# Geometrical Validity of High-Order Triangular Finite Elements

A. Johnen[1], J.-F. Remacle[2], and C. Geuzaine[1]

[1] Université de Liège, Department of Electrical Engineering and Computer Science, Montefiore Institute B28, Grande Traverse 10, 4000 Liège, Belgium
{a.johnen, cgeuzaine}@ulg.ac.be
[2] Institute of Mechanics, Materials and Civil Engineering, Université catholique de Louvain, Avenue Georges-Lemaître 4, 1348 Louvain-la-Neuve, Belgium
jean-francois.remacle@uclouvain.be

**Summary.** This paper presents a method to compute accurate bounds on Jacobian determinants of high-order (curvilinear) triangular finite elements. This method can be used to guarantee that a curvilinear triangle is geometrically valid, i.e., that its Jacobian determinant is strictly positive everywhere in its reference domain. It also provides an efficient way to measure the quality the triangles. The key feature of the method is to expand the Jacobian determinant using a polynomial basis, built using Bézier functions, that has both properties of boundedness and positivity. Numerical results show the sharpness of our estimates.

**Key words:** Finite element method; high-order methods; mesh generation; Bézier functions

## 1 Introduction

High-order finite element solvers critically depend on the availability of high-quality curvilinear meshes. Such curvilinear meshes are usually built by curving an initial, straight sided mesh [1, 2, 3]. If we assume that the straight sided mesh is composed of well shaped elements, curving elements introduces a "shape distortion" that should be controlled so that the final curvilinear mesh is also composed of well shaped elements.

In this paper we present a method to analyze the quality of curvilinear meshes in terms of their elementary Jacobian determinants. The method does not deal with the actual generation of the high order mesh. Instead, it provides an efficient way to guarantee that each curvilinear element is geometrically valid, i.e., that its Jacobian determinant is strictly positive everywhere in its reference domain. It also provides a way to measure the distortion of the curvilinear element. The key feature of the method is to adaptively expand

the elementary Jacobian determinants in a polynomial basis that has both properties of boundedness and positivity. Bézier functions are used to generate these bases in a recursive manner. Contrary to existing approaches using Bézier functions [4, 5], which can only provably detect valid elements (the validity of some triangles remaining uncertain), the adaptive scheme allows to provably detect both valid *and* invalid elements. The proposed method can be either used to check the validity and the distortion of an existing curvilinear mesh, or embedded in the curvilinear mesh generation procedure to assess the validity and the quality of the elements on the fly. The algorithm described in this paper has been implemented in the open source mesh generator Gmsh [6], where it is used in both ways.

## 2 Curvilinear Meshes, Jacobian Determinants and Distortion

Let us consider a mesh that consists of a set of straight-sided triangles of order $p$. Each triangle is defined geometrically through its nodes $\mathbf{x}_i$, $i = 1, \dots, N_p$ and a set of Lagrange shape functions $\mathcal{L}_i^{(p)}(\boldsymbol{\xi})$, $i = 1, \dots, N_p$ [7]. The Lagrange shape functions (of order $p$) are based on the nodes $\mathbf{x}_i$ and allow to map a reference unit triangle onto the real one:

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

The mapping $\mathbf{x}(\boldsymbol{\xi})$ should be bijective, which implies that the Jacobian determinant $\det \mathbf{x}_{,\boldsymbol{\xi}}$ (which is constant over the reference domain since the triangle is straight-sided) has to be strictly positive. In all what follows we will always assume that the straight-sided mesh is composed of well-shaped elements, so that the positivity of $\det \mathbf{x}_{,\boldsymbol{\xi}}$ is guaranteed. This standard setting is presented on Figure 1 for the quadratic triangle.

Let us now consider a curved triangle obtained after application of the curvilinear meshing procedure, i.e., after moving some or all of the nodes of the straight-sided triangle. The nodes of the deformed triangle are called $\mathbf{X}_i$, $i = 1 \dots N_p$, and we have

$$\mathbf{X}(\boldsymbol{\xi}) = \sum_{i=1}^{N_p} \mathcal{L}_i^{(p)}(\boldsymbol{\xi})\, \mathbf{X}_i. \tag{2}$$

Again, the deformed triangle is valid if the Jacobian determinant $J(\boldsymbol{\xi}) := \det \mathbf{X}_{,\boldsymbol{\xi}}$ is strictly positive everywhere over the $\boldsymbol{\xi}$ reference domain. The Jacobian determinant $J$, however, is not constant anymore over the reference domain, and computing $J_{\min} := \min_{\boldsymbol{\xi}} J(\boldsymbol{\xi})$ is necessary to ensure positivity.

The approach that is commonly used is to sample the Jacobian determinant on a very large number of points. Such a technique is however both

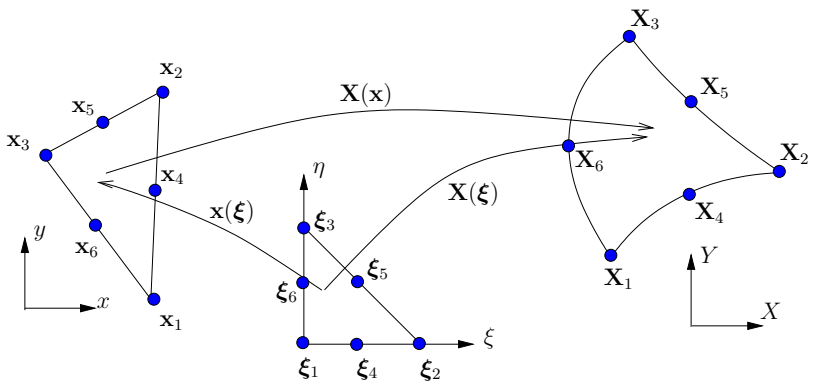



**Fig. 1.** Reference unit triangle in local coordinates $\boldsymbol{\xi} = (\xi, \eta)$ and the mappings $\mathbf{x}(\boldsymbol{\xi})$, $\mathbf{X}(\boldsymbol{\xi})$ and $\mathbf{X}(\mathbf{x})$.

expensive and not fully robust since we only get a necessary condition. In this paper we follow a different approach: because the Jacobian determinant $J$ is a polynomial in $\boldsymbol{\xi}$, $J$ can be interpolated exactly as a linear combination of specific polynomial basis function over the element. We would then like to obtain provable bounds on $J_{\min}$ by using the properties of these basis functions.

In addition, we are also interested in measuring the distortion induced by the transformation $\boldsymbol{X}(\boldsymbol{x})$ that maps straight sided elements onto curvilinear elements. It is possible to write its determinant in terms of the $\boldsymbol{\xi}$ coordinates as:

$$\det \boldsymbol{X}_{,\boldsymbol{x}} = \frac{\det \boldsymbol{X}_{,\boldsymbol{\xi}}}{\det \boldsymbol{x}_{,\boldsymbol{\xi}}} = \frac{J(\boldsymbol{\xi})}{\det \boldsymbol{x}_{,\boldsymbol{\xi}}}. \tag{3}$$

We call the determinant of the mapping $\delta(\boldsymbol{\xi}) := \det \boldsymbol{X}_{,\boldsymbol{x}}$ the distortion and it should be as close to $\delta = 1$ as possible in order not to degrade the quality of the straight sided element. In order to guarantee a reasonable distortion it is thus necessary to find a reliable bound on $J_{\min}$ and $J_{\max} := \max_{\boldsymbol{\xi}} J(\boldsymbol{\xi})$ over the whole element.

## 3 Jacobian Determinant Bounds using Bezier Functions

Computing exact bounds on the Jacobian determinant is only possible for simple cases. For example, the minimum Jacobian determinant of the planar 2nd order triangle is located either at its unique, easily computable stationary point, or on the boundary of the triangle.

For arbitrarily high-order curvilinear triangles, computing exact bounds is computationally impractical: the Jacobian determinant of a $p$-th order trian-

gle is a polynomial function of order at most $2(p-1)$. In order to compute approximate bounds, we propose to expand the Jacobian determinant using a polynomial basis that has both properties of boundedness and positivity—so that one can verify the positivity of the Jacobian determinant by checking the positivity of the coefficients in the expansion.

A first candidate for the expansion is the hierarchical basis [8], but quick analysis shows that the bounds are of poor quality. A better candidate is the Bézier basis [9]. Bézier functions are positive and also have the nice property that they form a partition of unity. As a consequence, the minimum (*resp.* maximum) of the coefficients in the expansion is always smaller (*resp.* greater) than the Jacobian determinant. In order to compute these coefficients for the high-order Lagrange triangles used in standard FEM codes [7], we must first compute the change of basis from the Lagrange to the Bézier basis.

### 3.1 From a Lagrange to a Bezier Basis

The order-$p$ Bézier functions for a triangle in terms of $\xi$ and $\eta$ are defined in a subspace $E_T$ of $\mathbb{R}^2$, with $E_T = \{(\xi,\eta) : \xi \geq 0,\ \eta \geq 0,\ \xi+\eta \leq 1\}$ [9]:

$$\mathcal{T}^{(p)}_{i,j}(\xi,\eta) := \binom{p}{i}\binom{p-i}{j}\ \xi^i\ \eta^j\ (1-\xi-\eta)^{p-i-j} \qquad (i+j \leq p), \tag{4}$$

where $\binom{a}{b} = \frac{a!}{b!(a-b)!}$ is the binomial coefficient. The Jacobian determinant of triangles is a polynomial of order at most $n = 2(p-1)$. Given the expansion of the Jacobian determinant $J$ in terms of Lagrange polynomials

$$J(\xi,\eta) = \sum_i J_i \mathcal{L}^{(n)}_i(\xi,\eta),$$

where the $J_i$'s are the values of the Jacobian determinant calculated at Lagrange points, it is possible to express it as an expansion into Bézier triangular polynomials:

$$J(\xi,\eta) = \sum_{i+j\leq n} b_{ij}\mathcal{T}^{(n)}_{i,j}(\xi,\eta), \tag{5}$$

where the coefficients $b_{ij}$ are the control values of the Bézier expansion.

Thanks to the properties of the Bézier functions, i.e. positivity and partition of unity, the control values bound the Jacobian,

$$\min_{(\xi,\eta)\in E_T} J(\xi,\eta) \geq b_{\min} := \min_{i,j} b_{ij} \quad \text{and} \quad \max_{(\xi,\eta)\in E_T} J(\xi,\eta) \leq b_{\max} := \max_{i,j} b_{ij}. \tag{6}$$

Moreover, there is only one non-zero function at each corner, as for example at $(0,1)$

$$\begin{cases} \mathcal{T}^{(n)}_{0,n}(0,1) = 1 \\ \mathcal{T}^{(n)}_{i,j}(0,1) = 0, \quad \forall\,(i,j) \neq (0,n). \end{cases}$$

This implies that the 3 control values $b_{00}$, $b_{0n}$ and $b_{n0}$, are values of the interpolated function, and we have

$$\min_{(\xi,\eta)\in E_T} J(\xi,\eta) \leq \min(b_{00}, b_{0n}, b_{n0}) \quad \text{and} \quad \max_{(\xi,\eta)\in E_T} J(\xi,\eta) \geq \max(b_{00}, b_{0n}, b_{n0}). \tag{7}$$

Since Lagrangian and Bézier functions span the same function space, computation of the Bézier values $b_i$ from the nodal values $J_i$ (and conversely) is done by a transformation matrix. The transformation matrix $\boldsymbol{T}^{(n)}_{\mathcal{B}\to\mathcal{L}}$, which computes nodal values from control values, is created by evaluating Bézier functions at sampling points:

$$\boldsymbol{T}^{(n)}_{\mathcal{B}\to\mathcal{L}} = \begin{bmatrix} \mathcal{T}^{(n)}_{0,0}(0,0) & \dots & \mathcal{T}^{(n)}_{n,0}(0,0) \\ \mathcal{T}^{(n)}_{0,0}(0,1/n) & \dots & \mathcal{T}^{(n)}_{n,0}(0,1/n) \\ \vdots & \ddots & \vdots \\ \mathcal{T}^{(n)}_{0,0}(1,0) & \dots & \mathcal{T}^{(n)}_{n,0}(1,0) \end{bmatrix}. \tag{8}$$

Those sampling points are taken uniformly, i.e., at the location of the nodes of the triangle of order $n$. The inverse transformation $\boldsymbol{T}^{(n)}_{\mathcal{L}\to\mathcal{B}}$ is ${\boldsymbol{T}^{(n)}_{\mathcal{B}\to\mathcal{L}}}^{-1}$ and from the interpolation of the Jacobian determinant (5), we can write

$$\begin{aligned} J &= \boldsymbol{T}^{(n)}_{\mathcal{B}\to\mathcal{L}}\, B, \\ B &= \boldsymbol{T}^{(n)}_{\mathcal{L}\to\mathcal{B}}\, J\,, \end{aligned} \tag{9}$$

where $B$ and $J$ are the vectors containing respectively the control values of the Jacobian determinant $b_{ij}$ and the $J_i$'s. For example, for quadratic triangles we obtain

$$\boldsymbol{T}^{(2)}_{\mathcal{L}\to\mathcal{B}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -1/2 & -1/2 & 0 & 2 & 0 & 0 \\ 0 & -1/2 & -1/2 & 0 & 2 & 0 \\ -1/2 & 0 & -1/2 & 0 & 0 & 2 \end{pmatrix}. \tag{10}$$

### 3.2 Adaptive Subdivision

Subdivision consists in interpolating the Jacobian determinant on subdomains of the triangle. By defining 4 triangular subdomains as shown in Figure 2 (top left) all the initial domain is recovered, and the subdivision can be carried out in a simple, recursive manner.

Consider the one-to-one linear map from the reference space to itself that associates a point $(\xi^{[q]}, \eta^{[q]}) \in {E_T}^{[q]}$ of the $q^{\text{th}}$ subdomain to every point $(\xi,\eta) \in E_T$ of the reference domain, where ${E_T}^{[q]}$ is the subspace of $\mathbb{R}^2$ that corresponds to the $q^{\text{th}}$ subdomain. The new interpolation must verify

$$J(\xi^{[q]},\eta^{[q]}) := \sum_{i+j\leq n} \mathcal{T}^{(n)}_{i,j}(\xi^{[q]},\eta^{[q]})\, b_{ij} = \sum_{i+j\leq n} \mathcal{T}^{(n)}_{i,j}(\xi,\eta)\, b^{[q]}_{ij} \qquad (\xi^{[q]},\eta^{[q]}) \in {E_T}^{[q]}, \tag{11}$$

where the coefficients $b^{[q]}_{ij}$ are the control values corresponding to the $q^{\text{th}}$ subdomain.

Considering the uniform sampling points $(\xi_k, \eta_k)$ as in equation (8) and their image $(\xi^{[q]}_k, \eta^{[q]}_k)$, the expression (11) reads

$$\begin{aligned}\begin{bmatrix} \mathcal{T}^{(n)}_{0,0}[\xi^{[q]}(0,0),\eta^{[q]}(0,0)] & \dots & \mathcal{T}^{(n)}_{n,0}[\xi^{[q]}(0,0),\eta^{[q]}(0,0)] \\ \mathcal{T}^{(n)}_{0,0}[\xi^{[q]}(0,1/n),\eta^{[q]}(0,1/n)] & \dots & \mathcal{T}^{(n)}_{n,0}[\xi^{[q]}(0,1/n),\eta^{[q]}(0,1/n)] \\ \vdots & \ddots & \vdots \\ \mathcal{T}^{(n)}_{0,0}[\xi^{[q]}(1,0),\eta^{[q]}(1,0)] & \dots & \mathcal{T}^{(n)}_{n,0}[\xi^{[q]}(1,0),\eta^{[q]}(1,0)] \end{bmatrix} B \\ := \boldsymbol{T}^{(n)}_{\mathcal{B}\to\mathcal{L}}{}^{[q]}\, B \;=\; \boldsymbol{T}^{(n)}_{\mathcal{B}\to\mathcal{L}}\, B^{[q]}, \end{aligned} \tag{12}$$

where $B^{[q]}$ is the vector containing the control values of the related subdomain. This implies that we have to compute only one matrix per subdomain:

$$B^{[q]} = \left[\boldsymbol{T}^{(n)}_{\mathcal{L}\to\mathcal{B}}\, \boldsymbol{T}^{(n)}_{\mathcal{B}\to\mathcal{L}}{}^{[q]}\right] B = \boldsymbol{M}^{[q]} B. \tag{13}$$

Each set of new control values bounds the Jacobian determinant on its own subdomain and we have:

$$b'_{\min} := \min_{i,j,q} b^{[q]}_{ij} \;\leq\; J_{\min} \;\leq\; \min_{q} \min(b^{[q]}_{00}, b^{[q]}_{0n}, b^{[q]}_{n0}) \tag{14}$$

and

$$\max_{q} \max(b^{[q]}_{00}, b^{[q]}_{0n}, b^{[q]}_{n0}) \;\leq\; J_{\max} \;\leq\; b'_{\max} := \max_{i,q} b^{[q]}_{ij}. \tag{15}$$

If an estimate is not sufficiently sharp, we can thus simply subdivide the appropriate parts of the element. This leads to a simple adaptive algorithm, exemplified in Figure 2. In this particular case the original estimate (6)-(7) is not sharp enough. After one subdivision, the Jacobian determinant is proved to be positive on all subdomains. In practice, as will be seen in Section 4, a few levels of refinement lead to the desired accuracy. The convergence of the subdivision can be proven to be quadratic [10, 11].

### 3.3 Implementation

As mentioned in Section 2, the Jacobian determinant bounds can be used to either make the distinction between valid and invalid elements with respect to a condition on $J_{\min}$, or to measure the quality of the elements by systematically computing $J_{\min}$ and $J_{\max}$ with a prescribed accuracy.

In both cases the same operations are executed on each element. First, the Jacobian determinant is sampled on a determined number of points $N_s =$

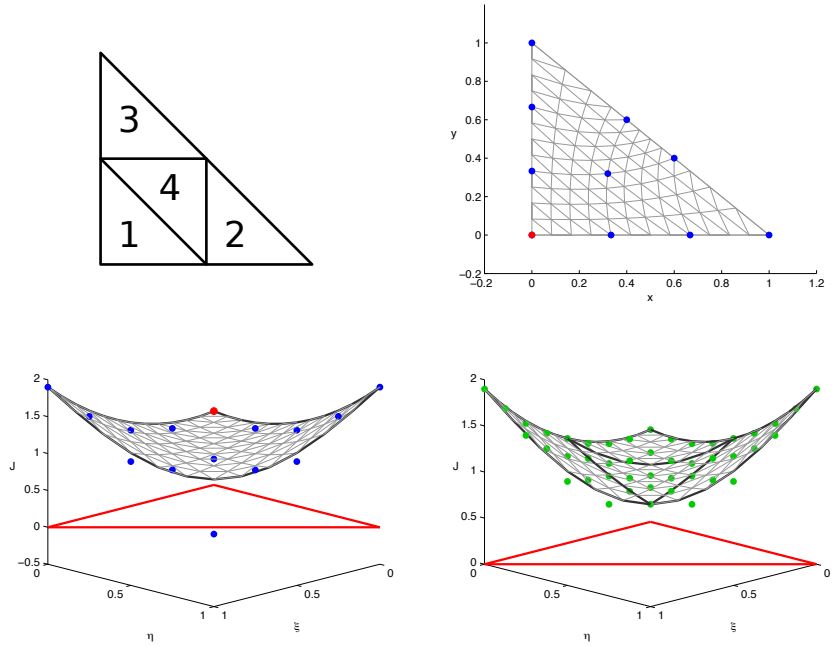



**Fig. 2.** Top left: 4 subdomains of the triangle. Top right: third-order planar triangle. Bottom left: exact Jacobian determinant and control values (dots) on the original control points (before subdivision); the validity of the element cannot be asserted. Bottom right: exact Jacobian determinant and control values (dots) after one subdivision; the element is provably correct.

$\frac{(n+1)(n+2)}{2}$, equal to the dimension of the Jacobian determinant space, and so to the number of Bézier functions. Second, Bézier values are computed. Then adaptive subdivision is executed if necessary. Algorithm 1 shows in pseudo-code the algorithm used to determine whether the Jacobian determinant of the element is everywhere positive or not. The algorithm assumes that the corner points of the triangle are ordered at the 3 first indices.

Algorithm 2 could be further improved by optimizing the loop on line 5, by first selecting $q$ for which we have the best chance to have a negative Jacobian determinant (line 4, algo 2). In practice, this improvement is not significant since the only case for which we can save calculation is for invalid elements—and the proportion of them that require subdivision in order to be detected is usually small. Note that we may also want to find, for example, all the elements for which the Jacobian determinant is somewhere smaller than 20% of its average. We then just have to compute this average and replace the related lines (4 and 7 for algorithm 1).

Another possible improvement is to relax the condition of rejection. We could accept elements for which all control values are positive but reject an

**Algorithm 1**: Check if a triangle is valid or invalid

**Input**: a pointer to a triangle.
**Output**: *true* if the triangle is valid, *false* if the element is invalid

set sampling points $P_i$, $i = 1, \ldots, N_s$;
compute Jacobian determinants $J_i$ at points $P_i$;
**for** $i = 1$ **to** $N_s$ **do**
    **if** $J_i <= 0$ **then return** *false*;
compute Bézier coefficients $b_i$, $i = 1, \ldots, N_s$ using (9);
$i = 1$;
**while** $i \leq N_s$ **and** $b_i \geq 0$ **do**
    $i = i + 1$;
**if** $i > N_s$ **then return** *true*;
call algorithm 2 with $b_i$ as arguments and return its output;

**Algorithm 2**: Compute the control values of the subdivisions

**Input**: Bézier coefficients $b_i$, $i = 1, \ldots, N_s$
**Output**: *true* if the Jacobian determinant on the domain is everywhere positive, *false* if not

compute new Bézier coefficients $b_i^{[q]}$, $q = 1, \ldots, 4$ as in equation (13);
**for** $q = 1$ **to** 4 **do**
    **for** $i = 1$ **to** 3 **do**
        **if** $b_i^{[q]} <= 0$ **then return** *false*;
**for** $q = 1$ **to** 4 **do**
    $i = 1$;
    **while** $i \leq N_s$ **and** $b_i^{[q]} \geq 0$ **do**
        $i = i + 1$;
    **if** $i \leq N_s$ **then**
        call algorithm 2 with $b_i^{[q]}$ as arguments and store *output*;
        **if** *output* = *false* **then return** *false*;
**return** *true*;

element as soon as we find a Jacobian determinant value smaller than a defined percent of the average Jacobian determinant. The computational gain can be significant, since elements that were classified as good and that needed a lot of subdivisions (and have a Jacobian determinant close to zero) will be instead rapidly be detected as invalid.

More interestingly, the computation of sampled Jacobian determinants and the computation of Bézier control values in algorithm 1 can easily be executed for a whole groups of elements at the same time. This allows to use efficient

BLAS 3 (matrix-matrix product) functions, which significantly speeds up the computations.

The algorithm used for all the tests in the next section is implemented in the open source mesh generator Gmsh [6] as the `AnalyseCurvedMesh` plugin.

## 4 Numerical Results

We start by comparing the new adaptive computation of Jacobian determinant bounds with the brute-force sampling of the Jacobian determinant for the detection of invalid high-order triangles.

The points at which we sample the Jacobian determinant for the brute-force method are taken as the nodes of an element of order $k$. We started the test for $k = 1$ and we incremented $k$ until the brute-force approach detected all the invalid elements. We still executed the algorithm 10 times (while incrementing $k$) so as to plot the change in the number of invalid element detected. In order to make the comparison as fair as possible, we have implemented the brute-force computation as efficiently as possible, i.e., for $k$ ($> n$) sufficiently large we sample the Jacobian determinant on the points computed for an element at order $n$ (the order of the Jacobian determinant) and then compute the desired Jacobian determinant values by a matrix-vector product, just like in our own adaptive method.

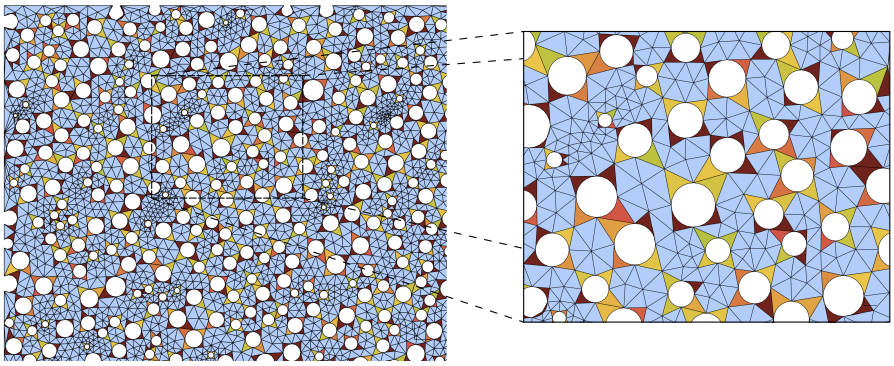


**Fig. 3.** Two-dimensional mesh with sixth order triangles; 23.6% of the elements are curved. The straight elements are in blue and the invalid are in dark red. The other are colored in function of the minimum of their distortion. They are green if they are nearly straight ($J_{\min}/J_{max} \simeq 1$) and rather light red if really distorted ($J_{\min}/J_{max} \simeq 0$).

We consider the two-dimensional microstructure with circular holes depicted in Figure 3, meshed with 331,050 sixth-order triangles. In this mesh

78,180 triangles are curved, and 45,275 are invalid. The new algorithm successfully detects all the 45,275 invalid elements in 6.194s. Some elements needed as much as 8 levels of subdivisions in order to be classified: see Table 1. The brute-force approach required 666 sample points per triangle in order to detect all the invalid elements, and took 4 times longer. But far worse, increasing the number of sampling points beyond 666 can actually lead to a decreased accuracy of the prediction, as shown in Figure 4.

| | Curved Element Classification | |
|---|---|---|
| | Valid curved elements | Invalid curved elements |
| First stage | 29303 | 44967 |
| 1 subdivision | 2436 | - |
| 2 subdivisions | 1119 | 299 |
| 3 subdivisions | 23 | - |
| 4 subdivisions | 10 | 4 |
| 5 subdivisions | 9 | 2 |
| 6 subdivisions | 5 | - |
| 7 subdivisions | - | 2 |
| 8 subdivisions | - | 1 |
| Subtotal | 32,905 | 45,275 |
| Total | 78,180 | |

**Table 1.** Number of elements detected as valid or invalid at each stage of the adaptive algorithm; 5 % of the curved elements had to be subdivided adaptively.

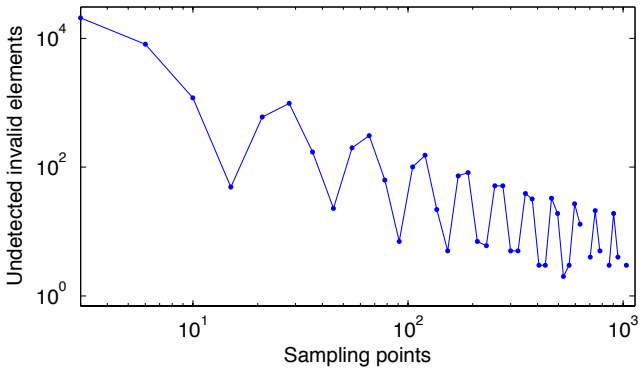
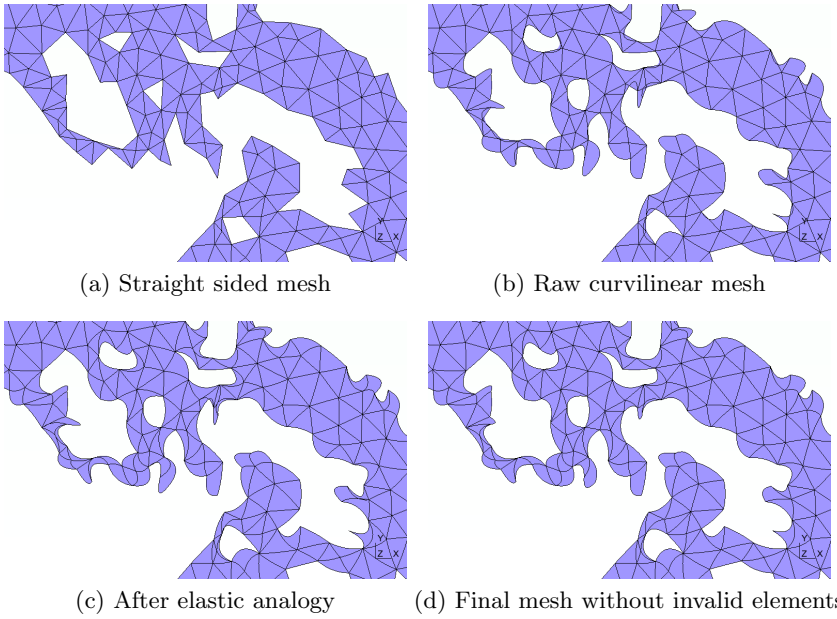



**Fig. 4.** Number of undetected invalid elements using brute-force sampling of the Jacobian determinant. The three data points not displayed correspond to the correct result, i.e., when no invalid triangle is left undetected.

Let us now examine the use of the adaptive Jacobian determinant bounds in the curvilinear meshing algorithms as implemented in Gmsh. We consider the mesh of a rather coarse version of the world ocean. In our CAD model, shorelines are described using cubic B-splines: for example, Europe and Asia are discretized by only one B-spline with about 3,500 control points. The description of this kind of meshing procedure is described in [12]. The quadratic triangular mesh is generated as follows. We first generate a straight sided mesh (see Figure 5/(a)). Then, every mesh edge that is classified on a model edge is curved by snapping its center vertex on the model edge. High order nodes are then inserted in the middle of every edge that is classified on a model face (see Figure 5/(b)). This simple procedure does not guarantee that the final mesh is valid. In our case, 175 elements are invalid. Then, a global elasticity analogy is applied to the quadratic mesh that enables to reduce the number of invalid elements to 70 (see Figure 5/(c)). Then local optimizations are performed to remove all invalid elements (see Figure 5/(d)). The final curvilinear mesh contains about 30% of curved elements. During the meshing process, the adaptive Jacobian determinant bound computation allowed to detected all invalid elements. After optimization, the final mesh is composed of elements that have a distortion $> 0.1$.



(a) Straight sided mesh (b) Raw curvilinear mesh

(c) After elastic analogy (d) Final mesh without invalid elements

**Fig. 5.** The four stages of the curvilinear mesh procedure for the world ocean, meshed with second order triangles.

The validity check presented in this article applies directly to triangles in the plane as shown in the previous examples. When meshing 3D (curved) surfaces, an additional mapping between the parametric plane where the mesh generation is performed [6] and the surface has to be taken into account. This mapping depends on the underlying geometrical representation of the surface: B-Spline or NURBS-based for CAD models, piece-wise polynomial for reparametrized STL models [13, 14], ...

In Figure 6 we used the adaptive bounds in the parametric plane for generating a fourth-order mesh of the STEP model of a rotor. After generating a first order mesh (Figure 6(a)) and snapping the high-order vertices on the geometrical model (Figure 6(b)), the adaptive Jacobian determinant bound computation allowed to pinpoint invalid elements. The final, locally optimized mesh is displayed in Figure 6(c).

This procedure however does not guarantee the general validity of the 3D mesh. Indeed, a valid element in the parametric plane can be made invalid after mapping to the curved surface since only the nodes of the triangle are mapped (and problems can arise between these nodes depending on the actual mapping). The converse is also true: an invalid element in the parametric plane can become valid after mapping. This is the subject of ongoing research, as is the optimization of the quality of general curvilinear finite elements.

## 5 Conclusion

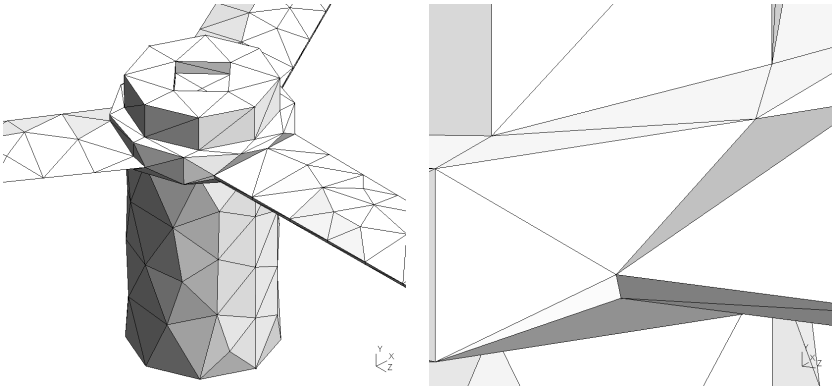
In this paper we presented a way to compute accurate bounds on Jacobian determinants of high-order triangular finite elements, based on the efficient expansion of these Jacobian determinants in terms of Bézier functions. The proposed algorithm can either be used to determine the validity or invalidity of curved elements, or provide an efficient way to measure their distortion.

Numerical tests show that the method is robust, and a user-defined error tolerance permits to adjust the accuracy vs. computational time ratio. The extension to all classical finite element shapes (quadrangles, tetrahedra, prisms and hexahedra) is currently underway in the open source finite element mesh generator Gmsh.

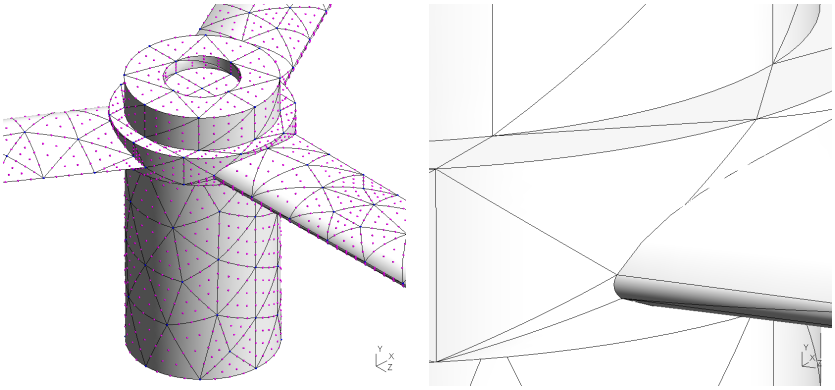
## Acknowledgement

This research project was funded in part by the Walloon Region under WIST 3 grant 1017074 (DOMHEX).

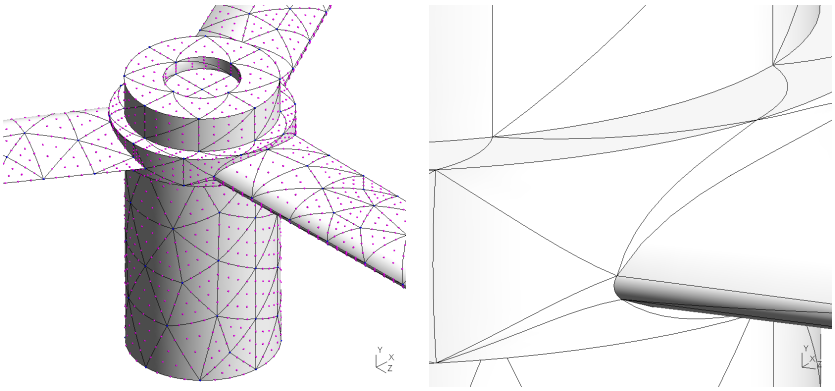
Authors gratefully thank E. Bechet from the University of Liège and K. Hillewaert from Cenaero for insightful discussions about Bézier functions and curvilinear mesh generation. Authors also thank V. D. Nguyen for providing the microstructure geometry used in Figure 3.



(a) Straight sided mesh



(b) Raw curvilinear mesh



(c) After elastic analogy

**Fig. 6.** Curvilinear mesh of a rotor using fourth-order curved triangles.

## References


1. S. Dey, R. M. O'Bara, and M. S. Shephard. Curvilinear mesh generation in 3D. *Computer Aided Geom. Design*, 33:199–209, 2001.
2. M. S. Shephard, J. E. Flaherty, K. E. Jansen, X. Li, X. Luo, N. Chevaugeon, J.-F. Remacle, M. W. Beall, and R. M. O'Bara. Adaptive mesh generation for curved domains. *Applied Numerical Mathematics*, 52:251–271, 2005.
3. S. J. Sherwin and J. Peiró. Mesh generation in curvilinear domains using high-order elements. *International Journal for Numerical Methods in Engineering*, 53:207–223, 2002.
4. X.-J. Luo, M. S. Shephard, R. M. O'Bara, R. Nastasia, and M. W. Beall. Automatic p-version mesh generation for curved domains. *Engineering with Computers*, 20:273–285, 2004.
5. O. Sahni, X.J. Luo, K.E. Jansen, and M.S. Shephard. Curved boundary layer meshing for adaptive viscous flow simulations. *Finite Elements in Analysis and Design*, 46:132–139, 2010.
6. C. Geuzaine and 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):1309–1331, 2009.
7. T. Hughes. *The Finite Element Method*. Dover, 2003.
8. I. Babuška, B. Szabò, and R. L. Actis. Hierarchic models for laminated composites. *International Journal for Numerical Methods in Engineering*, 33:503–535, 1992.
9. G. E. Farin. *Curves and surfaces for CAGD: a practicle guide*. Morgan-Kaufmann, 2002.
10. J. M. Lane and R. F. Riesenfeld. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(1):35–46, 1980.
11. E. Cohen and L. L. Schumacker. Rates of convergence of control polygons. *Computer Aided Geometric Design*, 2:229–235, 1985.
12. J. Lambrechts, R. Comblen, V. Legat, C. Geuzaine, and J.-F. Remacle. Multiscale mesh generation on the sphere. *Ocean Dynamics*, 58:461–473, 2008.
13. J.-F. Remacle, C. Geuzaine, G. Compère, and E. Marchandise. High-quality surface remeshing using harmonic maps. *International Journal for Numerical Methods in Engineering*, 83(4):403–425, 2010.
14. E. Marchandise, C. Carton de Wiart, W. G. Vos, C. Geuzaine, and J.-F. Remacle. High quality surface remeshing using harmonic maps. Part II: Surfaces with high genus and of large aspect ratio. *International Journal for Numerical Methods in Engineering*, 86(11):1303–1321, 2011.