# Connectivity-change moving mesh methods for high-order meshes: Toward closed advancing-layer high-order boundary layer mesh generation

Rémi Feuillet, Adrien Loseille, David Marcum, Frédéric Alauzet

# Connectivity-change moving mesh methods for high-order meshes: Toward closed advancing-layer high-order boundary layer mesh generation

Rémi Feuillet[1,*], Adrien Loseille[1,†], David Marcum[2,‡], Frédéric Alauzet[1,§]

[1] *GAMMA3 Team, INRIA Saclay Ile-de-France, Palaiseau, France*

[2] *Center for Advanced Vehicular Systems (CAVS), Mississippi State University, MS 39762, USA*

**Curved mesh generation starting from a $P^1$ mesh and closed advancing-layer boundary layer mesh generation both rely on mesh deformation and mesh optimization techniques. The approach presented in this work is to generalize connectivity-change moving mesh methods to high-order meshes. This approach is based on a high-order linear elasticity solver for the mesh deformation and on high-order mesh optimization operators such as mesh smoothing and generalized swapping. Thanks to this method, $P^k$ meshes are generated from $P^1$ meshes and closed advancing-layer boundary layer mesh generation will soon be possible.**

## I. Introduction

In numerical simulation, unstructured meshes are massively used. More specifically, in Computational Fluid Dynamics (CFD) they are used to help solve industrial and governmental problems. In CFD, regions near vehicles, aircrafts or surfaces of study known as viscous boundary-layer regions provide highly relevant physical phenomena like turbulence that must be captured by the numerical simulation. In these regions, optimal meshes are generally pseudo-structured and highly aligned with the boundary.[3]
Moreover, in the last decade high-order resolution methods (continuous Galerkin,[7,8] discontinuous Galerkin,[16] spectral differences,[30] k-exact,[14] ...) are more and more used. To preserve the high-order of convergence of these methods, it is required to have a high-order representation of the geometry in the mesh.[18] These meshes are curved in order to fit at the best with the boundary of the studied geometric shape. In this context, the generation of both high-order meshes and high-order curved boundary layer meshes is necessary.

To generate high-order meshes, several approaches exist. Some are using a PDE or variational approach to curve a $P^1$ mesh into a $P^k$ mesh,[1,15,22,23] others are based on optimization and smoothing operations and start from a $P^1$ mesh with a constrained $P^k$ curved boundary in order to generate a suitable $P^k$ mesh.[17,24,27] In all these approaches, the key feature is to find the best deformation to apply to the initial $P^1$ mesh.

To generate boundary layer meshes,[5,21,25] two main methodologies exist. The open-type advancing layer method where a boundary surface mesh is the starting point and it is inflated or advanced one or more layers at a time and the closed advancing-layer method which starts from an existing volume mesh. The chosen method[3] is a closed-advancing layer method. This method starts from the boundary and inflates the boundary by creating a layer or a group of layers. In response to this inflation the volume mesh is deformed and displaced. The key component of this method is therefore the displacement of the mesh. The treatment of the motion of the mesh can be done in several ways.[5,21,26] Here, the motion of the mesh is ensured by a connectivity-change moving mesh algorithm based on a method using a linear-elasticity analogy and mesh optimization[2] (smoothing and swapping). The final aim of this work is to extend this closed-advancing layer

*Ph.D. Student, remi.feuillet@inria.fr
†Researcher, adrien.loseille@inria.fr
‡Professor, marcum@cavs.msstate.edu
§Senior Researcher, frederic.alauzet@inria.fr

method to high-order curved boundary layers mesh generation.

The high-order mesh deformation algorithm appears to be the core component of the two previous high-order mesh generation algorithms. In this work, a connectivity-change moving mesh methods for high-order meshes is presented. It is based on a high-order resolution of the linear elasticity equation in which all the degrees of freedom of the problem are intrinsically represented. This gives us the motion of the vertices. Then it uses local mesh optimization operators such as mesh smoothing (for vertices and nodes) and generalized swapping coupled with an optimization of the position of the nodes inside the swap cavity. All this work can be first applied to the generation of a $P^k$ mesh starting from a $P^1$ mesh and then to the generation of closed-advancing high-order boundary layer meshes.

The paper is outlined as follow. Section II defines what a high-order mesh and sets up validity and quality criteria. Section III deals with high-order mesh optimization. Section IV presents the high-order Finite Element linear elasticity solver. Section V shows a $P^k$ mesh generation technique by curving a $P^1$ mesh based on the previous solver. Section VI describes the connectivity-change moving mesh method on high order meshes based on the same solver. High-order meshes can be generated by the method of section V. Finally, section VII deals with perspectives driven by this work and in particular closed advancing-layer high-order boundary layer mesh generation.

## II. Definition of a high-order element, validity and quality criteria

In general, a finite element is defined[6] by the triplet $\{K, \Sigma_K, V_h\}$ where K denotes the geometric element (triangle, etc), $\Sigma_K$ the list of nodes of $K$, and $V_h$, the space of the *shape functions*, here it will be the Lagrange polynomial functions (or interpolants). To properly define the geometry and these functions, a reference space (that can also be seen as a parameter space) is defined where all coordinates are between 0 and 1. In this space, the reference element is denoted $\widehat{K}$ and has a fixed (and sometimes uniform) distribution of nodes. The element $K$, also called physical element, is thus the image of $\widehat{K}$ via a mapping, denoted $F_K$ (see Figure 1). More specifically, for each point $M$ of $K$, there is a point $\widehat{M}$ of $\widehat{K}$ such that $M = F_K(\widehat{M})$.

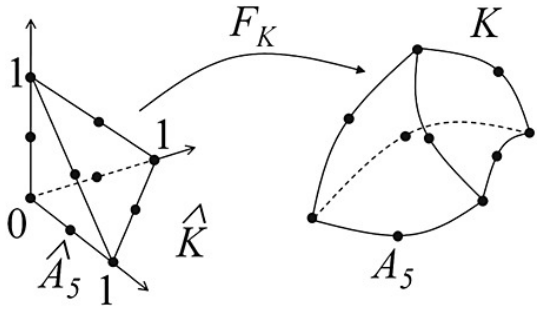


**Figure 1. Mapping $F_K$ from $\widehat{K}$ to $K$.**

Finally, the position of a point M inside K is defined by

$$M = \sum_{i=1}^{n} \phi_i(\widehat{M}) A_i,$$

where n is the number of nodes, $A_i = F_K(\widehat{A}_i)$ with $\widehat{A}_i$ the nodes of the reference element which map to $A_i$ the nodes of the physical element, and $\phi_i$ are the Lagrange polynomial functions defined such that:

$$\phi_i(\widehat{A}_j) = \delta_{ij} \quad \text{and} \quad \sum_{i=1}^{n} \phi_i = 1.$$

It is important to note that in order to define a complete finite element of degree $k$ on a simplex of dimension $d$ (edge for $d = 1$, triangle for $d = 2$, tetrahedron for $d = 3$), the number of (distinct) nodes needs to be

equal to $\frac{\Pi_{j=1}^{d}(k+j)}{d!}$. Also, on a simplex, the reference coordinates $(\widehat{x},\widehat{y},\widehat{z})$ can be used to define the simplex barycentric coordinates $(u,v,w,t)$. For instance, for a triangle we have: $u=1-\widehat{x}-\widehat{y}$, $v=\widehat{x}$ and $w=\widehat{y}$, and for a tetrahedron $u=1-\widehat{x}-\widehat{y}-\widehat{z}$, $v=\widehat{x}$, $w=\widehat{y}$ and $t=\widehat{z}$.
A point M of the simplex K can also be expressed in Bézier form using the Bernstein polynomials $B_{ijlm}^{d}$ as:

$$M=\sum_{i+j+l+m=k} B_{ijlm}^{d}(u,v,w,t)P_{ijlm},$$

with $B_{ijlm}^{d}(u,v,w,t)=\frac{d!}{i!j!l!m!}u^i v^j w^l t^m$. For a triangle, $m=0$. The points $(P_{ijlm})_{i+j+l+m=k}$, also called $(C_i)_{1\leq i\leq n}$ (see Figure 2) are the Bézier control points and are directly related to the nodes $(A_i)_{1\leq i\leq n}$.
For instance, to compute $C_5$ ($P_{1100}$) in Figure 2, we use the formula:

$$C_5=\frac{4A_5-A_1-A_2}{2}.$$

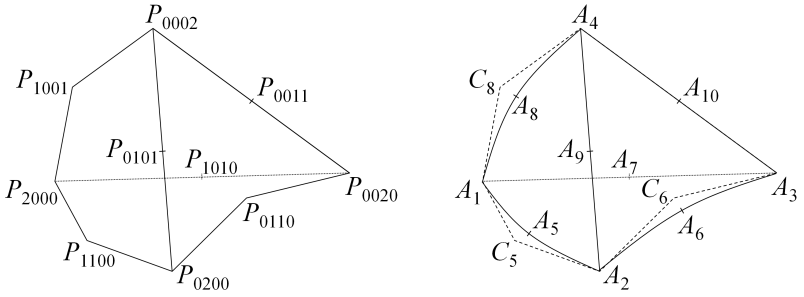


**Figure 2. Correspondence between the control points and the nodes on a $P^2$ tetrahedron.**

The validity of an element means that the associated mapping $F_K$ is a diffeomorphism. It can be ensured if the minimum of the determinant $\mathcal{J}_K$ of the jacobian matrix of the mapping $F_K$ is strictly positive everywhere inside the element.[10,11] In the case of a simplicial element, jacobian $\mathcal{J}_K$ can be written as polynomial of degree $d\times(k-1)$ in the barycentric coordinates of the simplex, where $d$ is the dimension of the simplex and $k$ the degree of the simplex. When the element is of degree 1 (e.g. straight-sided), it simply means that the oriented volume/area is strictly positive. The jacobian can also be expressed in the Bernstein polynomial basis:

$$\mathcal{J}_K(u,v,w,t)=\sum_{i+j+l+m=d(k-1)} B_{ijlm}^{d(k-1)}(u,v,w,t)N_{ijlm}^{K},$$

where $N_{ijlm}^{K}$ are the control coefficient of the jacobian. These coefficients can be explicitly found and have a geometrical meaning. In the case of a $P^2$-triangle (see Figure 3), the 6 control coefficients are:

$$N_{200}^{K}=4\det(\overrightarrow{P_{200}P_{110}},\overrightarrow{P_{200}P_{101}}), \tag{1}$$
$$N_{110}^{K}=2\det(\overrightarrow{P_{200}P_{110}},\overrightarrow{P_{110}P_{011}})+2\det(\overrightarrow{P_{110}P_{020}},\overrightarrow{P_{200}P_{101}}), \tag{2}$$
$$N_{020}^{K}=4\det(\overrightarrow{P_{110}P_{020}},\overrightarrow{P_{110}P_{011}}), \tag{3}$$
$$N_{011}^{K}=2\det(\overrightarrow{P_{110}P_{020}},\overrightarrow{P_{101}P_{002}})+2\det(\overrightarrow{P_{101}P_{011}},\overrightarrow{P_{110}P_{011}}), \tag{4}$$
$$N_{002}^{K}=4\det(\overrightarrow{P_{101}P_{011}},\overrightarrow{P_{101}P_{002}}), \tag{5}$$
$$N_{101}^{K}=2\det(\overrightarrow{P_{200}P_{110}},\overrightarrow{P_{101}P_{002}})+2\det(\overrightarrow{P_{101}P_{011}},\overrightarrow{P_{200}P_{101}}). \tag{6}$$

Consequently, a sufficient condition to prove that $\mathcal{J}_K$ is strictly positive everywhere is to ensure that all $N_{ijlm}^{K}$ are strictly positive, but it is a too strong condition. On the contrary, if a $N_{ijlm}^{K}$ is negative in a corner, it means that $\mathcal{J}_K$ is negative somewhere inside the element. However, if a control coefficient lying on edges faces and volumes is negative, it does not mean that $\mathcal{J}_K$ is negative somewhere inside the element.

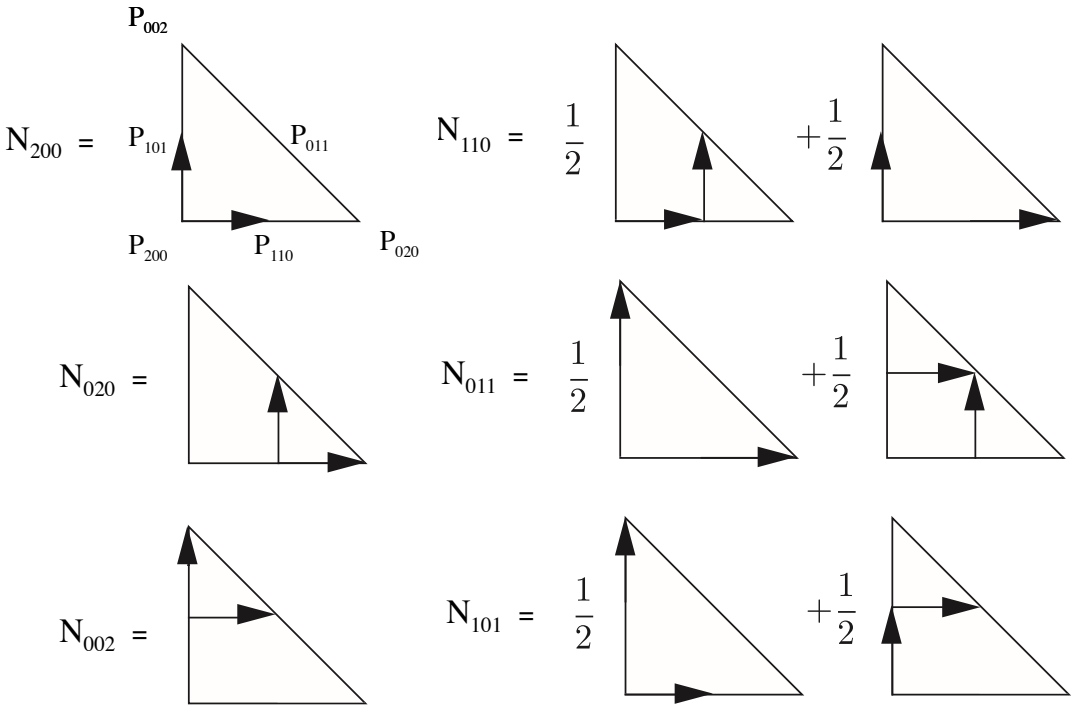


**Figure 3. Vectors involved in the determinant for the computation of all the control coefficients of a $P^2$ triangle.**

We cannot conclude on the positiveness of the jacobian without any further analysis. In this case, a few iterations of a De Casteljau's algorithm[2,10] are required to have more accurate bounds of the jacobian.
Once the validity of the element is established, it is interesting to set up a quality criterion[12] for the shape of the element:

$$Q = \alpha \frac{hS_k}{V_k} \frac{\max(V_1, V_k)}{\min(V_1, V_k)} \left( \frac{N^K_{max}}{N^K_{min}} \right)^{1/d},$$

with $S_k$ the exterior surface of the polyhedron defined by the nodes (or the half perimeter defined by the polygon of nodes in 2D), $V_k$ the volume of the polyhedron (resp. surface of the polygon), $h$ the element's largest edge $P^k$-length (e.g the length of the union of straight-sided lines defined by the nodes), $V_1$ the volume/surface of the equivalent $P^1$ element, e.g the element without inner nodes and finally $N^K_{min}$ (resp. $N^K_{max}$) the smallest (resp. largest) control coefficient of the jacobian of the element, and $\alpha$ is a normalization factor, dependent of the dimension such that $Q = 1$ for a regular simplex.
This quality function is actually a product of 3 terms. The first one is only a generalization of the $P^1$ quality function and measures the gap to the regular element, the second one measures the distance between the volume of curved element and the volume of the straight element, and finally the third one gives us a measure of the distortion of the element and can detect if the element is false or almost. Let us note that if the element is straight, the standard $P^1$ quality function is found.

## III. High-order mesh optimization

In the same way as we want to have an optimal $P^1$ mesh in terms of quality, we want to have an optimal $P^k$ mesh. Several optimization techniques exist to correct a false $P^k$ mesh[1,17,27] and to optimize the geometrical accuracy.[28]
The idea here is to extend two classic mesh optimization operators[2] to high-order meshes to increase its quality.
In this section and in the following sections, the nodes defining the simplex will be denoted as vertices

whereas the other nodes, that define high-order entities, will still be denoted as nodes or inner nodes.

### A. Swap operator

The swap operator (see figure 4) locally changes the connectivity of the mesh in order to improve its quality. In 2D, it consists in flipping of an edge shared by 2 triangles to form two new triangles with the same 4 vertices.
In 3D, two types of swap exist: face and edge swapping. The face swapping is the extension of the 2D edge swapping, it consists in replacing the common face of two neighboring tetrahedra by the edge linking the opposite vertices to the face of each tetrahedron, also called $2 \rightarrow 3$. The edge swapping is a bit different. First, the shell of the edge to delete (e.g. the set of elements containing this edge) is constructed. From a shell of size $n$, a non-planar pseudo-polygon formed by $n$ vertices is obtained. The swap consists in deleting the edge, generating a triangulation of the polygon and creating two tetrahedra for each triangle of the triangulation thanks to the two extremities of the former edge. These swaps are designated as $n \rightarrow m$ with $n \geq 3$, the initial number of tetrahedra and $m = 2(n-2)$ the final number of tetrahedra.

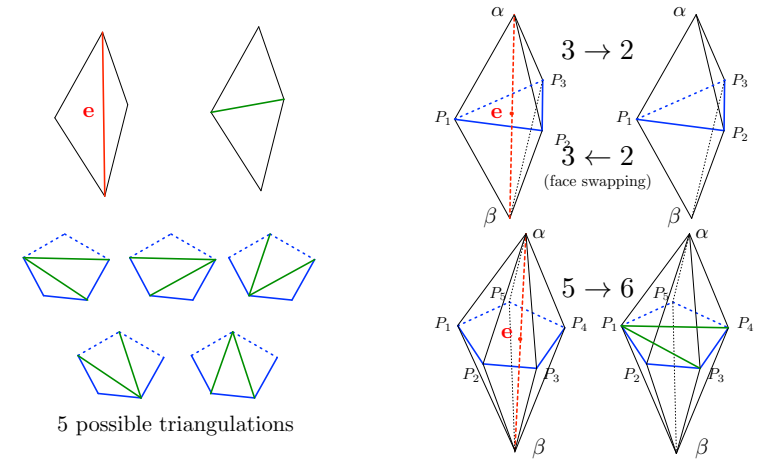


**Figure 4.** ***Top left*** **the swap operation in 2D.** ***Top right*** **edge swap $3 \rightarrow 2$ and face swap $2 \rightarrow 3$.** ***Bottom left*** **the five possible triangulations of the pseudo polygon for a shell having five elements.** ***Bottom right*** **an example of $5 \rightarrow 6$ edge swap. For all these figures, shells are in *black*, old edges are in red, new edges are in *green*.**

For each possible swapped configuration, if the worst quality of all the elements the shell is improved, the configuration is kept and will be in the new mesh unless another swapped configuration of the shell provides a better quality improvement. Sometimes, a small local degradation of the shell's worst quality is required to improve the global quality of the mesh.
To generalize it to high-order meshes, the nodes of the edges of the shell in $P^2$ have to be taken into account. For instance, for the $P^2$ case in 2D, there is one node on the swapped edge and if we want the swap to be performed, we need first to find an optimal position for the node in the swapped configuration and then to check if this configuration improves the quality function (see Figure 5). The key feature is therefore to find a functional whose optimum will give the optimal position for the node in the swapped configuration in term of quality.

The idea here in $P^2$ was to find a simple and smooth functional that will be easy to optimize. It appeared quickly that the quality function was not a good candidate as it is not smooth and that is why the following functional (inspired from the work of[27]) was proposed:

$$f_{2D}(\mathbf{x}) = \sum_{K \in \mathcal{S}(e)} \sum_{i+j+k=2} \omega_{ijk} \left( \frac{N^K_{ijk}(\mathbf{x})}{2|K_s|} - 1 \right)^2,$$

where $\mathbf{x}$ is the coordinates of the node of the edge $e$ to optimize, $\mathcal{S}(e)$ the shell of the edge $e$ (e.g. the set of elements $K$ containing $e$) $\omega_{ijk}$ is a weight function equal to 2 for the corner coefficients and equal to 1

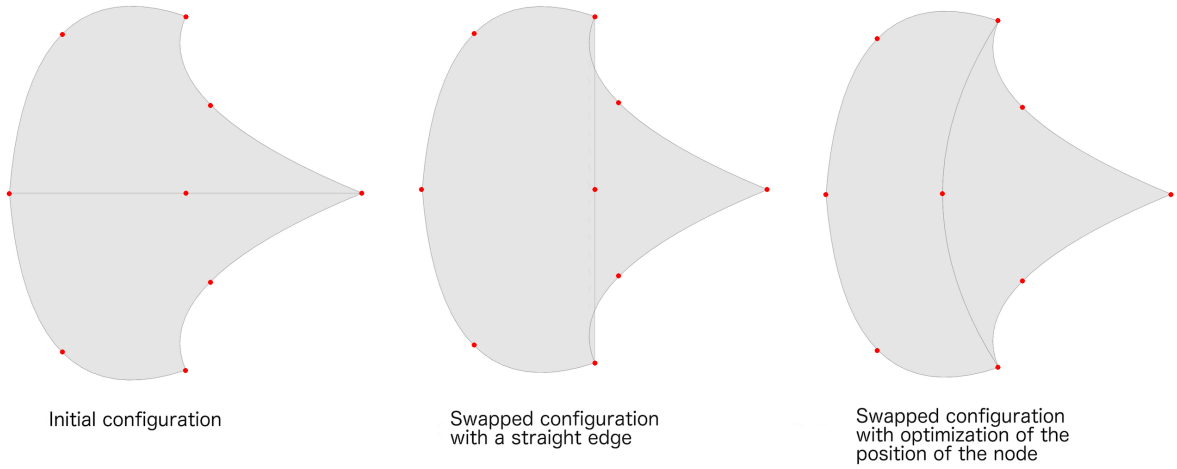


**Figure 5. Three steps of $P^2$ swap in 2D.**

everywhere that measures the importance of the control coefficients and $|K_s|$ is the volume of the straight-sided element $K_s$ deduced from $K$. $f_{2D}$ has the following properties : it is a positive define quadratic form as $\mathbf{x} \rightarrow N_{ijk}^K(\mathbf{x})$ is linear in $\mathbf{x}$, which means that the functional as a unique minimum that can be easily computed thanks to a L-BFGS algorithm.[19] Also, on every regular swap configuration, the minimum of $f_{2D}$ in $\mathbf{x}$ is the same as the minimum of the worst quality of the swapped shell in $\mathbf{x}$. Using the result of the optimization problem in the swap configuration gives therefore a very good approximation of the best configuration that can be obtained. Since the best swap configuration is found, we are able to conclude if this swap will increase the quality or not.
This process can be generalized in 3D by using the following functional:

$$f_{3D}(\mathbf{X}) = \sum_{K \in \mathcal{S}(e)} \sum_{i+j+k+l=3} \omega_{ijkl} \left( \frac{N_{ijkl}^K(\mathbf{X})}{6|K_s|} - 1 \right)^2,$$

where $\mathbf{X} = [\mathbf{x_1}, ..., \mathbf{x_n}]$ is a *set* of $n$ coordinates to be optimized. It can either be empty (in which case no optimization is required), or contain more than one node's coordinates. $\mathbf{X} \rightarrow N_{ijk}^K(\mathbf{X})$ is a function that depends in the worst case of two variables of $\mathbf{X}$. The weights $\omega_{ijkl}$ are equal to 4 for a corner control coefficient, equal to 2 for an edge control coefficient and equal to 1 otherwise. In this work, considered swaps are $2 \rightarrow 3$, $3 \rightarrow 2$ and $4 \rightarrow 4$ which means that the functional of the problem is in the worst case a positive define quadratic form. For the other swaps ($5 \rightarrow 6$, $6 \rightarrow 8$), the problem begins to be highly costly in term of CPU.

## B. Mesh smoothing

Mesh smoothing is a technique that consists in relocating some points inside the mesh. In $P^1$, the idea is to relocate each point $P_i$ inside its ball of elements (see figure 6). For each element $K_j$ of the opposite face to $P_i$ denoted by $F_j$ gives an optimal position $P_j^{opt}$ and the vertex is relocated to a weighted average of the proposed positions. If the final position is a less optimal configuration in term of quality, relaxation is performed until the configuration becomes more optimal in term of quality. The optimal configuration is computed as follows:

$$P_j^{opt} = G_j + \sqrt{\frac{2}{3}} h_j \frac{\mathbf{n}_j}{||\mathbf{n}_j||},$$

where $G_j$ is the gravity center, of the face, $h_j$ the average length of the edges of $F_j$, and $\mathbf{n}_j$, the outward normal to $F_j$. The proposed position is then computed with:

$$P_i^{opt} = \frac{\sum_{K_j \in Ball(P_i)} \max(Q(K_j), Q_{\max}) P_j^{opt}}{\sum_{K_j \in Ball(P_i)} \max(Q(K_j), Q_{\max})},$$

where $Q_{\max}$ is a parameter to be defined. Here $Q_{\max} = 10$

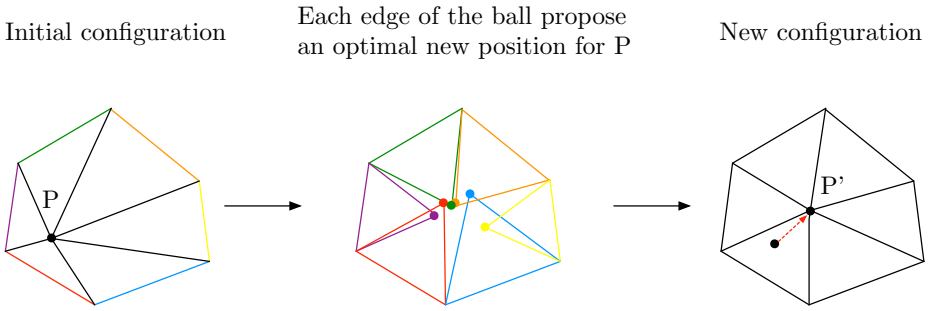


**Figure 6. Laplacian smoothing in two dimensions. Each element of the ball of considered vertex $P_i$ suggests an optimal position for $P_i$. The resulting new optimal position for $P_i$ is computed as a weighted average of all these proposed locations.**

To extend it to $P^2$ meshes, the edge's node needs to be taken into account. The idea here, is to perform two independent smoothing operations:

- A vertex smoothing.
- A node smoothing.

The vertex smoothing can be seen as a generalization of the $P^1$ smoothing. The optimal position of the vertex is computed in the same way as in $P^1$, and the vertex is located exactly in the same way as before. In order to keep straight the edges of a ball with initial straight edges, the displacement of all the inner nodes of the ball cavity is, using Thales theorem, set to half of the value of the displacement of the central vertex. The optimization of the node position is close to the $P^2$ swap-operator. The idea is in $P^2$ to optimize the position of one node in a shell. For this purpose, functionals $f_{2D}$ and $f_{3D}$ can be re-used to find the optimal node position. In this case, there is always only one node coordinates to optimize and consequently the problem is quadratic.

## IV. High-order Finite Element resolution of linear elasticity equation

In this section, the resolution of linear elasticity equation with a $P^k$ Finite Element Method (FEM) is presented. Let us consider the linear elasticity equations:

$$\nabla \cdot (\sigma(\mathcal{E})) = -\mathbf{f}\,, \quad \text{with} \quad \mathcal{E} = \frac{\nabla\xi + {}^T\nabla\xi}{2}\,, \tag{7}$$

where $\sigma$, and $\mathcal{E}$ are respectively the Cauchy stress and strain tensors, $\xi$ is the Lagrangian displacement and $\mathbf{f}$ are the body forces.
The Cauchy stress tensor follows the Hooke's law for isotropic homogeneous medium, where $\nu$ is the Poisson ratio, $E$ the Young modulus of the material, and $\lambda$ and $\mu$ are the Lamé coefficients:

$$\sigma(\mathcal{E}) = \lambda \mathrm{Tr}(\mathcal{E})\mathrm{I} + 2\mu\mathcal{E} \quad \text{or} \quad \mathcal{E}(\sigma) = \frac{1+\nu}{E}\sigma - \frac{\nu}{E}\mathrm{Tr}(\sigma)\mathrm{I}.$$

Dirichlet boundary conditions are used to enforce the displacement on the boundary. For symmetry planes or imprint surfaces, wall-slip boundary conditions are used to enforce a displacement in the tangent plane.

After a standard mathematical analysis using a continuous Galerkin approximation of the variational form associated to Problem (7) on a mesh $\mathcal{H}$, the Finite Element Method leads us to the following linear system:

$$\mathbb{K}\Xi = F.$$

where $\mathbb{K}$ is the linear elasticity stiffness matrix with blocks of size $d \times d$ and $F$ is a vector with blocks of size $d$, where $d$ is the dimension, and are defined (for $d = 2$) by :

$$\mathbb{K}_{IJ} = \begin{pmatrix} \sum\limits_{K\ni(P_I,P_J)} \int_K (\lambda+2\mu)\frac{\partial\phi_J}{\partial x}\frac{\partial\phi_I}{\partial x} + \mu\frac{\partial\phi_J}{\partial y}\frac{\partial\phi_I}{\partial y}\mathrm{d}\Omega & \sum\limits_{K\ni(P_I,P_J)} \int_K \lambda\frac{\partial\phi_J}{\partial y}\frac{\partial\phi_I}{\partial x} + \mu\frac{\partial\phi_J}{\partial x}\frac{\partial\phi_I}{\partial y}\mathrm{d}\Omega \\ \sum\limits_{K\ni(P_I,P_J)} \int_K \mu\frac{\partial\phi_J}{\partial y}\frac{\partial\phi_I}{\partial x} + \lambda\frac{\partial\phi_J}{\partial x}\frac{\partial\phi_I}{\partial y}\mathrm{d}\Omega & \sum\limits_{K\ni(P_I,P_J)} \int_K \mu\frac{\partial\phi_J}{\partial x}\frac{\partial\phi_I}{\partial x} + (\lambda+2\mu)\frac{\partial\phi_J}{\partial y}\frac{\partial\phi_I}{\partial y}\mathrm{d}\Omega \end{pmatrix},$$

$$F_I = \begin{pmatrix} \sum\limits_J f_{J,1} \sum\limits_{K\ni(P_I,P_J)} \int_K \phi_J\phi_I\mathrm{d}\Omega \\ \sum\limits_J f_{J,2} \sum\limits_{K\ni(P_I,P_J)} \int_K \phi_J\phi_I\mathrm{d}\Omega \end{pmatrix}.$$

$K$ are the elements of the mesh (edges, triangles, tetrahedra) and $P_I$, $P_J$ are two nodes/vertices of the mesh. The solution vector $\Xi$ is defined by blocks (for $d = 2$) as :

$$\Xi = \begin{pmatrix} \xi_{I,1} \\ \xi_{I,2} \end{pmatrix},$$

where $\xi_{I,j}$ (resp. $f_{I,j}$) are referring to the $j-th$ coordinates of the evaluation of $\xi$ (resp. $\mathbf{f}$) at $P_I$.
The $\phi_I$ are the $P^k$ finite element shape functions associated with the nodes/vertices $P_I$. In the case of the Lagrangian interpolation in a $d-$simplex , these functions restricted to a simplex are a polynomial combination of the $d+1$ elementary barycentric functions of this simplex. More precisely, in $P^1$, they are exactly these barycentric functions.
The key feature in this method is to compute the integrals

$$\int_K \frac{\partial\phi_J}{\partial x_l}\frac{\partial\phi_I}{\partial x_m}\mathrm{d}\Omega \quad \text{and} \quad \int_K \phi_J\phi_I\mathrm{d}\Omega.$$

In $P^1$, the exact formulas are known thanks to these formulas[7,8] :

$$\int_{K_d} \prod_{i=1}^{d+1} \lambda_i^{\beta_i}\mathrm{d}\Omega = \frac{\prod\limits_{i=1}^{d+1}\beta_i!}{\left(d+\sum\limits_{i=1}^{d+1}\beta_i\right)!} d!|K_d|, \quad \forall(\beta_i)\in\mathbb{N}^{d+1} \text{ with } i\in[\![1,..,d+1]\!], \tag{8}$$

$$\nabla_{\mathbf{X}}\lambda_i = \frac{1}{d!|K_d|}\mathbf{n}_i, \tag{9}$$

where $K_d$ is the simplex of dimension $d$, $\lambda_i$ its associated barycentric functions and $\mathbf{n}_i$ its associated normals. For $P^k$ with $k \geq 2$, there are two cases:

- The high-order element $K$ is straight. In other words, $K$ has exactly the same shape as a simplicial element. In this case, it is possible to compute these integrals without any quadrature (and with a lower CPU cost) by remembering that the shape functions and their derivatives are only a polynomial combination of the barycentric functions and their (constant) derivatives. Using the two formulas above, the analytical formula can be found in all the cases.

- The high-order element K is curved. In this case, the previous trick cannot work. The element K is mapped to a straight reference element $\widehat{K}$ on which a Gauss quadrature formula is used. Since a Gauss quadrature of order $n$ is exact for polynomials of degree $2n-1$ or less, if the order of Gauss quadrature is high enough, the exact result can be found as both functions product and mapping are polynomial. Nonetheless, both computation of the jacobian of the mapping and Gauss quadrature are costly in terms of CPU.

Once the linear system is assembled, both $\mathbb{K}$ and $F$ are modified using a *pseudo-elimination* technique in order to take into account Dirichlet and wall slip boundary conditions.
The FE system is then solved by a Conjugate Gradient algorithm coupled with a LU-SGS pre-conditioner.

## V. High-order mesh generation by curving an initial $P^1$ mesh

Most of the techniques to generate an high-order mesh is to start from a $P^1$ mesh and then to curve it, in a way or another, in order to obtain a $P^k$ mesh.[1,9,17,22,23,29] The main reason to use a post-treatment is that all existing $P^1$ mesh generation algorithms can be reused. It would be harder to implement a directly high-order mesh generator. To curve this mesh, the used models are numerous : PDE or variational models,[1,15,22,23] smoothing and/or optimization procedures,[17,24,27] etc ... Our choice here is to use the linear elasticity equation as a model for the motion of the vertices to generate a $P^k$ mesh from a $P^1$ mesh. For this purpose, let us consider Equation (7) with $\mathbf{f} = \mathbf{0}$ and Dirichlet boundary conditions. These Dirichlet boundary conditions represent the gap between the $P^k$-nodes of the straight boundary elements and their position on the *real* boundary. For mesh boundary vertices, the gap is equal to 0.
To compute the gap, two different techniques are used:

- First case, the boundary is analytically known. In this case, the gap is only the difference between a node on the straight boundary element and its projection on the analytical surface (see Fig. 7).
- Second case, the boundary is only known via its $P^1$ discretization. In this case, a cubic reconstruction of the surface is performed to replace the analytical function. The gap is then computed in the same way as in the first case.

The cubic reconstruction[31] (see Fig. 7) relies on the Bézier representation of a curve. Let us give two normals at two vertices A and B, the idea is to find the two Bézier control points P and Q by choosing:

- P as the orthogonal projection of the point X such that $X = \frac{1}{3}A + \frac{2}{3}B$ on the tangent vector/plane associated with the first point.
- Q as the orthogonal projection of the point Y such that $Y = \frac{1}{3}B + \frac{2}{3}A$ on the tangent vector/plane associated with the second point.

The normal at a vertex is computed as the weighted sum of the normals of all the boundary elements containing this vertex. This way a $P^3$ curve is obtained. For the computation of the inner Bézier control points of the $P^3$ triangle, a *Serendip* model[13] is used. The central Bézier control point is computed as a weighted sum of the vertices and the edges control points of the triangle ABC:

$$P^{central} = -\frac{1}{6}(A + B + C) + \frac{1}{4}\sum_{i=1}^{3}(P_1^{edge_i} + P_2^{edge_i}).$$

The gap is then computed by using the projection of the $P^k$ nodes on this $P^3$ reconstruction.

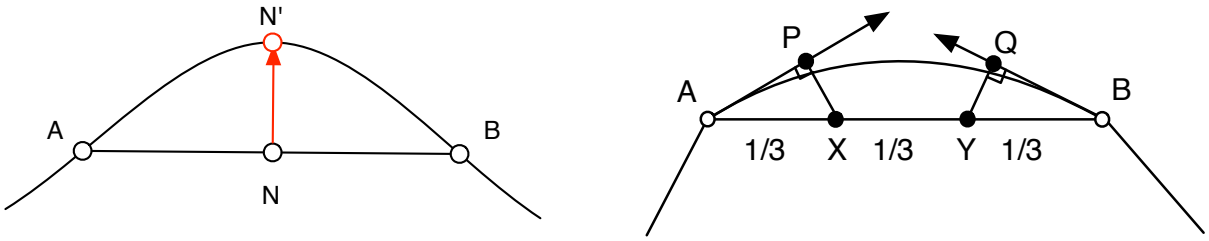


**Figure 7. Left, method to compute the gap between nodes on the straight line and real boundary. Right, the cubic reconstruction technique.**

Once Dirichlet boundary conditions are set, the high-order linear elasticity code is called. As it can be noticed, high-order elements are straight as they are the same elements as the $P^1$ mesh.
The elasticity problem using the high-order FEM provides the new position of the internal vertices and nodes. It is then used to generate the high-order mesh by moving the vertices of the straight mesh with the associated values in the solution vector. With the use of an high-order FE resolution, the degrees of freedom are intrinsically represented which gives more consistency to the obtained motion. The process can be summed up by Algorithm 1.

**Algorithm 1** Mesh curving algorithm

1. Generate a $P^1$ mesh.
2. (Optional) Perform mesh optimization pre-processing
3. Perform cubic reconstruction of the boundary or use its analytical representation to set Dirichlet boundary conditions for the linear elasticity equation.
4. Solve linear elasticity equation with the finite element method at the order of the wanted mesh.
5. Generate the $P^k$ mesh by moving the $P^1$ mesh with the solution of the linear elasticity.
6. (Optional) Perform mesh optimization post-processing
7. Check validity of $P^k$ elements and locally relax the previous solution if necessary or desired.

The major fact with this method is that the deformed mesh is without boundary layer and is consequently only made of isotropic or almost isotropic elements. In this context, the use of the elasticity problem is efficient and always provides a valid mesh. However, this technique does not work well for a mesh with a boundary layer as elements are anisotropic in the boundary layer mesh and that is why it is proposed to directly generate the curved boundary layer by using a closed advancing method. Some results in $P^2$ are presented below. Notice that all figures are obtained by using Vizir.[20]

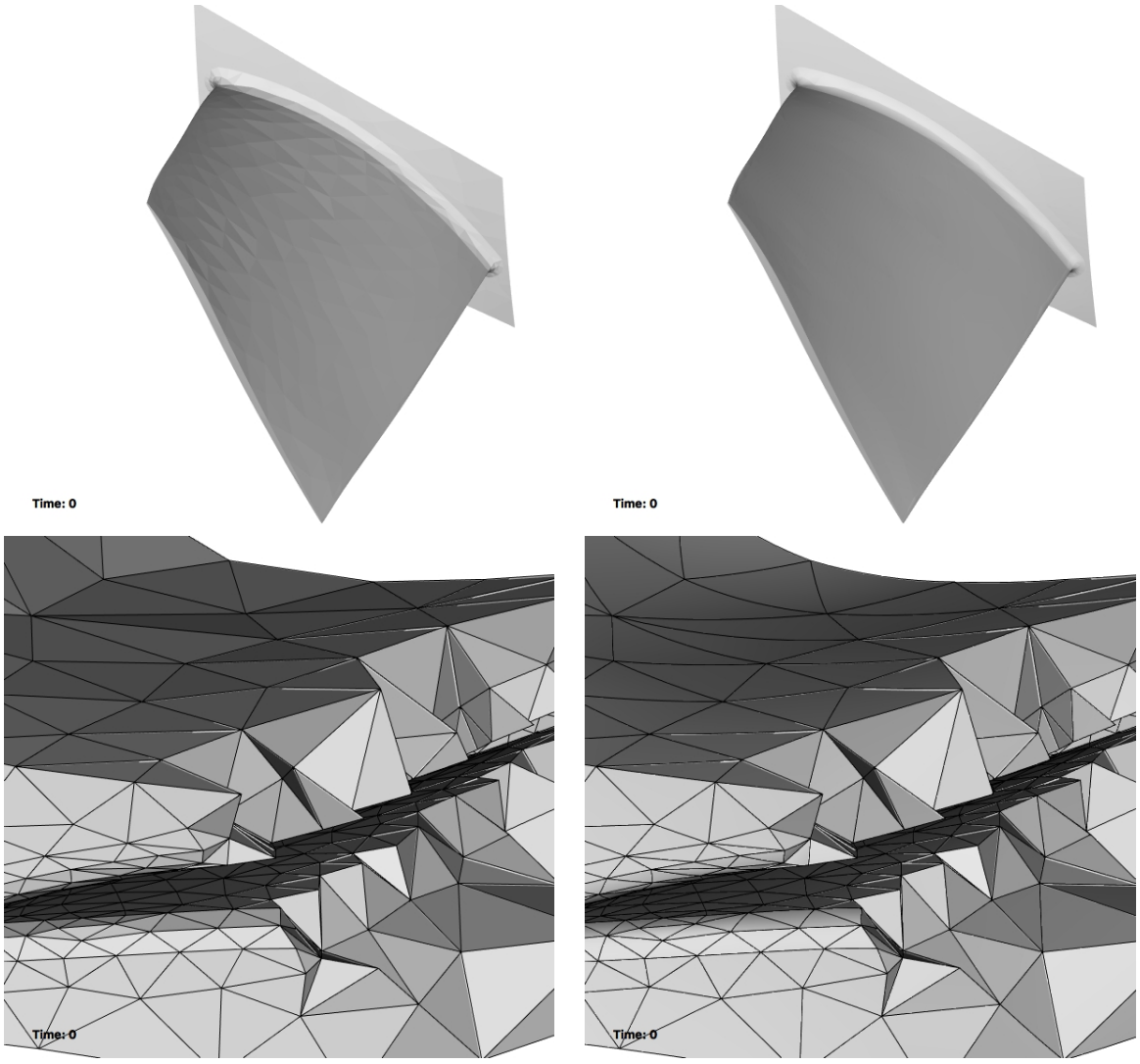


**Figure 8. Left, the initial straight $P^1$ mesh of a NASA RO37 rotor . Right, the $P^2$ mesh generated by the precedent algorithm, with a cubic reconstruction.**

| RO37 mesh | Average quality | Best Q | Worst Q | # vertices | # tets | # invalid |
|---|---|---|---|---|---|---|
| Initial straight mesh | 6.1619 | 1.0082 | 1682.9 | 16985 | 10970 | 0 |
| Mesh curved | 5.3904 | 1.0082 | 1729.1 | 16985 | 10970 | 0 |
| Mesh curved & optimization | 2.5053 | 1.0117 | 1346.5 | 16877 | 10862 | 0 |

**Figure 9. Quality histograms associated to NASA RO37 meshes.**

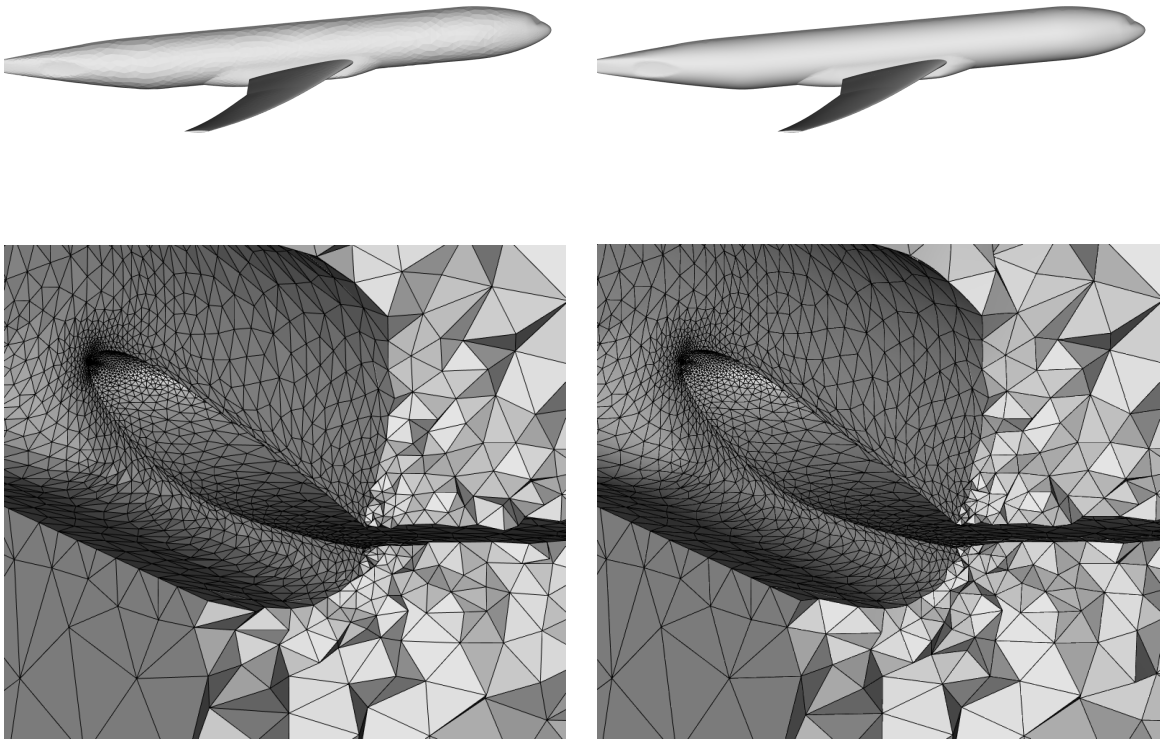

**Figure 10. Left, the initial straight $P^1$ mesh of a high-lift NASA Common Research Model from the high lift prediction workshop. Right, the $P^2$ mesh generated by the precedent algorithm, with a cubic reconstruction.**

| HLCRM mesh | Average Q | Best Q | Worst Q | # vertices | # tets | # invalid |
|---|---|---|---|---|---|---|
| Initial straight mesh | 2.2897 | 1.0003 | 1314 | 761493 | 535672 | 0 |
| Mesh curved | $\infty$ | 1.0003 | $\infty$ | 761493 | 535672 | 4 |
| Mesh curved & optimization | 1.4741 | 1.0003 | 1777.2 | 760967 | 535146 | 0 |

**Figure 11. Quality histograms associated to high-lift NASA Common Research Model meshes.**

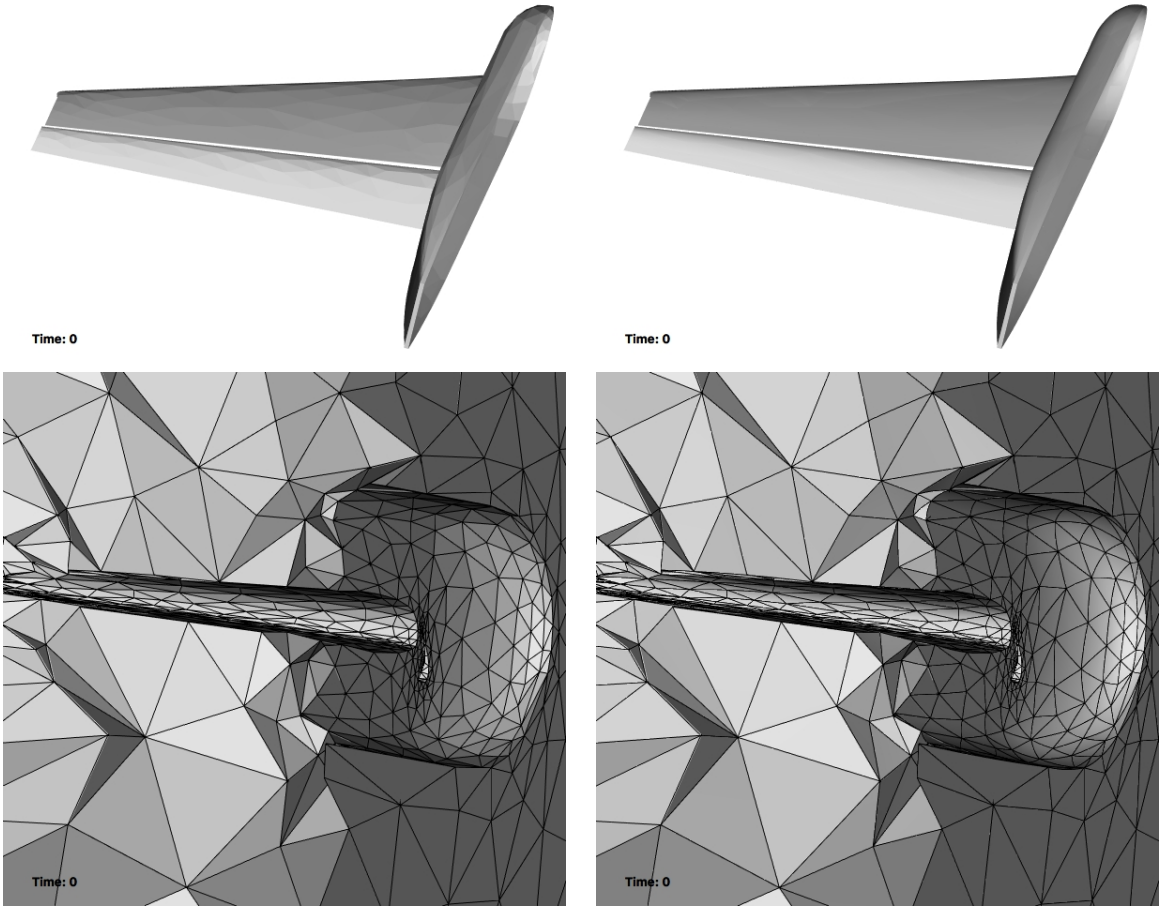


**Figure 12. Left, the initial straight $P^1$ mesh of a high-lift wing body configuration from the 2nd high lift prediction workshop. Right, the $P^2$ mesh generated by the precedent algorithm, with a cubic reconstruction.**

| HL wing body mesh | Average Q | Best Q | Worst Q | # vertices | # tets | # invalid |
|---|---|---|---|---|---|---|
| Initial straight mesh | 18.842 | 1.0111 | 13958 | 37252 | 24794 | 0 |
| Mesh curved | $\infty$ | 1.0107 | $\infty$ | 37252 | 24794 | 1 |
| Mesh curved & optimization | 3.8388 | 1.0045 | 1946.4 | 37185 | 24727 | 0 |

**Figure 13. Quality histograms associated to high-lift wing body configuration meshes.**

It can be noticed that apart from the boundary, the volume elements are almost straight. Optimization in the pre-processing procedure helps the curvature process to be more robust whereas optimization post-processing improves the quality of the final mesh and untangle invalid elements if any.

## VI. A moving mesh technique for high-order elements

The idea behind the high-order moving mesh is to extend the used moving mesh method[2] in $P^k - mesh$ generation from a $P^1$ mesh and in the boundary-layer generation process.
In this case, the initial mesh is a $P^k$-mesh whose boundary has an initial displacement. Using a linear elasticity analogy, the resolution of the elasticity equation with high-order finite elements gives us a displacement for all the vertices and nodes in the volume. Then the mesh is moved to the new position. The motion of the vertices can be also enhanced by using a local stiffness factor technique. The local stiffness factor technique consists by replacing:

$$\int_K \frac{\partial \phi_J}{\partial x_k} \frac{\partial \phi_I}{\partial x_l} \mathrm{d}\mathbf{x},$$

with:

$$\int_K \left( \frac{|\widehat{K}|}{\mathcal{J}_K(\mathbf{x})} \right)^{\chi} \frac{\partial \phi_J}{\partial x_k} \frac{\partial \phi_I}{\partial x_l} \mathrm{d}\mathbf{x},$$

where $\chi > 0$ is the stiffening power and $\widehat{K}$ the reference element. This technique locally multiplies $\lambda$ or $\mu$ that are in factor of these integrals by a factor proportional to $\mathcal{J}_K(\mathbf{x})^{-\chi}$. $\chi$ determines the degree by which smaller elements are rendered stiffer than larger ones. We use $\chi = 1$.[2]
Afterwards, connectivity change can be performed on the mesh to improve the *quality* of the elements. It is an efficiency way to get rid of any shearing that occurs in the mesh.
This can be summed up by the algorithm 2.

**Algorithm 2** Basic moving mesh algorithm

1. Mesh deformation algorithm.
   (a) Compute body displacement from body translation and rotation data.
   (b) Solve linear elasticity equation with the finite element method at the order of the mesh.
   (c) Check validity of the obtained displacement and relax it if necessary/desired until the displacement is valid.
2. Move the mesh.

**Algorithm 3** Improved moving mesh algorithm

1. Mesh deformation algorithm.
   (a) Compute body displacement from body translation and rotation data.
   (b) Solve linear elasticity equation with the finite element method at the order of the mesh.
   (c) Perform high-order mesh optimization
   (d) Check validity of the obtained displacement and relax it if necessary/desired until the displacement is valid.
2. Move the mesh.

It can be noticed that when no optimization is done, shearing appears. Mesh optimization appears with the implementation of Algorithm 3 and occurs after each motion step. The high-order linear elasticity resolution gives to the elements a curvature that fits to the displacement of the sphere. Indeed, in front of the sphere, the deformed elements fit to the shape of the sphere, whereas in the wake, the curvature of elements

is made so that the shearing is reduced. This way, connectivity change is less numerous and appears later than in the straight case.

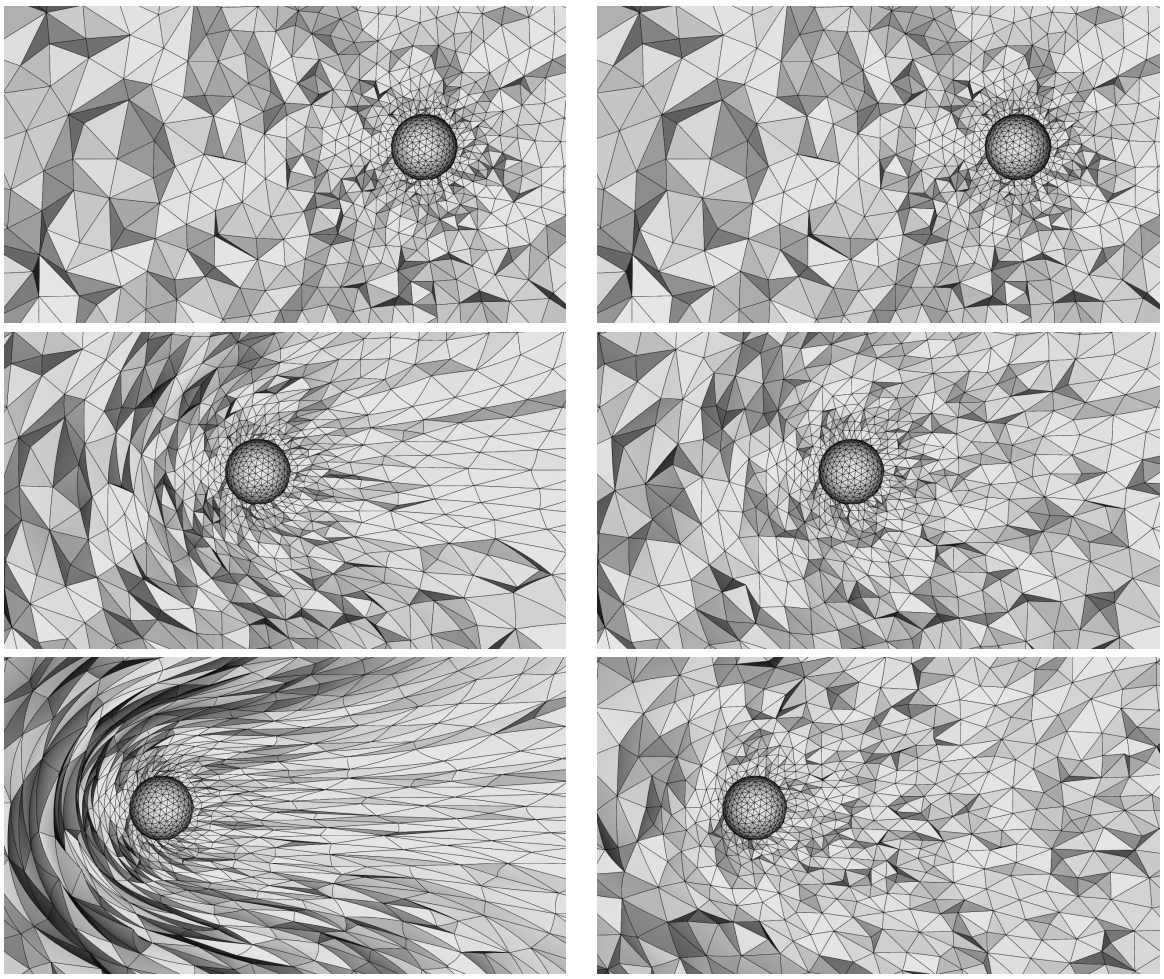

**Figure 14. From up to bottom, translation of a sphere in a $P^2$ mesh. Left, without mesh optimization operators. Right, with mesh optimization operators.**

| HLCRM mesh | Average Q | Best Q | Worst Q | # vertices | # tets | # invalid |
|---|---|---|---|---|---|---|
| Initial mesh | 1.2744 | 1.0000 | 2.5146 | 101658 | 72094 | 0 |
| Small displacement | 1.6474 | 1.0000 | 5.7105 | 101658 | 72094 | 0 |
| Small displacement & optimization | 1.4322 | 1.0000 | 3.1693 | 101153 | 71589 | 0 |
| Large displacement | $\infty$ | 1.0093 | $\infty$ | 101658 | 72094 | 84 |
| Large displacement & optimization | 1.4860 | 1.0000 | 3.5164 | 100854 | 71290 | 0 |

**Figure 15. Quality histograms associated to the moving sphere examples.**

## VII. Conclusion and perspectives: the generation of high-order curved boundary layers

The idea is to extend the boundary layer mesh generation method of[4] with curved elements. This approach relies on algorithm 4.

**Algorithm 4** Overall closed advancing-layer algorithm

For $i_{lay} = 1, .., N^K_{lay}$

1. Create layer $i_{lay}$: For each active point propose its optimal position using the advancing-layer method
2. If ( mesh deformation criteria ) Then
   $\mathbf{d}_{|\partial\Omega_h}$ = Get boundary vertex displacement from inflating boundary layer
   $\mathbf{d}$ = Get inner vertex displacement by solving the high order elasticity system $\left(\mathbf{d}_{|\partial\Omega_h}\right)$
   Else
   $\mathbf{d} = \beta\ \mathbf{d}$ increment vertex displacement by the growth rate
   EndIf
3. Set $t = 0$, $T = 1$ and vertex speed $\mathbf{v} = \mathbf{d}$
4. While $(t < T)$
   (a) $\delta t$ = Get moving mesh time step $\left(\mathcal{H}^k, \mathbf{v}, CFL^{geom}\right)$
   (b) $\mathcal{H}^k$ = Connectivity optimization
   (c) $\mathbf{v}^{opt}$ = Vertex smoothing
   (d) $\mathcal{H}^{k+1}$ = Move the mesh $\left(\mathcal{H}^k, \delta t, \mathbf{v}, \mathbf{v}^{opt}\right)$
   (e) Check mesh validity:
   If ( element invalid ) Then
   Cancel element's vertices displacements
   Freeze element's vertices : $\mathbf{v} = 0$
   EndIf
   (f) $t = t + \delta t$

   EndWhile
5. Move back vertices that have moved less than a threshold percentage of the layer size

EndFor

To extend the closed boundary layer approach presented in[4] to high-order meshes and directly generate curved boundary layer mesh, it is required to:

- start form an initial high-order mesh that is obtained using the method of section V.
- consider the connectivity-change moving mesh method for high-order mesh presented in section VI to deform the initial high-order mesh when the boundary layer mesh is inflated inside the domain.
- generate directly high-order elements in the boundary layer when it is inflated using the advancing layer approach presented in.[3]

The future work to do is the last item. To this end, the advancing layer method will be modified to take into account the high-order boundary layer elements. The new position of the nodes in the boundary layer will be given using the same process as the one for proposing the new position for the vertices.
Note that more accurate normals will be obtained as they will be computed on the high-order mesh instead of a $P^1$-straight mesh. This is an important point as the quality of the boundary layer is highly dependent on the accuracy of the normal computations.[3]

As preliminary results, the curvature technique of section V has been used. On the case of the sphere (see Fig. 16 left), the high-order curvature is not kept along the layers and the process generates a lot of invalid elements in the boundary layer mesh as they are highly anisotropic there whereas the curvature will be kept with the ongoing method. The question to keep this curvature or not is at this time an open question. Moreover, it can be noticed that the boundary layer is not well developed on the left for the NASA RO37 rotor (see Fig. 16 right). An high-order boundary layer mesh generation could be a way to enhance the development of the boundary layer in this area.

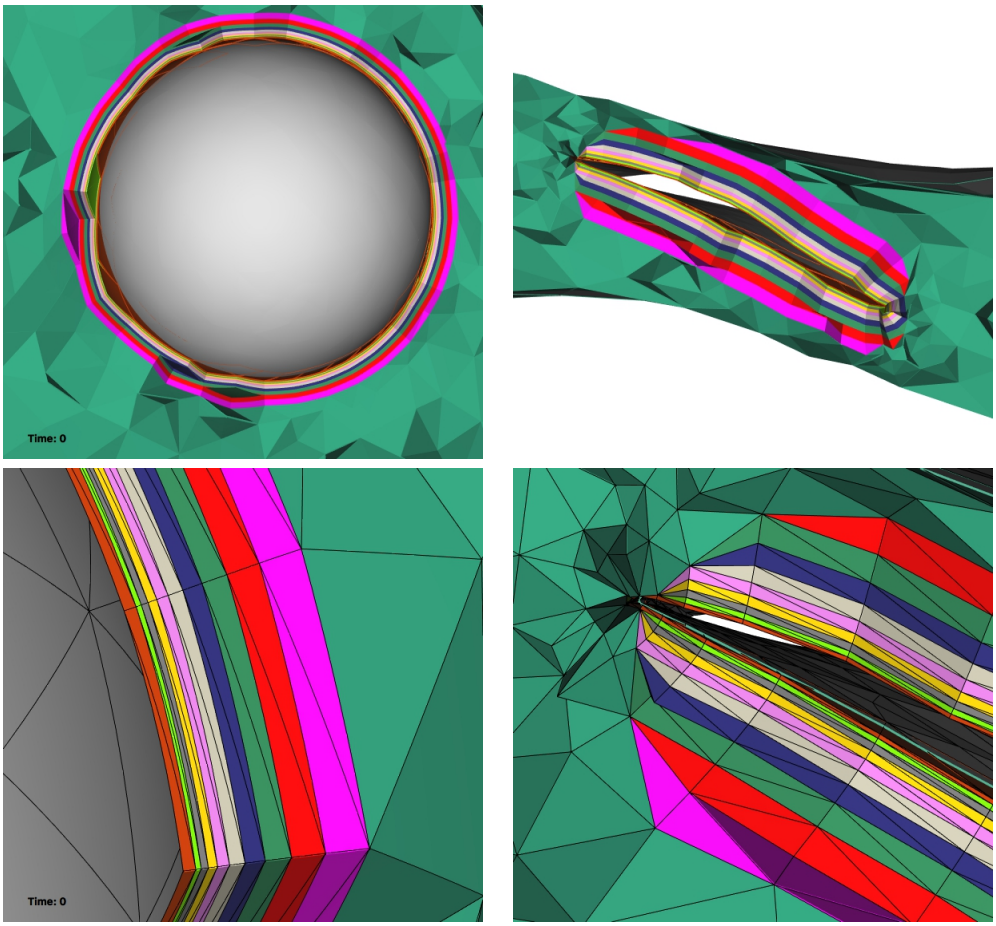


**Figure 16. Left, boundary layer mesh for a sphere with the method of section V. Right, straight boundary layer for the NASA RO37 rotor with problems to generate layers on the trailing edge ridge.**

## VIII. Acknowledgment

This work was supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH.

## References

[1]R. Abgrall, C. Dobrzynski, and A. Froehly. A method for computing curved meshes via the linear elasticity analogy, application to fluid dynamics problems. *International Journal for Numerical Methods in Fluids*, 76(4):246–266, 2014.

[2]F. Alauzet. A changing-topology moving mesh technique for large displacements. *Engineering with Computers*, 30(2):175–200, Apr 2014.

[3]F. Alauzet, A. Loseille, and D. Marcum. On a robust boundary layer mesh generation process. In *55th AIAA Aerospace Sciences Meeting*, AIAA Paper2017-0585, Grapevine, TX, USA, Jan 2017.

[4]F. Alauzet and D. Marcum. A closed advancing-layer method with connectivity optimization based mesh movement for viscous mesh generation. *Eng. w. Comp.*, 31:545–560, 2015.

[5]R. Aubry. Ensuring a smooth transition from semi-structured surface boundary layer mesh to fully unstructured anisotropic surface mesh. In *53rd AIAA Aerospace Sciences Meeting*, AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, 2015. DOI: 10.2514/6.2015-1507.

[6]H. Borouchaki and P.-L. George. *Meshing, Geometric Modeling and Numerical Simulation 1: Form Functions, Triangulations and Geometric Modeling*. John Wiley & Sons, 2017.

[7]P.G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland Publishing Company, 1978.

[8]P. Ciarlet Jr. and E. Lunéville. *La Méthode des Eléments Finis. De la Théorie à la Pratique. I. Concepts Généraux*. Coll. Les Cours, Les Presses de l'ENSTA, 200 pages, 3 2009.

[9]M. Fortunato and P.-O. Persson. High-order Unstructured Curved Mesh Generation Using the Winslow Equations. *J. Comput. Phys.*, 307(C):1–14, February 2016.

[10]P.-L. George and H. Borouchaki. Construction of tetrahedral meshes of degree two. *International Journal for Numerical Methods in Engineering*, 2012.

[11]P.-L. George and H. Borouchaki. Validité des éléments finis de Lagrange de degré 1 et 2. Research Report RR-8376, INRIA, October 2013.

[12]P.-L. George, H. Borouchaki, F. Alauzet, P. Laug, A. Loseille, and L. Maréchal. *Meshing, Geometric Modeling and Numerical Simulation 2: Metrics, Meshing and Mesh Adaptation*. John Wiley & Sons, 2018 (to appear).

[13]P.-L. George, H. Borouchaki, and N. Barral. Construction et validation des éléments réduits associés à un carreau simplicial de degré arbitraire. Research Report RR-8571, INRIA, July 2014.

[14]F. Haider, P. Brenner, B. Courbet, and J.-P. Croisille. Parallel implementation of k-exact finite volume reconstruction on unstructured grids. In *High order nonlinear numerical schemes for evolutionary PDEs*, pages 59 – 75, Bordeaux, France, 2014. Springer International Publishing.

[15]R. Hartmann and T. Leicht. Generation of unstructured curvilinear grids and high-order discontinuous Galerkin discretization applied to a 3D high-lift configuration. *International Journal for Numerical Methods in Fluids*, 82(6):316–333, 2016.

[16]J.S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods: algorithms, analysis and applications*. Springer Publishing Company, Incorporated, 2008.

[17]S. L. Karman, J T. Erwin, R. S. Glasby, and D. Stefanski. High-order mesh curving using WCN mesh optimization. In *46th AIAA Fluid Dynamics Conference*, AIAA AVIATION Forum. American Institute of Aeronautics and Astronautics, 2016. DOI: 10.2514/6.2016-3178.

[18]S. L. Karman, N. Wyman, and J. P. Steinbrenner. Mesh generation challenges: A commercial software perspective. In *23rd AIAA Computational Fluid Dynamics Conference*, AIAA AVIATION Forum. American Institute of Aeronautics and Astronautics, 2017. DOI: 10.2514/6.2017-3790.

[19]D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, Aug 1989.

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

[21]J. S. Masters. Mesh Manipulation for 3D Tetrahedral Meshes. In *53rd AIAA Aerospace Sciences Meeting*, AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, 2015. DOI: 10.2514/6.2015-0913.

[22]D. Moxey, D. Ekelschot, Ü. Keskin, S.J. Sherwin, and J. Peirò. High-order curvilinear meshing using a thermo-elastic analogy. *Computer-Aided Design*, 72(Supplement C):130 – 139, 2016. 23rd International Meshing Roundtable Special Issue: Advances in Mesh Generation.

[23]P.-O. Persson and J. Peraire. Curved mesh generation and mesh refinement using lagrangian solid mechanics. In *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, Aerospace Sciences Meetings. American Institute of Aeronautics and Astronautics, 2009. DOI: 10.2514/6.2009-949.

[24]E. Ruiz-Gironès, X. Roca, and J. Sarrate. High-order mesh curving by distortion minimization with boundary nodes free to slide on a 3D CAD representation. *Computer-Aided Design*, 72:52 – 64, 2016. 23rd International Meshing Roundtable Special Issue: Advances in Mesh Generation.

[25]M. Sharbatdar and C. Ollivier Gooch. Anisotropic mesh adaptation: Recovering quasi-structured meshes. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, Aerospace Sciences Meetings. American Institute of Aeronautics and Astronautics, 2013. DOI: 10.2514/6.2013-149.

[26]C. Sheng and C. Allen. Efficient mesh deformation using radial basis functions on unstructured meshes. In *42nd AIAA Fluid Dynamics Conference and Exhibit*, Fluid Dynamics and Co-located Conferences. American Institute of Aeronautics and Astronautics, 2012. DOI: 10.2514/6.2012-2685.

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

[28]T. Toulorge, J. Lambrechts, and J.-F. Remacle. Optimizing the geometrical accuracy of curvilinear meshes. *Journal of Computational Physics*, 310:361 – 380, 2016.

[29]M. Turner, J. Peirò, and D. Moxey. A Variational Framework for High-order Mesh Generation. *Procedia Engineering*, 163(Supplement C):340 – 352, 2016. 25th International Meshing Roundtable.

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

[31]A. Vlachos, P. Jörg, C. Boyd, and J. L. Mitchell. Curved PN Triangles. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, I3D '01, pages 159–166, New York, NY, USA, 2001. ACM.