$f_1(\boldsymbol{x}_i, \boldsymbol{K}, \bar{\epsilon}_{min}, \bar{\epsilon}_{max}) = \mathcal{E}(\boldsymbol{x}_i, \boldsymbol{K}) + \mathcal{F}(\boldsymbol{x}_i, \bar{\epsilon}_{min}) + \mathcal{F}(\boldsymbol{x}_i, \bar{\epsilon}_{max})$$

where the term $\mathcal{F}(\boldsymbol{x}_i, \bar{\epsilon}_{min})$ maintains the minimum Jacobian constraint at the value $\bar{\epsilon}_{min}$, while the term $\mathcal{F}(\boldsymbol{x}_i, \bar{\epsilon}_{max})$ (plotted in Figure 10) is handled as a moving barrier that is decreased to reach the desired maximum Jacobian $\bar{\epsilon}_{max}$. As a result, the elements located near the concave parts of the geometry also curve to adapt the high-order boundary, as shown in Figure 9.

In boundary layer-type meshes, where elements have high aspect ratio, the boundary deformation responsible for the mesh tangling is typically large

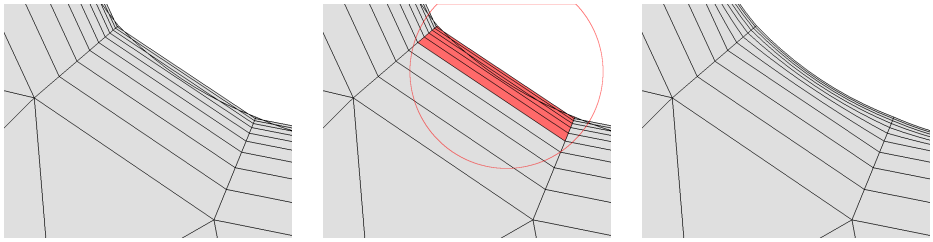


Figure 11: Detail of the mesh near the leading edge of the slat in the high-lift airfoil case: tangled second-order mesh (left), blob definition with a circle representing the geometrical criterion (center), and untangled mesh (right).

compared to the normal size of the tangled element, but small compared to its tangential size (see Figure 11). In order to untangle it, several layers must be curved in the direction normal to the boundary, whereas modifying the elements that are adjacent in the tangential direction is useless. Therefore, the blobs in which we apply the optimization procedure can be constructed by adding a geometrical criterion to the procedure described in Section 5.1, as illustrated in Figure 11: among the N layers of elements surrounding the invalid element, only those located within a certain distance are retained. This distance is defined by multiplying the distance between the straight-sided and high-order boundaries by a user-defined factor. Given the element aspect ratio and grid stretching in the boundary layer zone for this test case, we obtain good results with blobs of $N = 6$ layers of quads (or $N = 12$ layers of triangles) and a distance factor of 12.

We optimize meshes $\mathcal{M}_1$ and $\mathcal{M}_2$ with $\bar{\epsilon}_{min} = 0.4$ and $\bar{\epsilon}_{max} = 1.6$ at order $p = 2 \dots 5$. Information about the results is provided in Table 1. The number of operations needed to compute f and its derivatives depends on the size of matrix $\mathbf{B}$ of Equation (5), i.e. it is a function of $N_p \times N_q$. In table 1, we report N_p and N_q for the different mesh orders p. It must be noted that, for mesh $\mathcal{M}_2$, all elements in the boundary layer are quads, so that N_p and N_q correspond to quads. The number of Bézier points N_q for triangles corresponds to order $q = 2(p-1)$ while the Jacobians of quads are of order $q = 2p-1$. It is therefore not surprising that the computation time for untangling quartic meshes is about 10 times larger than to untangle a quadratic mesh. Quintic meshes ($p = 5$) however are disappointingly expensive to compute for quads. This problem may be essentially related to the bad conditioning of the optimization problem. In our implementation, we use Lagrange equidistant shape functions, which are known to be highly os-

Mesh	p	q	N_p	N_q	$N_{invalid}$	n_v	t(sec)	$J^e_{\min}/J^e_0$
$\mathcal{M}_1$	2	2	6	6	51	20,751	3.0	0.40
$\mathcal{M}_1$	3	4	10	15	56	45,452	8.2	0.40
$\mathcal{M}_1$	4	6	15	28	64	79,967	20.0	0.40
$\mathcal{M}_1$	5	8	21	45	74	124,296	65.2	0.41
$\mathcal{M}_2$	2	3	9	16	32	20,751	1.4	0.55
$\mathcal{M}_2$	3	5	16	36	34	45,452	7.0	0.50
$\mathcal{M}_2$	4	7	25	64	31	79,967	36.3	0.46
$\mathcal{M}_2$	5	9	36	100	31	124,296	330.7	0.44

Table 1: Statistics for the high-lift airfoil test case.

cillatory for high orders. The use of polynomial basis with a better Lebesgue constant such as those of Ref. [26] may improve the results. Nevertheless, our procedure remains quite efficient for high order meshes of orders that are required for engineering analysis, i.e. $p = 2$ and $p = 3$ [27, 28].

Table 1 may look surprising: the target for the minimum scaled Jacobian has been taken as $\bar{\epsilon}_{min} = 0.4$, but the final value of $J^e_{\min}/J^e_0$ is sometimes quite higher than the threshold. Indeed, in the optimization procedure, Jacobians are scaled by a constant value J^e_0 computed on the initial mesh. Yet, corner vertices of elements may be changed in the optimization process, leading to possible changes in values of straight sided Jacobians J^e_0. Thus, the final value of $J^e_{\min}/J^e_0$ could be lower than the threshold, but the positivity of the Jacobian is not threatened as long as the optimization procedure is successful.

Finally, Figure 12 shows a comparison between the linear elastic analogy and the present procedure for the mesh $\mathcal{M}_1$ at $p = 2$. It illustrates the failure of the simple elastic analogy approach in presence of highly stretched boundary layer elements [13], whereas the optimization procedure manages to untangle the mesh by propagating the high-order boundary deformation through the neighboring layers of elements.

6. Mesh curvature impact on the explicit time step: a simple idea

6.1. Another measure of deformation

It is shown in Ref. [29] that the conditioning of the operators deriving from high-order spatial discretization methods can be affected by the curvature of the elements. When using implicit time integration in simulations,

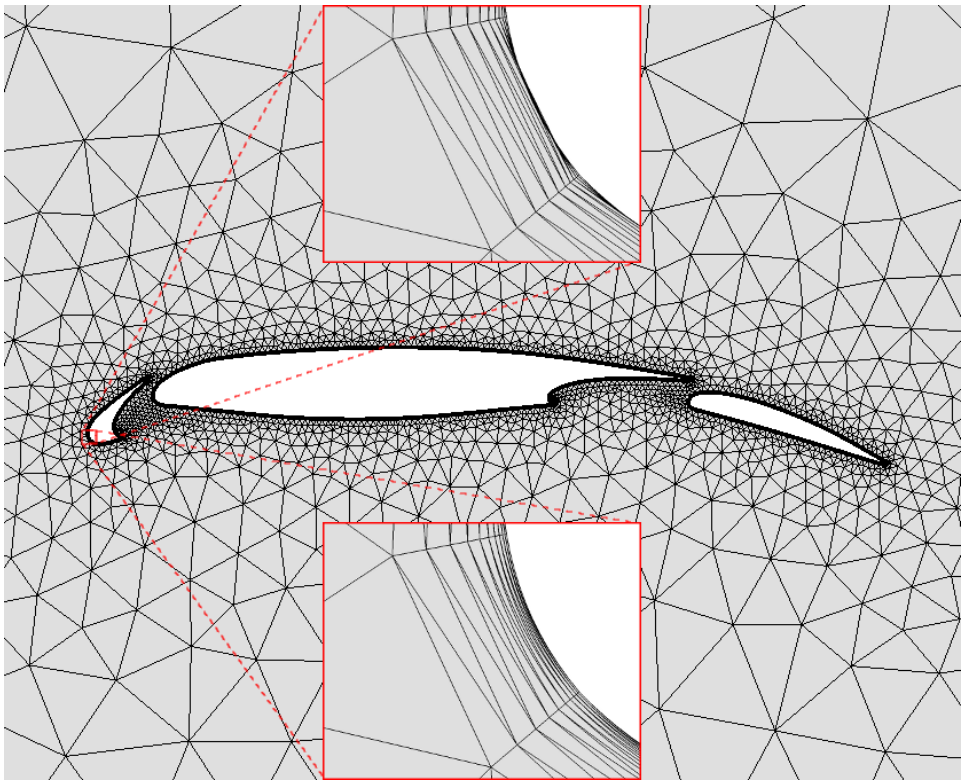


Figure 12: Quadratic triangular mesh for the high-lift airfoil case. Zoom at the leading edge of the slat: comparison between the mesh obtained by elastic analogy (upper frame) and by the present optimization procedure (lower frame).

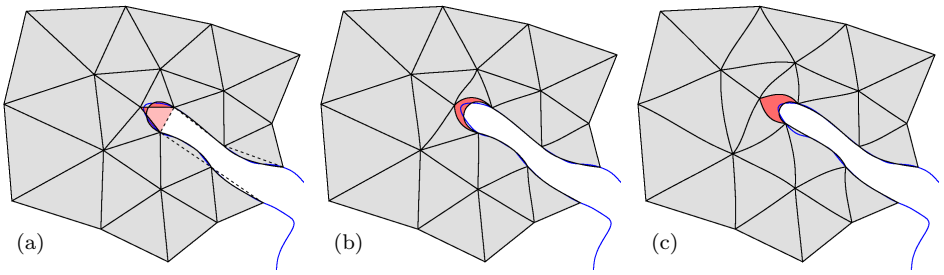


Figure 13: Typical optimization of a two-layer patch around an invalid element of the Great Barrier Reef 3$^{\text{rd}}$ order coarse mesh: (a) initial patch and original linear mesh (in dashed lines), (b) first optimization based on the Jacobian determinant and (c) second optimization based on the minimum eigenvalue of the metric matrix $\mathcal{M}$. The blue line is the coastline given by the CAD model.

this effect has a negative impact on the convergence of iterative linear algebra solvers. With explicit time stepping schemes, it reduces the maximum time step allowed by the stability restrictions, as explained more thoroughly in Section 6.2.

The optimization procedure described in Section 4 is very efficient at producing meshes with scaled Jacobians that are all positive and close to 1. In mathematical terms, the procedure tends to generate mappings $\boldsymbol{x}(\mathbf{X})$ that are one-to-one and close to be *equiareal* (in 2D). As mentioned in Section 4.1, this is due to the term $\mathcal{F}$ that dominates the objective function f for invalid elements. In comparison, the part $\mathcal{E}$ is small, so that the vertices can be moved enough to fix the tangled elements. In other words, the optimization procedure tries to untangle high-order elements by recovering the area in 2D (or the volume in 3D) that they have in the initial straight-sided mesh.

In some cases however, the approximate conservation of the area or volume may not prevent an untangled element to end up with a wildly different shape from the original straight-sided one, as illustrated in Figure 13. This may affect the numerical behaviour of the simulations using such meshes. For instance, consider the three quadratic triangles shown in Figure 14, that have a scaled Jacobian strictly constant and equal to one. It is clear that those triangles have very different interpolation properties in a finite element computation. Prescribing the Jacobian of the elements is thus required to ensure the mesh validity, but it is clearly not sufficient for controlling the mesh quality. The mesh optimization procedure should also be able to control element lengths, in order to avoid problematic cases like the the one shown in Figure 13.

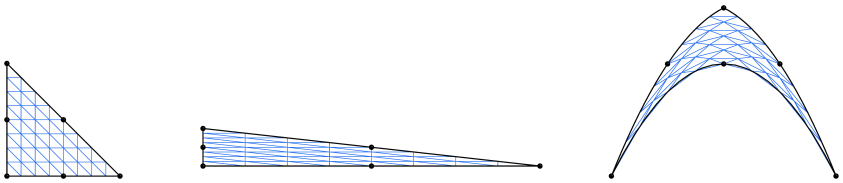


Figure 14: Three second-order triangles with the same area and the same constant Jacobian determinant but obviously leading to different stable time step.

Consider the two triangles of Figure 15. Let $\mathcal{A} = \boldsymbol{x}_{,\boldsymbol{X}}$ be the constant Jacobian matrix of the transformation between the two triangles. The untransformed triangle and the transformed triangle have the same surface ($\det \mathcal{A} = 1$). The first fundamental form or metric tensor of the mapping

$$\mathcal{M} = \mathcal{A}\mathcal{A}^*. \tag{8}$$

allows to compute variations of lengths. Assume $\boldsymbol{U}$ to be a vector that defines a direction in the undeformed space (see Figure 15). Vector $\boldsymbol{U}$ transforms in $\mathcal{A}\boldsymbol{U}$ as it is depicted on the figure. The ratio between the lengths of the two vectors is written as

$$l = \frac{\|\mathcal{A}\boldsymbol{U}\|}{\|\boldsymbol{U}\|} = \sqrt{\frac{\boldsymbol{U}^*\mathcal{M}\boldsymbol{U}}{\boldsymbol{U}^*\boldsymbol{U}}}.$$

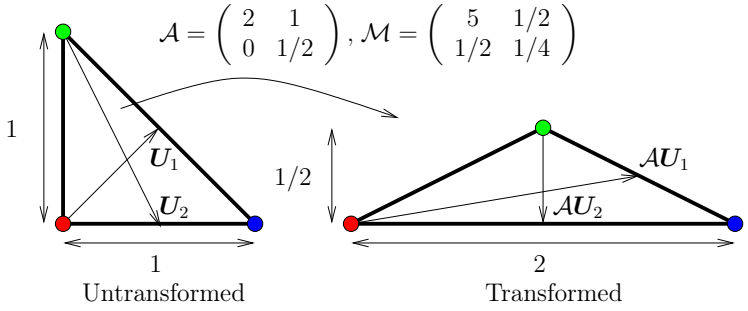


Figure 15: Two triangles that have the same area. We have here $\frac{\|\mathcal{A}\boldsymbol{U}_1\|}{\|\boldsymbol{U}_1\|} = \frac{\sqrt{37}}{2\sqrt{2}} \simeq 2.15$ which is close to $\sqrt{\lambda^{\max}(\mathcal{M})} = \frac{\sqrt{21+\sqrt{377}}}{2\sqrt{2}} \simeq 2.247$ and $\frac{\|\mathcal{A}\boldsymbol{U}_2\|}{\|\boldsymbol{U}_2\|} = \frac{1}{\sqrt{5}} \simeq 0.447$ which is close to $\sqrt{\lambda^{\min}(\mathcal{M})} = \frac{\sqrt{21-\sqrt{377}}}{2\sqrt{2}} \simeq 0.445$.

Then, we have

$$l^{\min} = \min_{\|\boldsymbol{U}\|>0} \frac{\|\mathcal{A}\boldsymbol{U}\|}{\|\boldsymbol{U}\|} = \sqrt{\lambda^{\min}(\mathcal{M})}$$

where $\lambda^{\min}(\mathcal{M})$ is the smallest eigenvalue of $\mathcal{M}$.

Thus, considering the eigenvalues of the metric tensor in addition to the Jacobian gives a better characterization of the mesh quality. This problem is well posed in the sense that the only mapping that result in a metric tensor that has three equal eigenvalues (e.g. the isometry) does not deform the element. In this perspective, we extend our approach not only to obtain positive Jacobians but also to control the minimum eigenvalue of $\mathcal{M}$. We thus look locally at how element lengths are transformed and try to control the minimum value $\lambda^{\min}$ on all the element.

In two dimensions, $\mathcal{M}$ is expressed as:

$$\mathcal{M} = \begin{pmatrix} \|x_{,\boldsymbol{X}}\|^2 & x_{,\boldsymbol{X}} \cdot y_{,\boldsymbol{X}} \\ x_{,\boldsymbol{X}} \cdot y_{,\boldsymbol{X}} & \|y_{,\boldsymbol{X}}\|^2 \end{pmatrix}.$$

And its smallest eigenvalue, $\lambda^{\min}$ can be obtained easily:

$$\lambda^{\min}(\mathcal{M}) = \|x_{,\boldsymbol{X}}\|^2 + \|y_{,\boldsymbol{X}}\|^2 - \sqrt{(\|x_{,\boldsymbol{X}}\|^2 - \|y_{,\boldsymbol{X}}\|^2)^2 + 4(x_{,\boldsymbol{X}} \cdot y_{,\boldsymbol{X}})^2}$$

For each element blob, we introduce a second stage to the optimization process described in Section 4. After the optimization of the Jacobian in the first stage, the same procedure is repeated with a modified objective function:

$$f_2(\boldsymbol{x}_i, \boldsymbol{K}, \bar{\epsilon}, \zeta) = \mathcal{E}(\boldsymbol{x}_i, \boldsymbol{K}) + \mathcal{F}(\boldsymbol{x}_i, \bar{\epsilon}) + \mathcal{G}(\boldsymbol{x}_i, \zeta)$$

The term $\mathcal{F}(\boldsymbol{x}_i, \bar{\epsilon})$ maintains the minimum Jacobian constraint $\bar{\epsilon}$, while the additional term $\mathcal{G}$ controls the minimum eigenvalue of the metric tensor:

$$\begin{aligned} \mathcal{G}(\boldsymbol{x}_i, \zeta) &= \sum_{e=1}^{n_e} \sum_{l=1}^{N_q} G_l^e(\boldsymbol{x}_i^e, \zeta) \\ G_l^e(\boldsymbol{x}_i^e, \zeta) &= \left[\log\left(\frac{\lambda^{\min}(\boldsymbol{x}_i^e) - \zeta}{1 - \zeta}\right)\right]^2 + \left(\lambda^{\min}(\boldsymbol{x}_i^e) - 1\right)^2. \end{aligned}$$

Unfortunately, due to the square root, $\lambda^{\min}$ is not polynomial. Therefore, it cannot be expressed as a sum of Bézier polynomials to guarantee that it is above the objective value everywhere on the element. In practice, the value of $\lambda^{\min}$ is sampled at the same points as the Jacobian and these additional

constraints, combined with those on the Jacobian, are enough to preserve the quality of the elements.

Assuming the usual expansion of Eq. (1) for $\boldsymbol{x}$, the derivative of $\lambda^{\min}$ with respect to nodal positions can be computed as

$$\lambda^{\min}(\mathcal{M})_{,x_i^e} = 2\left(x_{,\boldsymbol{X}} - \frac{(\|x_{,\boldsymbol{X}}\|^2 - \|y_{,\boldsymbol{X}}\|^2)x_{,\boldsymbol{X}} + 2(x_{,\boldsymbol{X}} \cdot y_{,\boldsymbol{X}})y_{,\boldsymbol{X}}}{\sqrt{(\|x_{,\boldsymbol{X}}\|^2 - \|y_{,\boldsymbol{X}}\|^2)^2 + 4(x_{,\boldsymbol{X}} \cdot y_{,\boldsymbol{X}})^2}}\right) \cdot \mathcal{L}_{i,\boldsymbol{X}}^{(p)}$$

$$\lambda^{\min}(\mathcal{M})_{,y_i^e} = 2\left(y_{,\boldsymbol{X}} - \frac{(\|y_{,\boldsymbol{X}}\|^2 - \|x_{,\boldsymbol{X}}\|^2)y_{,\boldsymbol{X}} + 2(x_{,\boldsymbol{X}} \cdot y_{,\boldsymbol{X}})x_{,\boldsymbol{X}}}{\sqrt{(\|x_{,\boldsymbol{X}}\|^2 - \|y_{,\boldsymbol{X}}\|^2)^2 + 4(x_{,\boldsymbol{X}} \cdot y_{,\boldsymbol{X}})^2}}\right) \cdot \mathcal{L}_{i,\boldsymbol{X}}^{(p)}.$$

Figure 13 shows how optimization on $\lambda^{\min}$ affects the resulting meshes. Left Figure shows the initial invalid mesh. Algorithm 1 is then applied, resulting in the mesh of the central figure. This mesh, while perfectly valid, has elements that are dramatically shrinked in some directions. Finally, optimization on $\lambda^{\min}$ is applied, resulting on the mesh of the right Figure. This mesh has element shapes which are very close to those of the straight-sided mesh.

6.2. Impact of the deformation with explicit time stepping

It is mentioned in Section 6.1 that the deformation of high-order elements may affect the numerical methods used in simulations through the mapping $\boldsymbol{x}(\boldsymbol{\xi})$. Hereafter, we explain how these effects can be characterized by the deformation measure introduced in Section 6.1, in the particular case of numerical methods using explicit time stepping schemes to resolve unsteady phenomena.

With explicit time stepping schemes, the curvature of the elements reduces the maximum time step allowed by the stability restrictions through the Jacobian matrix $\boldsymbol{x}_{,\boldsymbol{\xi}}$ of the mapping $\boldsymbol{x}(\boldsymbol{\xi})$, as shown in Ref. [29]. This is illustrated in Figure 16, where the evolution of the maximum time step is plotted while progressively deforming a grid, for a simple advection problem solved with a DG spatial discretization and an explicit Runge-Kutta time integrator. Thus, setting a scaled Jacobian barrier $\bar{\epsilon}$ closer to 1 tends to mitigate the adverse effects of high-order mesh curvature.

However, in the simple example of Figure 16, the elements are deformed in a constant direction. The general results of Ref. [29], as well as our experience with high-order computations, suggest that there is no simple relation between the conditioning of the semi-discrete operator and the Jacobian $\det \boldsymbol{x}_{,\boldsymbol{\xi}}$.

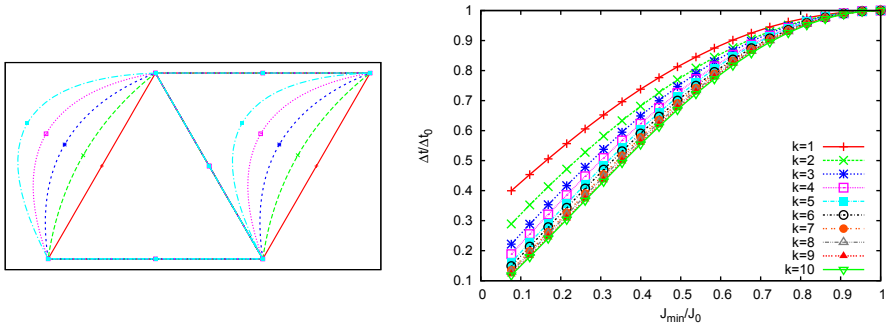


Figure 16: Impact of the progressive deformation of a structured grid for a scalar advection problem, solved with the classic explicit 4-stage Runge-Kutta time integrator and a DG spatial discretization of order k ranging from 1 to 10: deformation of the pattern of triangular elements of which the structured grid is made up (left), evolution of the time step Δt (scaled by the time step Δt_0 for straight-sided elements) as a function of the scaled Jacobian ${}^{J_{\min}}/_{J_0}$ (right). More details can be found in Ref. [29].

Consider a simple scalar advection equation, as a model for more general hyperbolic systems: find $u(\boldsymbol{x}, t)$ solution of

$$\frac{\partial u}{\partial t} + \boldsymbol{V} \cdot \nabla_{\boldsymbol{x}} u = 0. \tag{9}$$

The explicit integration in time of the ordinary differential equation resulting from the discretization of the spatial operator $\boldsymbol{V} \cdot \nabla_{\boldsymbol{x}}$ is subject to conditional stability. Linear stability conditions are usually more restrictive than non-linear stability conditions, so the stability results obtained for the conservation law (9) can be applied to non-linear partial differential equations (PDE's). These conditions can further be generalized to systems of hyperbolic PDE's by considering that u corresponds to a characteristic variable of the system and $\boldsymbol{V}$ corresponds to the associated characteristic velocity. The stability of the system is then driven by the characteristic leading to the most restrictive stability condition. However, the systems of multidimensional PDE's of practical interest often have an infinite set of characteristics forming a Monge cones instead of degenerating into lines. Thus, we are interested in a stability condition that takes all possible directions of $\boldsymbol{V}$ into consideration, i.e. a condition on the norm $\|\boldsymbol{V}\|$ of $\boldsymbol{V}$.

This so-called Courant-Friedrichs-Levy (CFL) condition can be defined as:

$$\|\boldsymbol{V}\| \, \Delta t \leq C \, \Delta x \tag{10}$$

where Δt is the stable time step, Δx is a length that represents the element size, and C a constant that depends on the numerical method [30]. This condition should be fulfilled everywhere on the element. In the case of straight sided triangular elements, Δx is usually chosen as the inner radius of the triangle.

How does this size definition extend to curvilinear meshes? In order to find a stability condition of the advection problem on the curved element, we transform it into a problem on a straight-sided element for which we can use the stability condition (10). The conservation law (9) is written in a system of coordinates, $\boldsymbol{x}(\boldsymbol{X})$, which can be seen as the transformation of another one, $\boldsymbol{X}$ corresponding to a straight-sided element. The Jacobian of this transformation is $\mathcal{A} = \boldsymbol{x}_{,\boldsymbol{X}}$. In the system of coordinates $\boldsymbol{X}$, the conservation law (9) reads:

$$\frac{\partial u}{\partial t} + \underbrace{\boldsymbol{V} \cdot \mathcal{A}^{-1}}_{\boldsymbol{V}'} \cdot \nabla_{\boldsymbol{X}} u = 0. \tag{11}$$

The problems (9) and (11) being equivalent, the stability condition (10) can be written in the system of coordinates $\boldsymbol{X}$ as:

$$\|\boldsymbol{V}'\| \, \Delta t = \|\boldsymbol{V} \cdot \mathcal{A}^{-1}\| \, \Delta t < C \, \Delta X$$

where ΔX is the size of the straight-sided element. Let $l^{\min}$ be the smallest singular value of $\mathcal{A}$ so that $\frac{1}{l^{\min}}$ is the largest singular value of $\mathcal{A}^{-1}$. We have:

$$\|\boldsymbol{V}\mathcal{A}^{-1}\| < \frac{\|\boldsymbol{V}\|}{l^{\min}}.$$

The condition:

$$\|\boldsymbol{V}\| \, \Delta t < C \, \frac{\Delta X}{l^{\min}}. \tag{12}$$

is then sufficient to satisfy the stability condition (12). Condition (10) can be seen as the usual CFL condition (10) with an element size $\Delta x = \Delta X / l^{\min}$ depending on the smallest singular value $l^{\min}$ of the mapping $\mathcal{A}$, i.e. the smallest eigenvalue of the metric tensor $\mathcal{M} = \mathcal{A}\mathcal{A}^T$, as defined in Section 6.1.

6.3. Application on a large scale example

In order to show the interest of the deformation measure introduced in Section 6.1 and Section 6.2 for practical applications, we apply the mesh optimization procedure to a large-scale oceanic simulation.

The patch of Figure 13 is a very small part of a larger mesh of the Great Barrier Reef (GBR), in Australia. The GBR is composed of more than 2500

reefs and islands. This complex topography generates circulation patterns on a wide range of scales and makes the generation of a curved mesh of this domain particularly challenging. Second-, third- and fourth-order versions of two linear meshes of this region are considered. The first mesh is composed of about 330,000 triangles and the second one, even coarser, is composed of about 88,000 triangles. Those meshes are presented on Figure 17. In order to test our method on a difficult case, we use in both meshes a coarse resolution compared to the small features of the underlying geometrical model, so that highly curved elements are generated near the boundaries.

On each mesh, we determine the maximum stable time step to solve the shallow water equations using a discontinuous Galerkin method of the same polynomial order as the mesh with a fourth-order explicit Runge-Kutta temporal scheme. The details of the parameterization and the boundary conditions can be found in [31]. For each mesh, an estimation of the maximum time step is progressively refined through a simple bisection method that determines the stability by running the simulation for a few iterations and checking the global norm of the solution.

Table 2 shows the different time steps obtained on linear meshes, on curvilinear meshes optimized with respect to the scaled Jacobians only and on meshes further optimized with respect to λ^{min}. By chance, the Jacobian-based optimization is enough to recover the same time step on the second-order mesh as on the linear mesh with a requirement of $J^e_{\min}/J^e_0 = 0.5$. However, this is not the case for the other meshes, even when a high scaled Jacobian is imposed. On the contrary, on all meshes optimized first with respect to the scaled Jacobians and then with respect to λ^{min} (using an objective value of 0.5 in both optimizations), the maximum stable time step is very close to the maximum stable time step of the same finite element method on the linear mesh. The CPU time taken by the optimization process ranges from 5 seconds for the second-order coarse mesh to about 10 minutes for the fourth-order fine mesh.

7. Conclusions

The ability of generating high order/curvilinear mesh is a prerequisite to the generalization of high order numerical methods in engineering analysis. This paper offers a framework that allows to generate such meshes in a robust and efficient way.

The methodology consists in solving a sequence of optimization problems in which the deformation of high-order mesh elements, compared to their linear counterpart, is minimized. The deformation measure is based

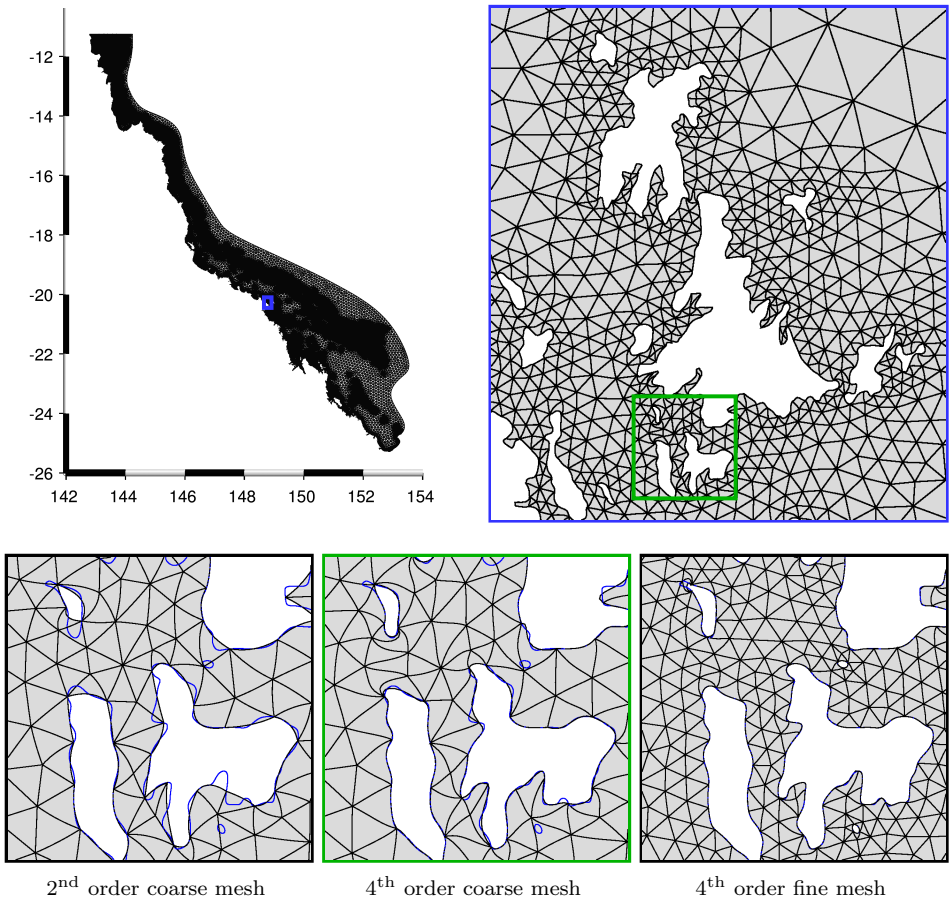


Figure 17: Top: fourth-order coarse curvilinear mesh (88k triangles) of the Great Barrier Reef, Australia and detail on the Whitsunday islands. Below: closer zoom on Hamilton Island for different meshes.

	p	$J^e_{\min}/J^e_0$		$\lambda^{\min}$		Linear
		0.1	0.5	0.1	0.5	
Coarse	2	1.7	3.0	2.5	3.1	3.0
	3	0.8	0.9	2.0	2.0	1.9
	4	0.5	0.5	1.2	1.2	1.2
Fine	2	0.8	1.0	1.4	3.1	3.0
	3	0.4	0.5	1.4	2.0	2.0
	4	0.4	0.5	0.9	1.3	1.4

Table 2: Maximum stable time step to solve the shallow water with discontinuous Galerkin method of polynomial order 2 to 4 and an explicit fourth order Runge-Kutta, on linear meshes, curvilinear meshes optimized for the Jacobians only and curvilinear meshes further optimized for the metric. The optimization process stops when all the Jacobians or the eigenvalues of the metric are greater than 0.1 or 0.5.

on the Jacobian of the transformation that maps a high-order element onto the straight-sided one, which enables to strictly controls the mesh validity. Optionally, the minimum eigenvalue of the corresponding metric can be used in a second step, with positive impact on simulations involving explicit time stepping. Constraints on the maximum deformation are imposed through moving barriers, in order to guarantee the quality of the high-order mesh generated. The restriction of the optimization process to blobs of elements surrounding a tangled element, instead of the whole mesh, make the methods computationally efficient.

We have tested the new untangling procedure on various other examples than those presented in this paper, and our conclusions can be summarized as follows:

- The use of moving log-barriers for imposing Jacobian constraints is clearly a better solution than using constrained optimization, both in terms of computational cost and of robustness.

- The energy part (6) of the objective function is not a critical parameter in the untangling process, its only purpose being most of the time to ensure a unique solution to the optimization process.

- It is of paramount importance to allow mesh vertices to move on the model entities they are classified on. This is especially true in 3D where both surface and volume meshes should be untangled at once.

- The untangling of quadratic meshes using the new technique is very

fast, robust and not sensitive to any parameter that we have introduced. We consider that issue to be well addressed.

All the material that is presented in this paper is readily available in Gmsh, the open source mesh generator[9]. We are confident that users will help us to improve further Gmsh's high-order mesh generation capabilities. Nevertheless, some direct extensions of this work are already planned.

First, the objective function should somehow take into account the geometrical accuracy of the high order mesh representations. High order vertices should allow to reduce some norm of the distance between the mesh and the CAD model.

Then, the influence of element shapes on the numerical solutions should be investigated. In this paper, we have shown that it is possible to compute a length quantity representing the size of a curvilinear element and use it to compute the CFL condition. We should now investigate the influence of mesh curving on the quality of numerical solutions, especially in computational fluid dynamics.

Acknowledgement

This research project was funded in part by the Walloon Region under WIST 3 grant 1017074 DOMHEX (Dominant Hexahedral Mesh Generation) and in part by the European Union under the 7th PCRD grant IDIHOM (Industrialisation of High-Order Methods - A Top-Down Approach).

[1] A. Johnen, J.-F. Remacle, C. Geuzaine, Geometrical validity of curvilinear finite elements, in: W. R. Quadros (Ed.), Proceedings of the 20th International Meshing Roundtable, Springer Berlin Heidelberg, 2012, pp. 255–271. doi:10.1007/978-3-642-24734-7_14.

[2] A. Johnen, J.-F. Remacle, C. Geuzaine, Geometric validity of curvilinear finite elements, Journal of Computational Physics.

[3] B. Cockburn, G. Karniadakis, C.-W. Shu (Eds.), Discontinuous Galerkin Methods, Vol. 11 of Lecture Notes in Computational Science and Engineering, Springer, Berlin, 2000.

[4] N. Kroll, H. Bieler, H. Deconinck, V. Couaillier, H. Van Der Ven, K. Sorensen, ADIGMA – A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications: Results of a Collaborative Research Project Funded by the

European Union, 2006-2009, Notes on Numerical Fluid Mechanics and Multidisciplinary Design, Springer, 2010.

[5] F. Bassi, S. Rebay, High-order accurate discontinuous finite element solution of the 2D Euler equations, J. Comput. Phys. 138 (2) (1997) 251–285. doi:10.1006/jcph.1997.5454.

[6] P.-E. Bernard, J.-F. Remacle, V. Legat, Boundary discretization for high-order discontinuous Galerkin computations of tidal flows around shallow water islands, International Journal for Numerical Methods in Fluids 59 (5) (2009) 535–557. doi:10.1002/fld.1831.

[7] T. Toulorge, W. Desmet, Curved boundary treatments for the discontinuous Galerkin method applied to aeroacoustic propagation, AIAA J. 48 (2) (2010) 479–489. doi:10.2514/1.45353.

[8] J.-F. Remacle, M. S. Shephard, An algorithm oriented mesh database, International Journal for Numerical Methods in Engineering 58 (2) (2003) 349–374. doi:10.1002/nme.774.

[9] C. Geuzaine, J.-F. Remacle, Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering 79 (11) (2009) 1309–1331.

[10] P. L. George, H. Borouchaki, Construction de maillages de degré 2 partie 3 : Tétraèdre p2, Tech. rep., INRIA (France) (2011).

[11] P. M. Knupp, Winslow smoothing on two-dimensional unstructured meshes, Engineering with Computers 15 (1999) 263–268. doi:10.1007/s003660050021.

[12] Z. Q. Xie, R. Sevilla, O. Hassan, K. Morgan, The generation of arbitrary order curved meshes for 3d finite element analysis, Computational Mechanics.

[13] P.-O. Persson, J. Peraire, Curved mesh generation and mesh refinement using lagrangian solid mechanics, in: Proceedings of the 47th AIAA Aerospace Sciences Meeting and Exhibit, Orlando (FL), USA, 5-9 January 2009, 2009.

[14] X. Li, M. Shephard, M. Beall, Accounting for curved domains in mesh adaptation, International Journal for Numerical Methods in Engineering 58 (2) (2003) 247–276.

[15] X. Luo, M. Shephard, R. O'Bara, R. Nastasia, M. Beall, Automatic p-version mesh generation for curved domains, Engineering with Computers 20 (3) (2004) 273–285.

[16] M. Shephard, J. Flaherty, K. Jansen, X. Li, X. Luo, N. Chevaugeon, J. Remacle, M. Beall, R. O'Bara, Adaptive mesh generation for curved domains, Applied Numerical Mathematics 52 (2) (2005) 251–271.

[17] L. A. Freitag, P. Plassmann, Local optimization-based simplicial mesh untangling and improvement, International Journal for Numerical Methods in Engineering 49 (1) (2000) 109–125.

[18] E. J. López, N. M. Nigro, M. A. Storti, J. A. Toth, A minimal element distortion strategy for computational mesh dynamics, International Journal for Numerical Methods in Engineering 69 (2007) 1898–1929.

[19] E. J. López, N. M. Nigro, M. A. Storti, Simultaneous untangling and smoothing of moving grids, International Journal for Numerical Methods in Engineering 76 (7) (2008) 994–1019.

[20] L. Diachin, P. Knupp, T. Munson, S. Shontz, A comparison of two optimization methods for mesh quality improvement, Engineering with Computers 22 (2) (2006) 61–74.

[21] L. Freitag, P. Knupp, T. Munson, S. Shontz, A comparison of optimization software for mesh shape-quality improvement problems., Tech. rep., Argonne National Lab., IL (US) (2002).

[22] A. Fiacco, G. McCormick, Nonlinear programming: sequential unconstrained minimization techniques, Vol. 4, Society for Industrial Mathematics, 1990.

[23] A. Waechter, C. Laird, F. Margot, Y. Kawajir, Introduction to ipopt: A tutorial for downloading, installing, and using ipopt (2009).

[24] D. Liu, J. Nocedal, On the limited memory bfgs method for large scale optimization, Mathematical programming 45 (1) (1989) 503–528.

[25] R. Fletcher, C. Reeves, Function minimization by conjugate gradients, The computer journal 7 (2) (1964) 149–154.

[26] J. Hesthaven, T. Warburton, Nodal discontinuous Galerkin methods: algorithms, analysis, and applications, Vol. 54, Springer-Verlag New York Inc, 2008.

[27] P. Vos, S. Sherwin, R. Kirby, From h to p efficiently: Implementing finite and spectral hp element methods to achieve optimal performance for low-and high-order discretisations, Journal of Computational Physics 229 (13) (2010) 5161–5181.

[28] P. Bernard, J. Remacle, R. Comblen, V. Legat, K. Hillewaert, High-order discontinuous galerkin schemes on general 2d manifolds applied to the shallow water equations, Journal of Computational Physics 228 (17) (2009) 6514–6535.

[29] T. Toulorge, W. Desmet, Spectral properties of discontinuous Galerkin space operators on curved meshes, in: J. S. Hesthaven, E. M. Rønquist, T. J. Barth, M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, T. Schlick (Eds.), Spectral and High Order Methods for Partial Differential Equations, Vol. 76 of Lecture Notes in Computational Science and Engineering, Springer Berlin Heidelberg, 2011, pp. 495–502. doi:10.1007/978-3-642-15337-2_48.

[30] B. Seny, J. Lambrechts, R. Comblen, V. Legat, J.-F. Remacle, Multirate time stepping for accelerating explicit discontinuous galerkin computations with application to geophysical flows, International Journal for Numerical Methods in Fluids in press. doi:10.1002/fld.3646.

[31] J. Lambrechts, E. Hanert, E. Deleersnijder, P.-E. Bernard, V. Legat, J.-F. Remacle, E. Wolanski, A multiscale model of the whole Great Barrier Reef hydrodynamics, Estuarine, Coastal and Shelf Science 79 (2008) 143–151. doi:10.1016/j.ecss.2008.03.016.