Robust Untangling of Curvilinear Meshes

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

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

Summary. This paper describes a technique that enables to generate high order / curvilinear meshes in a robust fashion. Accurate estimates of jacobian bounds are used for deriving an unconstrained optimization procedure. Both 2D and 3D valid high order meshes are presented that demonstrate the efficiency of the new technique.

1 Introduction

There is a growing consensus that state of the art Finite Volume technology requires, and will continue to require too extensive computational resources to provide the necessary resolution, even at the rate that computational 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) 2nd order provided by most industrial grade codes. This indicates that higher-order discretization methods will replace at some point the finite volume solvers of today, at least for part of their applications.

The development of high-order numerical technologies for CFD is underway for many years now. For example, Discontinuous Galerkin methods (DGM) have been largely studied in the literature, initially in a quite theoretical context [1], and now in the application point of view [2]. In many contributions, it is shown that the accuracy of the method strongly depends on the accuracy of the geometrical discretzation[3, 4, 5]. In consequences, the following question should be asked: how do we get the high order meshes that will be used by high order methods?

Modern mesh generation procedures take as input CAD models. In such models, four kinds of *model entities* [6] are defined: model vertices G_i^0 that are entities of dimension 0, model edges G_i^1 that are entities of dimension 1, model faces G_i^2 that are entities of dimension 2 and model regions G_i^3 that are entities of dimension 3.

Each model entity G_i^d has a shape, a geometry. The geometry of a model entity depends on the solid modeler for its underlying representation. Solid

modelers usually provide a parametrization of the shapes, i.e., 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 3-D 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 to a model region 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[1]. 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 the high order points can be chosen in such a way that the geometrical error, i.e. the distance between the CAD model and the mesh is minimized.

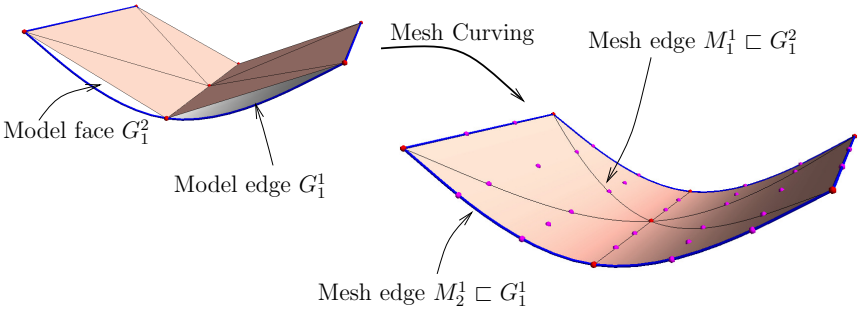

Fig. 1. Straight sided mesh (left) and curvilinear (cubic) mesh (right)

The naive curving procedure described just above do not ensure that the final curved mesh is only composed of valid elements. Figure 2 gives an illustration of that important issue. Some of the curved triangles are tangled: they self-intersect after having been curved. It is important to note that the mesh would not be valid even if the curved edge was assigned the exact geometry (blue curve on Figure 2).

Invalid elements may be detected by exploiting specific properties of the Jacobian. In a recent paper [7], a general formulation has been developed

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

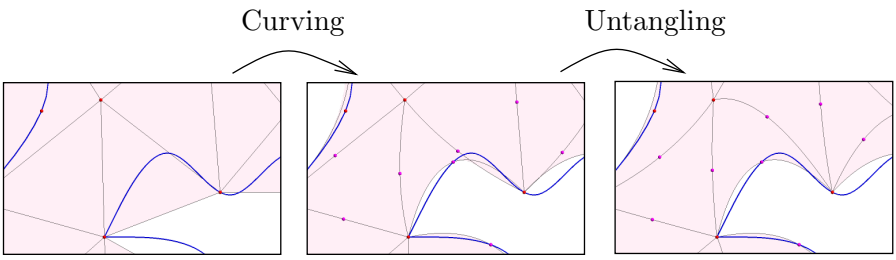

Fig. 2. Straight sided mesh (left) basic curvilinear (quadratic) mesh (center) with tangled elements and untangled mesh (right)

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. It is necessary to propagate the curvature inside the domain through some kind of smoothing scheme. Some smoothing schemes have been proposed in the literature. Those include linear smoothing techniques such as Laplacian smoothing [8], Winslow smoothing [9] or linear elasticity with varying stiffness [8]. 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.

Other authors [10, 11, 12] make use of mesh adaptation techniques by eliminating invalid elements by a combination of local mesh refinements, edge and face swaps, and node relocations.

Finally, authors of [13] propose an approach to deform a given mesh into a curved boundary conforming mesh through a nonlinear elasticity analogy. This approach definitively result in a valid curvilinear mesh. Yet, computing a non-linear mechanics problem including large deformations on a high-order and highly stretched mesh is at least as complex as, let's say, solving Navier-Stokes equations on the same grid.

In this paper, we propose a robust smoothing scheme that allows to build a curvilinear mesh for which every element is guaranteed to be valid at convergence. This new untangling procedure does not rely on any analogy: it specifically targets element jacobians and modifies node locations in such a way that jacobian values sits in a predefined range.

In §2, we briefly recall the results of paper [7] on jacobian bounds. Sensitivity of jacobians are computed with respect to the motion of mesh vertices. The next section §3 is dedicated to the practical computation of both jacobian bounds and their derivatives with respect to the motion of mesh vertices. Then in §4, an objective function that specifically targets invalid jacobians

is build. Constraints on jacobian positivity are imposed through log-barriers, allowing the use of unconstrained optimization procedures. In §5, an optimization strategy is described. The optimization starts with an invalid mesh and the asymptote in the log barrier is progressively moved into the valid region.

Finally, some examples are presented in §6 that demonstrate both the robustness and the efficiency of the new methodology. Both 2D and 3D examples are presented with timings and jacobian ranges.

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.

We note $\boldsymbol{X}_i^e$ the position of the ith node of the element e in the straight sided configuration and $\boldsymbol{x}_i^e$ the location of the same node, yet in some deformed configuration.

The shape of an element e is defined geometrically through their nodes $\boldsymbol{x}_i^e$, $i = 1 \dots N_p$ and a set of Lagrange shape functions $\mathcal{L}_i^{(p)}(\boldsymbol{\xi})$, $i = 1 \dots N_p$ at order p that allow 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). Mapping $\boldsymbol{x}(\boldsymbol{X})$ should be bijective i.e. it should admits an inverse. This implies that the determinant of the jacobian $\det \boldsymbol{x}_{,\boldsymbol{X}}$ has to be strictly positive, for every value of ξ and η. It is possible to write this determinant in terms of the $\boldsymbol{\xi}$ coordinates as:

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

where J_0^e is the strictly positive and constant[2] jacobian of the straight sided element. Function $\boldsymbol{x}(\boldsymbol{X})$ is called the distorsion 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 [7], it has been 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. In [7], we first remark that the jacobian J is a polynomial in $\boldsymbol{\xi}$. Jacobian determinants J can then be interpolated exactly as a linear combination of Bézier polynomials $\mathcal{B}_i^{(q)}$ at a

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

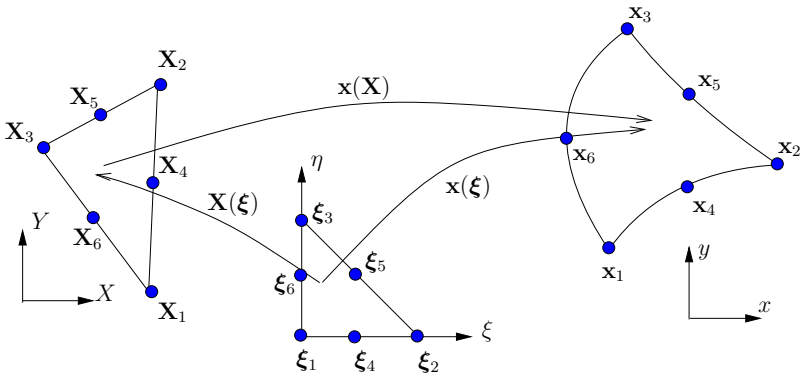

Fig. 3. Reference unit triangle in local coordinates $\boldsymbol{\xi} = (\xi, \eta)$ and the mappings $\mathbf{x}(\boldsymbol{\xi})$, $\mathbf{X}(\boldsymbol{\xi})$ and $\mathbf{X}(\mathbf{x})$

certain order $q \geq p$ over the element. Provable bounds for $J_{\min}$ and $J_{\max}$ are then computed using some interesting properties of the Bézier polynomials. Assuming that J is a polynomial at order q in $\boldsymbol{\xi}$, we write

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

and bounds can be computed as

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

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

3 Computation of Bézier Coefficients and Their Derivatives

The aim of our method is to be able 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:

$$J_k = J(\boldsymbol{\xi}_k) = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \zeta} + \frac{\partial z}{\partial \xi}\frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \zeta} + \frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \eta}\frac{\partial x}{\partial \zeta} - \frac{\partial z}{\partial \xi}\frac{\partial y}{\partial \eta}\frac{\partial x}{\partial \zeta} - \frac{\partial x}{\partial \xi}\frac{\partial z}{\partial \eta}\frac{\partial y}{\partial \zeta} - \frac{\partial y}{\partial \xi}\frac{\partial x}{\partial \eta}\frac{\partial z}{\partial \zeta}. \quad (2)$$

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 node i:

$$\frac{\partial J_k}{\partial x_i^e} = \frac{\partial \mathcal{L}_i^{(p)}}{\partial \xi}\frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \zeta} + \frac{\partial z}{\partial \xi}\frac{\partial \mathcal{L}_i^{(p)}}{\partial \eta}\frac{\partial y}{\partial \zeta} + \frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \eta}\frac{\partial \mathcal{L}_i^{(p)}}{\partial \zeta} - \frac{\partial z}{\partial \xi}\frac{\partial y}{\partial \eta}\frac{\partial \mathcal{L}_i^{(p)}}{\partial \zeta} - \frac{\partial \mathcal{L}_i^{(p)}}{\partial \xi}\frac{\partial z}{\partial \eta}\frac{\partial y}{\partial \zeta} - \frac{\partial y}{\partial \xi}\frac{\partial \mathcal{L}_i^{(p)}}{\partial \eta}\frac{\partial z}{\partial \zeta}. \quad (3)$$

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_l, matrix

$$\mathbf{B} = \begin{bmatrix} \frac{\partial B_1}{\partial x_1^e} & \cdots & \frac{\partial B_1}{\partial x_{N_p}^e} & \frac{\partial B_1}{\partial y_1^e} & \cdots & \frac{\partial B_1}{\partial y_{N_p}^e} & \frac{\partial B_1}{\partial z_1^e} & \cdots & \frac{\partial B_1}{\partial z_{N_p}^e} & B_1 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\ \frac{\partial B_{N_q}}{\partial x_1^e} & \cdots & \frac{\partial B_{N_q}}{\partial x_{N_p}^e} & \frac{\partial B_{N_q}}{\partial y_1^e} & \cdots & \frac{\partial B_{N_q}}{\partial y_{N_p}^e} & \frac{\partial B_{N_q}}{\partial z_1^e} & \cdots & \frac{\partial B_{N_q}}{\partial z_{N_p}^e} & B_{N_q} \end{bmatrix}. \quad (4)$$

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 B_i's and that allow to control the quality of elements is constructed.

4 An Objective Function for Curvilinear Mesh Optimization

This section describes the objective function $f(\boldsymbol{x}_i^e)$ that will serve for untangling invalid curved elements though 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 these kinds of meshes, we want to preserve as much as possible all these features of the mesh, which means keeping the nodes as close as possible to their initial positions in the straight sided mesh.

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

$$\mathcal{E}(\mathbf{K}, \boldsymbol{x}_i) = \frac{1}{2} \sum_e \sum_{i=1}^{N_p} \sum_{i=1}^{N_p} (\boldsymbol{x}_i^e - \boldsymbol{X}_i^e) \mathbf{K}_{ij} (\boldsymbol{x}_j^e - \boldsymbol{X}_j^e) \geq 0 \tag{5}$$

where $\mathbf{K}$ is a symmetric positive matrix of size $3n_v \times 3n_v$ and where $\mathbf{K}_{ij}$ is of size 3×3. In this paper, we choose $\mathbf{K}$ as the identity matrix multiplied by a constant factor κ. The determination of κ, even though it is not determinant in the convergence of the procedure, will be detailed in the examples section §6.

The second part of the functional $\mathcal{F}$ deals with jacobian positivity. We use a log barrier [14] in order to avoid jacobians that are too small and a quadratic function to penalize jacobians that are too high:

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

with

$$F_l^e(\boldsymbol{x}_i^e, \epsilon) = \left[\log \left(\left[\frac{1+2\epsilon}{1+\epsilon} \right] \frac{B_l^e(\boldsymbol{x}_i^e)}{J_0^e} - \frac{\epsilon}{1+\epsilon} \right) \right]^2 + \left(\frac{B_l^e(\boldsymbol{x}_i^e)}{J_0^e} - 1 \right)^2, \tag{6}$$

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 will converge to at least a local minimum in most cases, regardless of the convexity characteristics of the objective function and constraints [15]. Figure 4 shows the evolution of our barrier function for $\epsilon = 0.3$.

In the optimization process, we distinguish three sort of mesh vertices. Some 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 single parameter t, t being the parameter of the curve. We have therefore

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

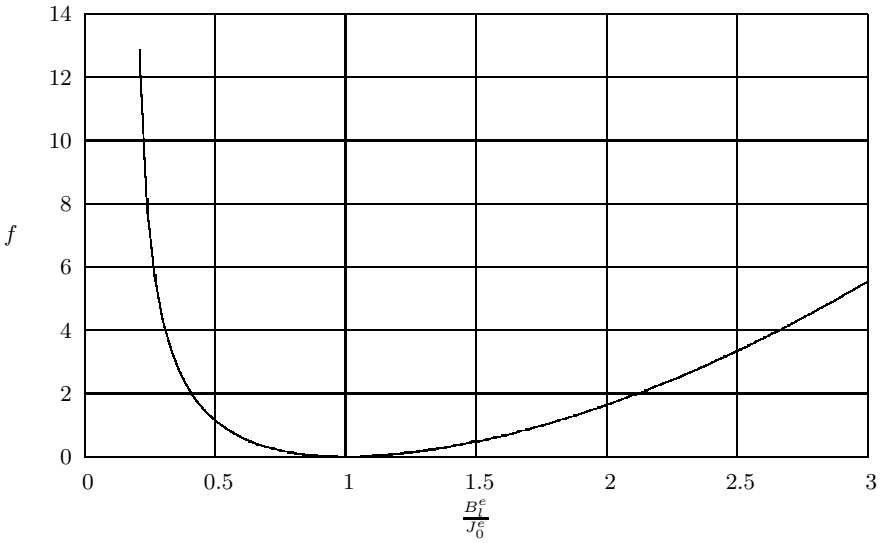

Fig. 4. Evolution of the barrier function $F(J_l^e)$ for $\epsilon = 0.3$

with $\frac{d\boldsymbol{x}_i^e}{dt}$ the tangent vector to the curve at point t that can be computed with the CAD model API.

Other vertices that are classified on model faces $M_i^0 \sqsubset G_j^2$ can only be moved on the surface. In this case, 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. Finally, mesh vertices that are classified on model vertices have no freedom to move.

5 Optimization Strategy

The problem of untangling curvilinear meshes has been 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 allow to solve unconstrained minimization problems. Here, we have tested a number of alternatives: IPOPT [16], LBFGS

[17] and conjugate gradients [18]. At the end, the use of conjugate gradients seemed to be the best choice in term of computational efficiency.

The most important part of the optimization strategy is to define the right sequence of optimization problems.

- The optimization should not applied to the whole mesh but locally. Blobs of elements that surround an invalid element are constructed. Mesh vertices that are on the boundary of the blob are fixed.
- Appropriate scaling is applied to optimization variables that are associated to parametric coordinates of mesh vertices that are classified on either a model edge or a model face.
- The mesh being initially wrong, the evaluation of (6) returns "not a number". We compute therefore a sequence of optimization problems with "moving barriers".

The optimization strategy is described in Algorithm 1.

Algorithm 1. Optimization strategy

```
1  Compute element blobs B_k, k = 1 ... N_B ;
2  for k = 1 to N_B do
3      repeat
4          compute κ = min_e min_l B_l^e / J_0^e,   e ∈ B_i,  l ∈ [1, N_q];
5          if κ < ε then
6              set ε̄ = 1.1 κ;
7          else
8              set ε̄ = ε;
9          solve min_{x_i} f(x_i, K, ε̄) for all elements of blob B_k;
10     until ε̄ < ε;
```

As an example, consider a coarse 3D tetrahedral mesh of a sphere, as presented in Figure 5. The surface of the sphere is described in the CAD system as one only patch that covers the whole range of sherical coordinates. In order to challenge our optimization strategy, high order nodes that are classified on the surface have been added along lines in the parameter plane. More precisely, high order points are added on every edge that is classified on the surface of the sphere in such a way that this edge remains straight sided in the parameter plane. Elements that are close to the poles are very distorded and the resulting mesh that is presented in the middle image of Figure 5 is clearly wrong. Our untangling strategy has then been successfully applied to the invalid mesh: the final valid mesh that is presented in Figure 5 has all elements with scaled jacobians in the range $[0.9, 1.1]$. Less than one second was required for converging.

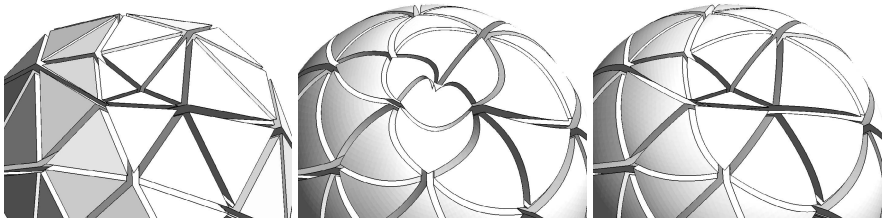

Fig. 5. Example of mesh untangling with mesh vertices motions on manifolds. The straight sided mesh (left) is made quadratic (center) and is subsequently untangled (right).

6 Examples

In this section, the new optimization scheme is applied to several 2D and 3D high order meshes. In our experiments, the untangling of 2D meshes, even when those are complex, is a matter of seconds. Figure 6 presents a quadratic mesh of a three component wing. Structured boundary layers have been generated on the three components. Untangling the quadratic mesh has taken 2.2 seconds. A cubic mesh has also been generated and has been untangled in

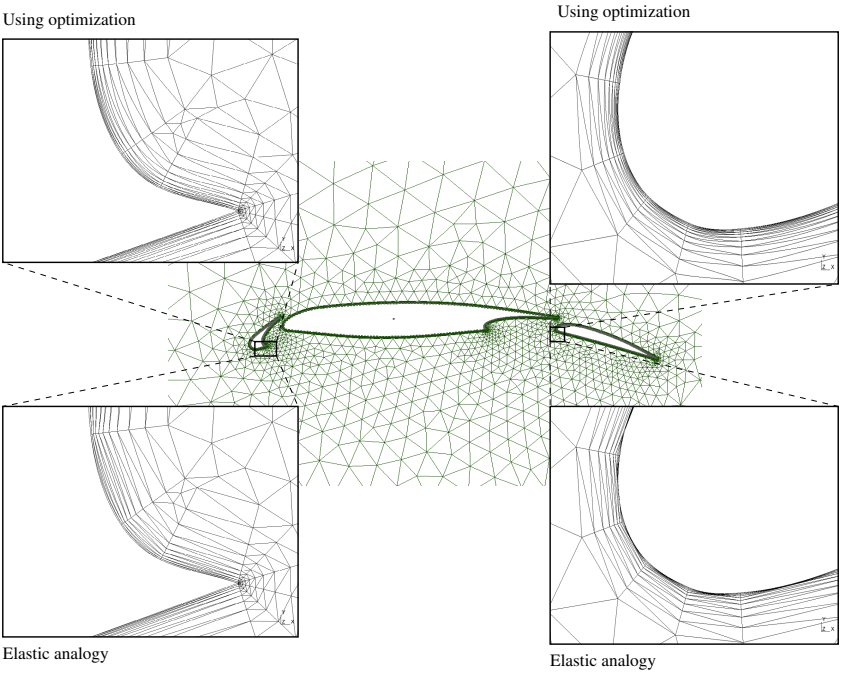

Fig. 6. Quadratic boundary layer mesh of a three component wing configuration

5.3 seconds. The worst jacobian is in both quadratic and cubic cases equal to 0.3. A comparison between linear elastic analogy and the present approach for the quadratic mesh. A simple elastic analogy approach is known to fail in providing a valid mesh in presence of highly stretched boundary layer elements [13].

As a first 3D example, Figure 7 shows a quadratic mesh of a mechanical part. The initial linear mesh is composed of 59,760 quadratic tetrahedra among which 1,566 were invalid. The untangling procedure has been applied to 14 separated blobs. About 132 seconds were necessary to optimize the mesh.

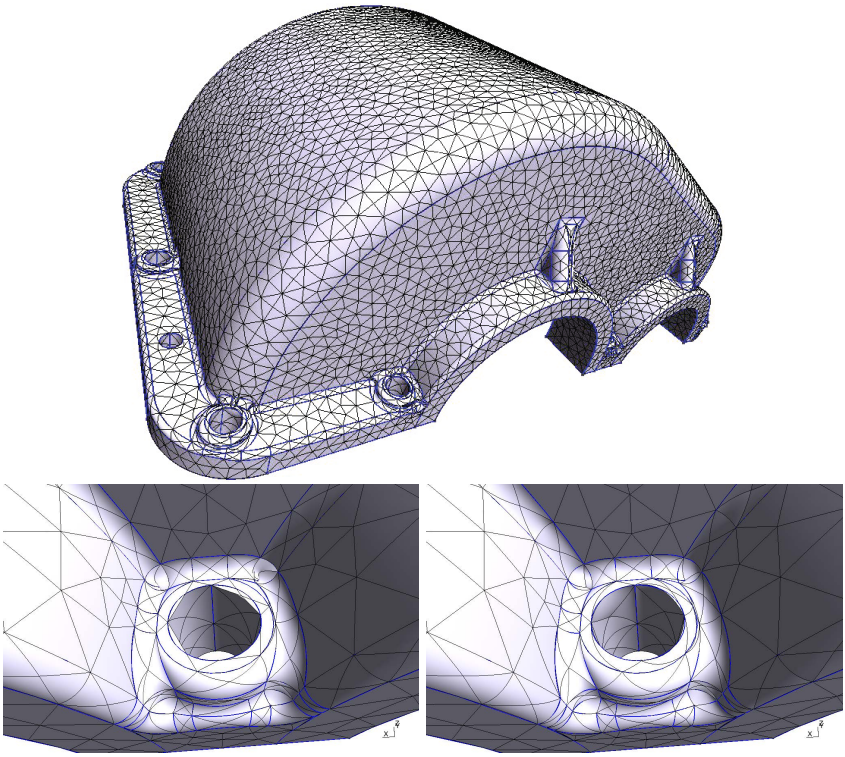

Fig. 7. Untangling of a quadratic mesh

Figure 8 shows a coarse cubic mesh of another mechanical part. The optimization procedure has been applied to one single blob and 60 seconds were necessary to untangle the mesh.

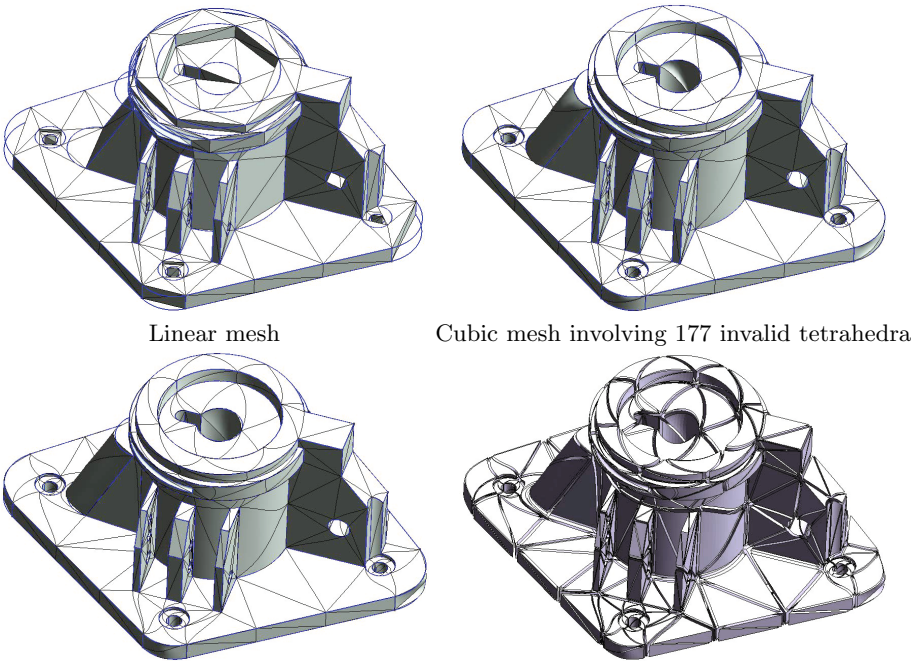

Fig. 8. Untangling of a coarse cubic mesh

7 Conclusions

A new procedure for untangling high order meshes is presented. It allows to efficiently generate high order meshes both in 2D and in 3D. The method relies on robust estimates of element jacobians. In further work, the influence of the jacoboians on finite element solution will be explored.

References

1. Cockburn, B., Karniadakis, G., Shu, C.-W. (eds.): Discontinuous Galerkin Methods. Lecture Notes in Computational Science and Engineering, vol. 11. Springer, Berlin (2000)
2. Kroll, N., Bieler, H., Deconinck, H., Couaillier, V., Van Der Ven, H., Sorensen, K.: 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)
3. Bassi, F., Rebay, S.: High-order accurate discontinuous finite element solution of the 2D Euler equations. J. Comput. Phys. 138(2), 251–285 (1997), doi:10.1006/jcph.1997.5454

4. Bernard, P.-E., Remacle, J.-F., Legat, V.: Boundary discretization for high-order discontinuous galerkin computations of tidal flows around shallow water islands. International Journal for Numerical Methods in Fluids 59(5), 535–557 (2009), doi:10.1002/fld.1831
5. Toulorge, T., Desmet, W.: Curved boundary treatments for the discontinuous galerkin method applied to aeroacoustic propagation. AIAA J. 48(2), 479–489 (2010), doi:10.2514/1.45353
6. Remacle, J.-F., Shephard, M.S.: An algorithm oriented mesh database. International Journal for Numerical Methods in Engineering 58(2), 349–374 (2003), doi:10.1002/nme.774
7. Johnen, A., Remacle, J.-F., Geuzaine, C.: Geometrical validity of curvilinear finite elements. In: Quadros, W.R. (ed.) Proceedings of the 20th International Meshing Roundtable, vol. 90, pp. 255–271. Springer, Heidelberg (2011), doi:10.1007/978-3-642-24734-7_14
8. George, P.L., Borouchaki, H.: Construction de maillages de degré 2 partie 3: Tétraèdre p2, Tech. rep., INRIA, France (2011)
9. Knupp, P.M.: Winslow smoothing on two-dimensional unstructured meshes. Engineering with Computers 15, 263–268 (1999), doi:10.1007/s003660050021
10. Li, X., Shephard, M.S., Beall, M.W.: Accounting for curved domains in mesh adaptation. International Journal for Numerical Methods in Engineering 58(2), 247–276 (2003)
11. Luo, X.J., Shephard, M.S., O'Bara, R.M., Nastasia, R., Beall, M.W.: Automatic p-version mesh generation for curved domains. Engineering with Computers 20(3), 273–285 (2004)
12. Shephard, M.S., Flaherty, J.E., Jansen, K.E., Li, X., Luo, X., Chevaugeon, N., Remacle, J.F., Beall, M.W., O'Bara, R.M.: Adaptive mesh generation for curved domains. Applied Numerical Mathematics 52(2), 251–271 (2005)
13. Persson, P.-O., Peraire, J.: Curved mesh generation and mesh refinement using lagrangian solid mechanics. In: Proceedings of the 47th AIAA Aerospace Sciences Meeting and Exhibit, Orlando (FL), USA, January 5-9 (2009)
14. Freitag, L., Knupp, P., Munson, T., Shontz, S.: A comparison of optimization software for mesh shape-quality improvement problems., Tech. rep., Argonne National Lab., IL, US (2002)
15. Fiacco, A.V., McCormick, G.P.: Nonlinear programming: sequential unconstrained minimization techniques, vol. 4. Society for Industrial Mathematics (1990)
16. Waechter, A., Laird, C., Margot, F., Kawajir, Y.: Introduction to ipopt: A tutorial for downloading, installing, and using ipopt (2009)
17. Liu, D., Nocedal, J.: On the limited memory bfgs method for large scale optimization. Mathematical Programming 45(1), 503–528 (1989)
18. Fletcher, R., Reeves, C.M.: Function minimization by conjugate gradients. The Computer Journal 7(2), 149–154 (1964)