
Geometrical Validity of Curvilinear Finite Elements

A. Johnen¹, J.-F. Remacle², and C. Geuzaine¹

¹ Université de Liège, Department of Electrical Engineering and Computer Science, Montefiore Institute B28, Grande Traverse 10, 4000 Liège, Belgium
`{ajohnen, 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. In this paper, we describe a way to compute accurate bounds on Jacobians of curvilinear finite elements of all kinds. Our condition enables to guarantee that an element is geometrically valid, i.e., that its Jacobian is strictly positive everywhere in its reference domain. It also provides an efficient way to measure the distortion of curvilinear elements. The key feature of the method is to expand the Jacobian 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

There is a growing consensus in the Finite Element community that higher-order discretization methods will replace at some point the solvers of today, at least for part of their applications. These high-order methods require a good accuracy of the geometrical discretization to be accurate—in other words, such methods will critically depend on the availability of high-quality curvilinear meshes.

The usual way of building such curvilinear meshes is to first generate a straight sided mesh. Then, mesh entities that are classified on the curved boundaries of the domain are curved accordingly [1, 2, 3]. Some internal mesh entities may be curved as well. 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. The optimization of the shape distortion is

a computationally expensive operation, especially when applied globally over the full mesh. It is thus crucial to be able to get fast and accurate bounds on the distortion in order to 1) evaluate the quality of the elements during the optimization process; and 2) reduce the sets of elements to be optimized, so that the optimization can be applied locally, i.e., only where it is necessary.

In this paper we present a method to analyze curvilinear meshes in terms of their elementary Jacobians. The method does not deal with the actual generation/optimization 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 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 Jacobians in a polynomial basis that has both properties of boundedness and positivity. Bézier functions are used to generate these bases in a recursive manner. 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 [4], where it is used in both ways.

2 Curvilinear Meshes, Distortion and Jacobian Bounds

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

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

The mapping $\mathbf{x}(\boldsymbol{\xi})$ should be bijective, which means that it should admit an inverse. This implies that the Jacobian $\det \mathbf{x}_{,\boldsymbol{\xi}}$ 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 element obtained after application of the curvilinear meshing procedure, i.e., after moving some or all of the nodes of the straight-sided element. The nodes of the deformed element are called \mathbf{X}_i , $i = 1 \dots N_p$, and we have

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

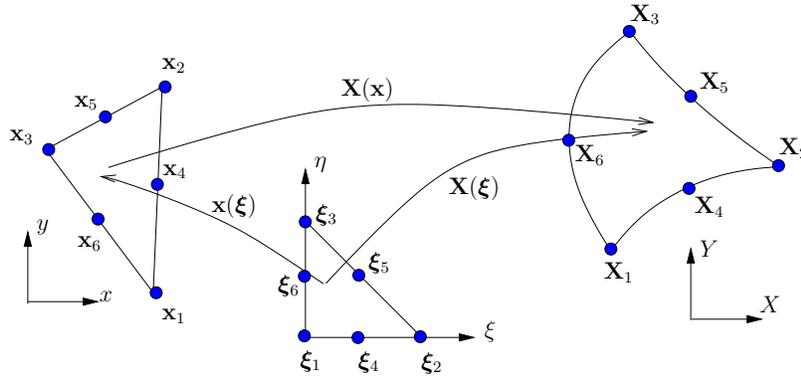


Fig. 1. Reference unit triangle in local coordinates $\xi = (\xi, \eta)$ and the mappings $\mathbf{x}(\xi)$, $\mathbf{X}(\xi)$ and $\mathbf{X}(\mathbf{x})$.

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

The approach that is commonly used is to sample the Jacobian on a very large number of points. Such a technique is however both expensive and not fully robust since we only get a necessary condition. In this paper we follow a different approach: because the Jacobian J is a polynomial in ξ , 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 to guaranteeing the geometrical validity of the curvilinear element, we are also interested in quantifying the distortion of the curvilinear element, i.e., the deformation induced by the curving. To this end, let us consider the transformation $\mathbf{X}(\mathbf{x})$ that maps straight sided elements onto curvilinear elements. It is possible to write this determinant in terms of the ξ coordinates as:

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

We call $\mathbf{X}(\mathbf{x})$ the distortion mapping and its determinant $\delta(\xi) := \det \mathbf{X}_{,\mathbf{x}}$ the distortion. The distortion δ should be as close to $\delta = 1$ as possible in order not to degrade the quality of the straight sided element. Elements that have negative distortions are of course invalid but elements that have distortions $\delta \ll 1$ or $\delta \gg 1$ lead to some alteration of the conditioning of the finite element problem. In order to guarantee a reasonable distortion it is thus necessary to find a reliable bound on J_{\min} and $J_{\max} := \max_{\xi} J(\xi)$ over the whole element.

Note that many different quality measures can be defined based on the Jacobian J . For example, one could look at the Jacobian divided by its average over the element instead of looking at the distortion. In any case, we see that obtaining bounds on J_{\min} and J_{\max} is still the main underlying challenge.

3 Jacobian Bounds for Second Order Planar Triangles

We start our analysis with the particular case of second order triangles for which a direct computation of J_{\min} is relatively easy. The determinant $J(\boldsymbol{\xi}) = J(\xi, \eta)$ for a planar triangle at order p is a polynomial in ξ and η of order at most $2(p-1)$. For quadratic planar triangles, $J(\xi, \eta)$ is therefore quadratic at most in ξ and η .

The geometry of the six-node quadratic triangle is shown in Figure 1. Inspection reveals two types of nodes: corners (1, 2 and 3) and midside nodes (4, 5 and 6). If J_i is defined as $J(\xi, \eta)$ evaluated at node i , it is possible to write the Jacobian exactly as a finite element expansion whose coefficients are the Jacobian at the nodes:

$$\begin{aligned}
 J(\xi, \eta) = & J_1 \underbrace{(1-\xi-\eta)(1-2\xi-2\eta)}_{\mathcal{L}_1^{(2)}(\xi, \eta)} + J_2 \underbrace{\xi(2\xi-1)}_{\mathcal{L}_2^{(2)}(\xi, \eta)} + J_3 \underbrace{\eta(2\eta-1)}_{\mathcal{L}_3^{(2)}(\xi, \eta)} + \\
 & J_4 \underbrace{4(1-\xi-\eta)\xi}_{\mathcal{L}_4^{(2)}(\xi, \eta)} + J_5 \underbrace{4\xi\eta}_{\mathcal{L}_5^{(2)}(\xi, \eta)} + J_6 \underbrace{4(1-\xi-\eta)\eta}_{\mathcal{L}_6^{(2)}(\xi, \eta)}. \quad (4)
 \end{aligned}$$

In equation (4), the functions $\mathcal{L}_i^{(2)}(\xi, \eta)$ are the equidistant quadratic Lagrange shape functions that are commonly used in the finite element community [5].

We first show how to compute the exact minimal Jacobian J_{\min} . Then we examine different bounds that can be provided on J_{\min} by exploiting the properties of the basis used in the Jacobian expansion.

3.1 Exact Computation of J_{\min}

From equation (4), the stationary point of J can be computed by solving

$$\frac{\partial J}{\partial \xi} = \frac{\partial J}{\partial \eta} = 0, \quad (5)$$

which leads to the following linear system of two equations and two unknowns ξ_{sta} and η_{sta} :

$$\begin{bmatrix} 4(J_1 + J_2 - 2J_4) & 4(J_1 - J_4 + J_5 - J_6) \\ 4(J_1 - J_4 + J_5 - J_6) & 4(J_1 + J_3 - 2J_6) \end{bmatrix} \begin{pmatrix} \xi_{sta} \\ \eta_{sta} \end{pmatrix} = \begin{pmatrix} -(-3J_1 - J_2 + 4J_4) \\ -(-3J_1 - J_3 + 4J_6) \end{pmatrix}. \quad (6)$$

Algorithm 1 allows to compute the minimal Jacobian over one quadratic planar element exactly. If the minimum of the function is outside of the element,

it computes the minimum on its border assuming a function $\text{MINQ}(a, b, c)$ that computes

$$\text{MINQ}(a, b, c) = \min_{x \in [0,1]} a x^2 + b x + c. \quad (7)$$

Algorithm 1: Exact computation of J_{\min} over a quadratic triangle

```

1 compute nodal Jacobians  $J_i, i = 1, \dots, 6$ ;
2 compute  $\xi_{sta}, \eta_{sta}$  as in equation (6);
3 if  $\eta_{sta} > 0$  and  $\xi_{sta} > 0$  and  $1 - \xi_{sta} - \eta_{sta} > 0$  then
4   |  $J_{\min} = \min(J(\xi_{sta}, \eta_{sta}), J_1, J_2, J_3)$ ;
5 else
6   |  $m_1 = \text{MINQ}(2(J_1 + J_2 - 2J_4), -3J_1 - J_2 + 4J_4, J_1)$ ;
7   |  $m_2 = \text{MINQ}(2(J_1 + J_3 - 2J_6), -3J_1 - J_3 + 4J_6, J_1)$ ;
8   |  $m_3 = \text{MINQ}(2(J_2 + J_3 - 2J_5), -3J_2 - J_3 + 4J_5, J_2)$ ;
9   |  $J_{\min} = \min(m_1, m_2, m_3)$ ;
10 return  $J_{\min}$ ;
```

Although Algorithm 1 is quite simple, applying similar techniques for higher order elements would become extremely expensive computationally. Instead of trying to evaluate J_{\min} directly, we should try to compute (the sharpest possible) bounds in a computationally efficient manner.

3.2 The Principle for Computing Bounds on J_{\min}

It is obvious that a necessary condition for having $J(\xi, \eta) > 0$ everywhere is that $J_i > 0, i = 1, \dots, 6$. Yet, this condition is not sufficient. The expression (4) does not give more information because the quadratic Lagrange shape functions $\mathcal{L}_i^{(2)}(\xi, \eta)$ change sign on the reference triangle. What polynomial basis should we chose to obtain usable bounds?

The first idea is to expand (4) into monomials, which gives:

$$J(\xi, \eta) = J_1 + (-3J_1 - J_2 + 4J_4)\xi + (-3J_1 - J_3 + 4J_6)\eta + 4(J_1 - J_4 + J_5 - J_6)\xi\eta + 2(J_1 + J_2 - 2J_4)\xi^2 + 2(J_1 + J_3 - 2J_6)\eta^2. \quad (8)$$

Every monomial being positive on the reference triangle, we have now a set of sufficient conditions that can be written as

$$4J_4 \geq 3J_1 + J_2, \quad 4J_6 \geq 3J_1 + J_3, \quad J_1 + J_5 \geq J_4 + J_6, \quad J_1 + J_2 \geq 2J_4, \quad J_1 + J_3 \geq 2J_6.$$

However these constraints do not provide a usable bound on J_{\min} and break the symmetry of the expression with respect to a rotation of corner nodes.

A second idea is to expand (4) in terms of the second order hierarchical basis functions $\psi_i(\xi, \eta), i = 1, \dots, 6$, which are also positive on the triangle [6]:

$$\begin{aligned}
J(\xi, \eta) = & J_1 \underbrace{(1 - \xi - \eta)}_{\psi_1(\xi, \eta)} + J_2 \underbrace{\xi}_{\psi_2(\xi, \eta)} + J_3 \underbrace{\eta}_{\psi_3(\xi, \eta)} + (4J_4 - 2J_1 - 2J_2) \underbrace{(1 - \xi - \eta)\xi}_{\psi_4(\xi, \eta)} + \\
& (4J_5 - 2J_3 - 2J_2) \underbrace{\xi\eta}_{\psi_5(\xi, \eta)} + (4J_6 - 2J_1 - 2J_3) \underbrace{(1 - \xi - \eta)\eta}_{\psi_6(\xi, \eta)}. \quad (9)
\end{aligned}$$

This last expression has the right symmetry, and leads to the following validity conditions:

$$J_1 \geq 0, \quad J_2 \geq 0, \quad J_3 \geq 0, \quad 4J_4 \geq 2J_1 + 2J_2, \quad 4J_5 \geq 2J_2 + 2J_3, \quad 4J_6 \geq 2J_3 + 2J_1. \quad (10)$$

Writing $J(\xi, \eta) := \sum_{i=1}^6 \psi_i(\xi, \eta) K_i$, we have

$$\min_{\xi, \eta} J(\xi, \eta) = \min_{\xi, \eta} \left(\sum_i \psi_i(\xi, \eta) K_i \right) \geq \min_{\xi, \eta} \left(\sum_i \psi_i(\xi, \eta) \right) \min_i K_i = \min_i K_i,$$

because $\sum_i \psi_i = 1 + \xi + \eta - \xi^2 - \eta^2 - \xi\eta$ has its minimum on the corner vertices (where its value is equal to 1). And since K_i , $i = 1, \dots, 3$ are values of the Jacobian, they form an upper bound on it. Thus, expansion (9) leads to the following estimate for the minimum of the Jacobian over the triangle:

$$\begin{aligned}
J_{\min} & \geq \min\{J_1, J_2, J_3, 4J_4 - 2J_1 - 2J_2, 4J_5 - 2J_2 - 2J_3, 4J_6 - 2J_3 - 2J_1\} \\
& \leq \min\{J_1, J_2, J_3\}. \quad (11)
\end{aligned}$$

It is easy to see that the estimate is however of very poor quality: for an element that has a constant and positive J , (11) simply tells us that $J_{\min} \geq 0$.

In order to find a sharper estimate, instead of the hierarchical quadratic functions $\psi_i(\xi, \eta)$, we can use the quadratic triangular Bézier functions $\mathcal{B}_2^{(2)}(\xi, \eta)$ [7]:

$$\begin{aligned}
J(\xi, \eta) = & J_1 \underbrace{(1 - \xi - \eta)^2}_{\mathcal{B}_1^{(2)}(\xi, \eta)} + J_2 \underbrace{\xi^2}_{\mathcal{B}_2^{(2)}(\xi, \eta)} + J_3 \underbrace{\eta^2}_{\mathcal{B}_3^{(2)}(\xi, \eta)} + \\
& \left(2J_4 - \frac{1}{2}(J_2 + J_1) \right) \underbrace{2\xi(1 - \xi - \eta)}_{\mathcal{B}_4^{(2)}(\xi, \eta)} + \left(2J_5 - \frac{1}{2}(J_3 + J_2) \right) \underbrace{2\xi\eta}_{\mathcal{B}_5^{(2)}(\xi, \eta)} + \\
& \left(2J_6 - \frac{1}{2}(J_1 + J_3) \right) \underbrace{2\eta(1 - \xi - \eta)}_{\mathcal{B}_6^{(2)}(\xi, \eta)}. \quad (12)
\end{aligned}$$

Since $\sum_{i=1}^6 \mathcal{B}_i^{(2)}(\xi, \eta) = 1$, we obtain the following estimate

$$\begin{aligned}
J_{\min} & \geq \min \left\{ J_1, J_2, J_3, 2J_4 - \frac{J_1 + J_2}{2}, 2J_5 - \frac{J_2 + J_3}{2}, 2J_6 - \frac{J_3 + J_1}{2} \right\} \\
& \leq \min \{J_1, J_2, J_3\}. \quad (13)
\end{aligned}$$

One can show that this estimate is always better than the one using the hierarchical basis. It provides two conditions on the geometrical validity of the triangle: a *sufficient* condition (if $\min\{J_1, J_2, J_3, 2J_4 - \frac{J_1+J_2}{2}, 2J_5 - \frac{J_2+J_3}{2}, 2J_6 - \frac{J_3+J_1}{2}\} > 0$, the element is valid) and a *necessary* condition (if $\min\{J_1, J_2, J_3\} < 0$, the element is invalid). However, these two conditions are sometimes insufficient to determine the validity of the element, as the bound (13) is often not sharp enough (having $\min\{2J_4 - \frac{J_1+J_2}{2}, 2J_5 - \frac{J_2+J_3}{2}, 2J_6 - \frac{J_3+J_1}{2}\} < 0$ does not imply that the element is invalid).

A sharp necessary and sufficient condition on the geometrical validity of an element can be achieved in a general way by refining the Bézier estimate adaptively so as to achieve any prescribed tolerance—and thus provide bounds as sharp as necessary for a given application.

4 Adaptive Jacobian Bounds for Arbitrary Curvilinear Finite Elements

In order to explain the adaptive bound computation let us first focus on the one-dimensional case, for “line” finite elements. Since Bézier functions can be generated for all types of common elements (triangles, quadrangles, tetrahedra, hexahedra and prisms), the generalization to 2D and 3D elements will be straightforward.

4.1 The One-Dimensional Case

In 1D the Bézier functions are the Bernstein polynomials:

$$\mathcal{B}_k^{(n)}(\xi) = \binom{n}{k} (1 - \xi)^{n-k} \xi^k \quad (\xi \in [0, 1] ; k = 0, \dots, n) \quad (14)$$

where $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ is the binomial coefficient. The Bézier interpolation requires $n + 1$ control values b_i . We have

$$J(\xi) = \sum_{k=0}^{N_n} \mathcal{B}_k^{(n)}(\xi) b_k. \quad (15)$$

Bernstein-Bézier functions have the nice following properties : (i) they form a partition of unity which means that $\sum_{k=0}^n \mathcal{B}_k^{(n)}(\xi) = 1$ for all $\xi \in [0, 1]$ and (ii) they are positive which means that $\mathcal{B}_k^{(n)}(\xi) \geq 0$ for all $\xi \in [0, 1]$. This leads to the well known property of Bézier interpolations:

$$\min_{\xi \in [0,1]} J(\xi) \geq b_{\min} = \min_i b_i \quad \text{and} \quad \max_{\xi \in [0,1]} J(\xi) \leq b_{\max} = \max_i b_i. \quad (16)$$

Moreover, they always present control values that are values of the interpolated function. Let assume they are ordered at the K_f first indices, we have

$$\min_{\xi \in [0,1]} J(\xi) \leq \min_{i < K_f} b_i \quad \text{and} \quad \max_{\xi \in [0,1]} J(\xi) \geq \max_{i < K_f} b_i. \quad (17)$$

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

$$\mathbf{T}_{\mathcal{B} \rightarrow \mathcal{L}}^{(n)} = \begin{bmatrix} \mathcal{B}_0^{(n)}(\xi_0) & \dots & \mathcal{B}_n^{(n)}(\xi_0) \\ \mathcal{B}_0^{(n)}(\xi_1) & \dots & \mathcal{B}_n^{(n)}(\xi_1) \\ \vdots & \ddots & \vdots \\ \mathcal{B}_0^{(n)}(\xi_n) & \dots & \mathcal{B}_n^{(n)}(\xi_n) \end{bmatrix}.$$

The inverse transformation is $\mathbf{T}_{\mathcal{L} \rightarrow \mathcal{B}}^{(n)} = \mathbf{T}_{\mathcal{B} \rightarrow \mathcal{L}}^{(n) -1}$ and from the expression of the interpolation of the Jacobian (15), we can write

$$\begin{aligned} J &= \mathbf{T}_{\mathcal{B} \rightarrow \mathcal{L}}^{(n)} B \\ B &= \mathbf{T}_{\mathcal{L} \rightarrow \mathcal{B}}^{(n)} J, \end{aligned} \quad (18)$$

where B and J are the vectors containing respectively the b_i 's and the J_i 's.

4.2 Adaptive Subdivision

Let assume that the domain is divided into Q parts and that the q^{th} of them interpolates the Jacobian on $[a, b]$ ($a < b$). Then, the variable ξ varies in this interval while the new one varies from 0 to 1 and the new interpolation must verify

$$J^{[q]}(\xi^{[q]}) = \sum_{k=0}^{N_n} \mathcal{B}_k^{(n)}(\xi^{[q]}) b_k^{[q]} = \sum_{k=0}^{N_n} \mathcal{B}_k^{(n)}(\xi(\xi^{[q]})) b_k \quad (\xi^{[q]} \in [0, 1]), \quad (19)$$

with $\xi(\xi^{[q]}) = a + (b - a) \xi^{[q]}$. Considering the nodes $\xi_k^{[q]}$ such that $\xi_k^{[q]} = \xi_k$ ($k = 0, \dots, n$) (i.e., such that they are ordered like the sampling points), the expression (19) reads

$$\mathbf{T}_{\mathcal{B} \rightarrow \mathcal{L}}^{(n)} B^{[q]} = \begin{bmatrix} \mathcal{B}_0^{(n)}(a + (b - a) \xi_0) & \dots & \mathcal{B}_n^{(n)}(a + (b - a) \xi_0) \\ \mathcal{B}_0^{(n)}(a + (b - a) \xi_1) & \dots & \mathcal{B}_n^{(n)}(a + (b - a) \xi_1) \\ \vdots & \ddots & \vdots \\ \mathcal{B}_0^{(n)}(a + (b - a) \xi_n) & \dots & \mathcal{B}_n^{(n)}(a + (b - a) \xi_n) \end{bmatrix} B = \mathbf{T}_{\mathcal{B} \rightarrow \mathcal{L}}^{(n) [q]} B,$$

where $B^{[q]}$ is the vector containing control values of the related subdomain. This implies that

$$B^{[q]} = \begin{bmatrix} \mathbf{T}_{\mathcal{L} \rightarrow \mathcal{B}}^{(n)} & \mathbf{T}_{\mathcal{B} \rightarrow \mathcal{L}}^{(n) [q]} \end{bmatrix} B = \mathbf{M}^{[q]} B. \tag{20}$$

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

$$b'_{\min} = \min_{i,q} b_i^{[q]} \leq J_{\min} \leq \min_{i < K_{f,q}} b_i^{[q]} \tag{21}$$

and

$$\max_{i < K_{f,q}} b_i^{[q]} \leq J_{\max} \leq b'_{\max} = \max_{i,q} b_i^{[q]}. \tag{22}$$

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 (16)-(17) is not sharp enough ($J_{\min} \in [-2.5, 1]$). After one subdivision, the Jacobian is proved to be positive on the second subdomain. The first subdomain is thus subdivided once more, which proves the validity. In practice, as will be seen in Section 5, a few levels of refinement lead to the desired accuracy. The convergence of the subdivision can be proven to be quadratic [8, 9].

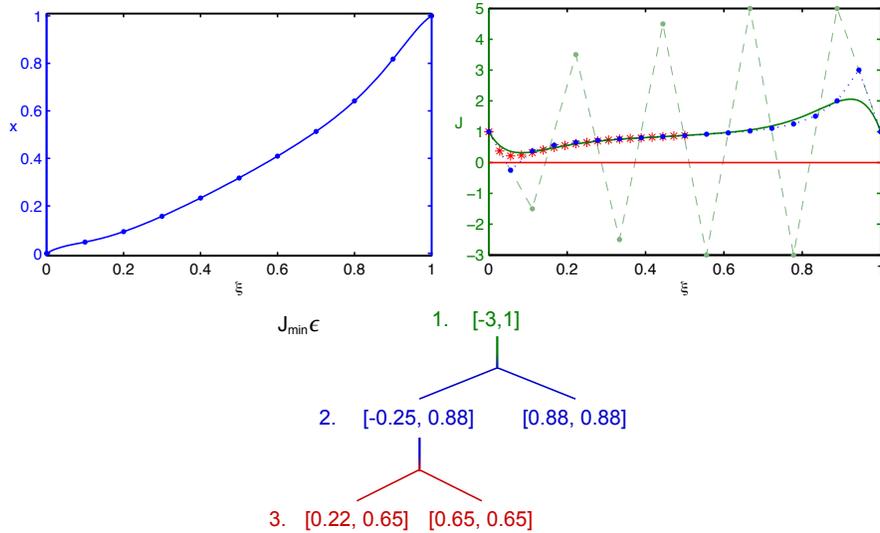


Fig. 2. Top left: One-dimensional element mapping $x(\xi)$. Top right: Exact Jacobian $J(\xi)$ (solid green), control values on the original control points (dashed green) and two adaptive subdivisions (blue and red). Bottom: Estimates of J_{\min} at each step in the adaptive subdivision process.

4.3 Extension to Higher Dimensions

The extension of the method to higher dimensions is straightforward, provided that Bézier functions can be generated and that a subdivision scheme is available: Jacobians J are polynomials of ξ, η in 2D and of ξ, η, ζ in 3D.

For high order triangles, the Bézier triangular polynomials are defined as

$$\mathcal{T}_{i,j}^{(p)}(\xi, \eta) = \binom{p}{i} \binom{p-i}{j} \xi^i \eta^j (1-\xi-\eta)^{p-i-j} \quad (i+j \leq p).$$

It is possible to interpolate any polynomial function of order at most p on the unit triangle $\xi > 0, \eta > 0, \xi + \eta < 1$ as an expansion into Bézier triangular polynomials. Recalling that, for a triangle at order p , its Jacobian $J(\xi, \eta)$ is a polynomial in ξ and η at order at most $n = 2(p-1)$, we can write

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

It is indeed possible to compute J in terms of Lagrange polynomials

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

where the J_i are the Jacobians calculated at Lagrange points. It is then easy to find a transformation matrix $T_{\mathcal{LB}}^n$ such that

$$B = T_{\mathcal{LB}}^n J,$$

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

$$T_{\mathcal{LB}}^2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -1/2 & -1/2 & 0 & 2 & 0 & 0 \\ 0 & -1/2 & -1/2 & 0 & 2 & 0 \\ -1/2 & 0 & -1/2 & 0 & 0 & 2 \end{pmatrix}, \quad (23)$$

which directly provides the estimate (13).

Other element shapes can be treated similarly. For quadrangles, tetrahedra, prisms and hexahedra, the Bézier are functions respectively:

$$\mathcal{Q}_{i,j}^{(p)}(\xi, \eta) = \mathcal{B}_i^{(p)}(\xi) \mathcal{B}_j^{(p)}(\eta) \quad (i \leq p, j \leq p),$$

$$\mathcal{T}_{i,j,k}^{(p)}(\xi, \eta, \zeta) = \binom{p}{i} \binom{p-i}{j} \binom{p-i-j}{k} \xi^i \eta^j \zeta^k (1-\xi-\eta-\zeta)^{p-i-j-k} \quad (i+j+k \leq p),$$

$$\mathcal{P}_{i,j,k}^{(p)}(\xi, \eta, \zeta) = \mathcal{T}_{i,j}^{(p)}(\xi, \eta) \mathcal{B}_k^{(p)}(\zeta) \quad (i + j \leq p, k \leq p)$$

and

$$\mathcal{H}_{i,j,k}^{(p)}(\xi, \eta, \zeta) = \mathcal{B}_i^{(p)}(\xi) \mathcal{B}_j^{(p)}(\eta) \mathcal{B}_k^{(p)}(\zeta) \quad (i \leq p, j \leq p, k \leq p).$$

Matrices of change of coordinates can then be computed inline for every polynomial order, and bounds of Jacobians computed accordingly. In all cases the subdivision scheme works exactly in the same way as for lines.

4.4 Implementation

As mentioned in Section 2, the Jacobian 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 defined precision.

In both cases the same operations are executed on each element. First, the Jacobian is sampled on a determined number of points N_s , equal to the dimension of the Jacobian space, and so to the number of Bézier functions. Second, Bézier values are computed. Then adaptive subdivision is executed if necessary. Algorithm 2 shows in pseudo-code the algorithm used to determine whether the Jacobian of the element is everywhere positive or not.

Algorithm 2: Check if an element is valid or invalid

Input: a pointer to an element.

Output: *true* if the element is valid, *false* if the element is invalid

```

1 set sampling points  $P_i, i = 1, \dots, N_s$ ;
2 compute Jacobians  $J_i$  at points  $P_i$ ;
3 for  $i = 1$  to  $N_s$  do
4   | if  $J_i \leq 0$  then return false;

5 compute Bézier coefficients  $b_i, i = 1, \dots, N_s$  using (18);
6  $i = 1$ ;
7 while  $i \leq N_s$  and  $b_i > 0$  do
8   |  $i = i + 1$ ;
9 if  $i > N_s$  then return true;
10 call algorithm 3 with  $b_i$  as arguments and return its output;
```

Algorithm 3 can 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 (line 4, algo 3). However, 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 which require subdivision in order to be detected is usually small. Note that we may also want to find, for example, all the elements for

Algorithm 3: Compute the control values of the subdivisions

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

```

1 compute new Bézier coefficients  $b_i^{[q]}$ ,  $q = 1, \dots, Q$  as in equation (20);
2 for  $q = 1$  to  $Q$  do
3   for  $i = 1$  to  $K_f$  do
4     if  $b_i^{[q]} \leq 0$  then return false;
5 for  $q = 1$  to  $Q$  do
6    $i = 1$ ;
7   while  $i \leq N_s$  and  $b_i^{[q]} > 0$  do
8      $i = i + 1$ ;
9   if  $i \leq N_s$  then
10    call algorithm 3 with  $b_i^{[q]}$  as arguments and store output;
11    if output = false then return false;
12 return true;
```

which the Jacobian 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 2).

Another possible improvement is to relax the condition of rejection. We could accept elements for which all control values are positive but reject an element as soon as we find a Jacobian value smaller than a defined percent of the average Jacobian. The computational gain can be significant, since elements that were classified as good and which needed a lot of subdivisions (and have a Jacobian close to zero) will be instead rapidly be detected as invalid.

More interestingly, the computation of sampled Jacobians and the computation of Bézier control values in algorithm 2 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 using the BLAS3 approach is implemented in the open source mesh generator Gmsh [4] as a the `AnalyseCurvedMesh` plugin, and was used for all the tests presented in the next section.

5 Numerical Results

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

The points at which we sample the Jacobian 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 on the points computed for an element at order n (the order of the Jacobian) and then compute the desired Jacobian values by a matrix-vector product, just like in our own adaptive method.

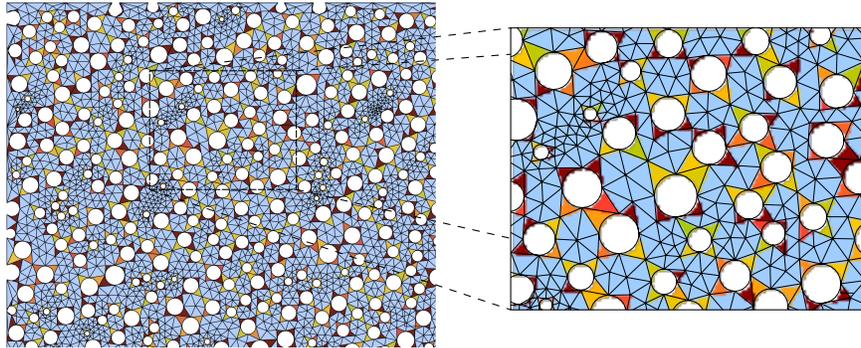


Fig. 3. Two-dimensional mesh with sixth order triangles; 23.6% of the elements are curved. The straight element are in blue and the invalid are in dark red. The other are colored in function of the distortion. They are green if they are nearly straight ($J_{\min}/J_{\max} \simeq 1$) and rather light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

Let us now examine the use of the adaptive Jacobian 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 [10]. The quadratic triangular mesh is generated as follows. We first generate a straight sided mesh (see Figure

	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

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.

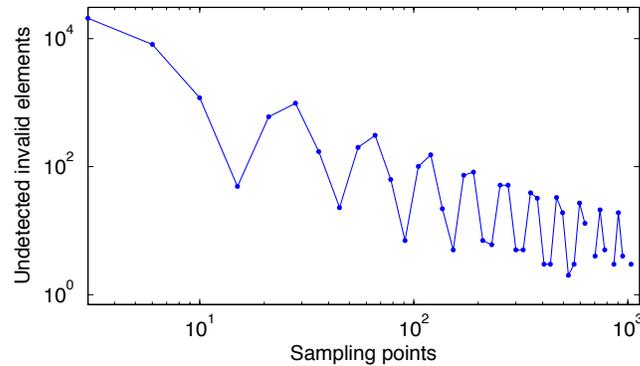


Fig. 4. Number of undetected invalid elements using brute-force sampling of the Jacobian. The three data points not displayed correspond to the correct result, i.e., when no invalid triangle is left undetected.

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 bound computation allowed to detected all invalid elements (the worst distortion that was observed was $\delta = -4.49702$). After optimization, the final mesh is composed of elements that have a distortion $\delta > 0.1$.

Finally, the same procedure is applied to the meshing of the STEP model of a rotor. After generating a first order mesh (Figure 6(a)) and snapping

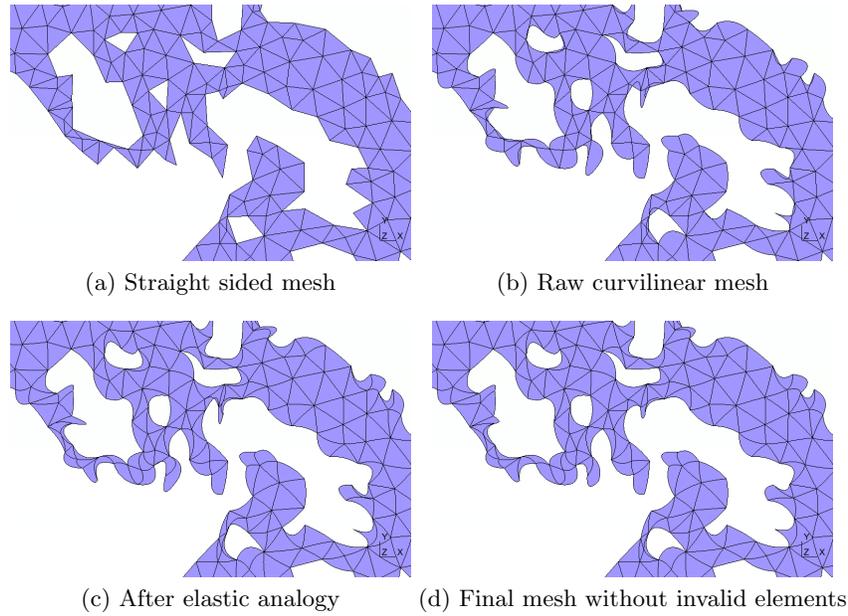


Fig. 5. The four stages of the curvilinear mesh procedure for the world ocean, meshed with second order triangles.

the high-order vertices on the geometrical model (Figure 6(b)), the adaptive Jacobian bound computation allowed to pinpoint all the invalid elements. The final, locally optimized mesh is displayed in Figure 6(c).

6 Conclusion

In this paper we presented a way to compute accurate bounds on Jacobians of curvilinear finite elements, based on the efficient expansion of these Jacobians 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. Triangles, quadrangles, tetrahedra, prisms and hexahedra can be analyzed using the same algorithm, which is available in the open source mesh generator Gmsh. Numerical tests show that the method is robust, and a user-defined error tolerance permits to adjust the accuracy vs. computational time ratio.

Acknowledgement

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

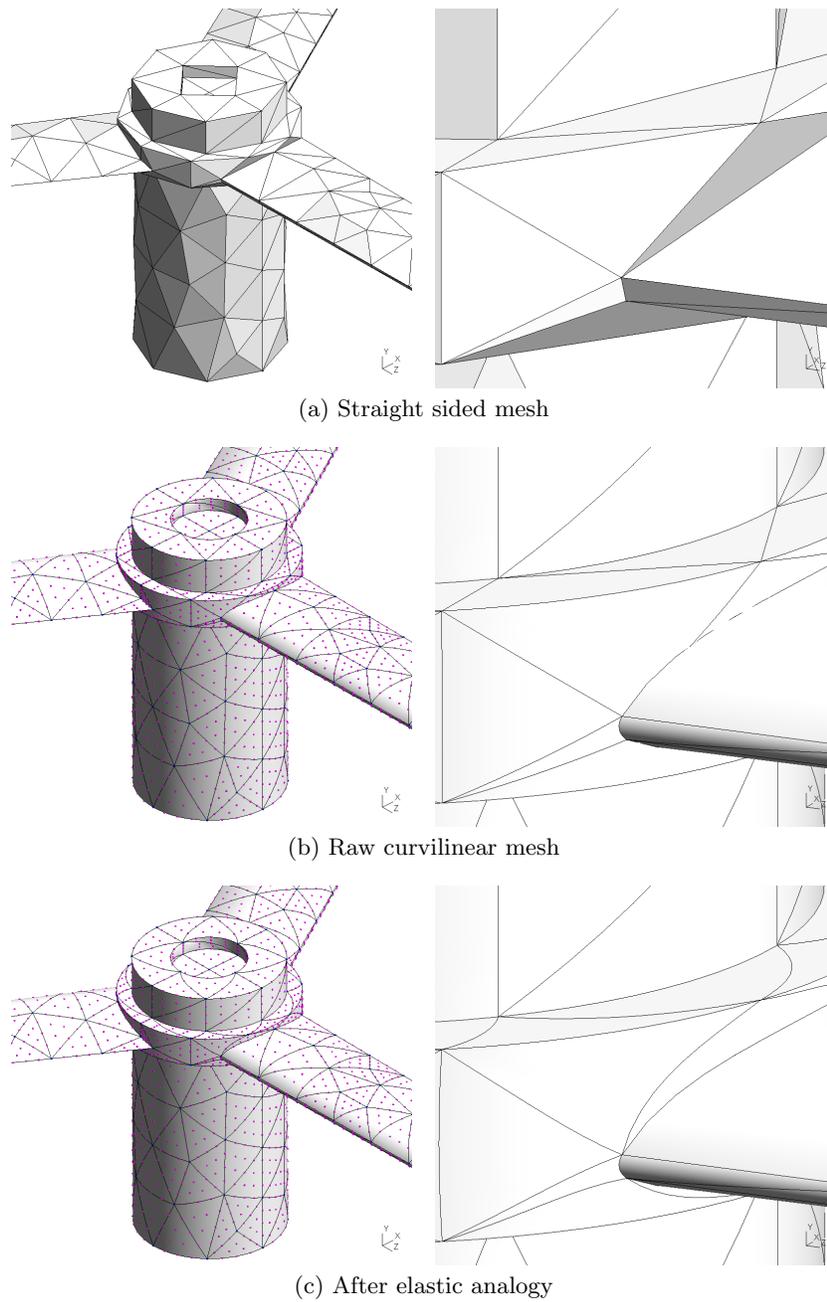


Fig. 6. Curvilinear mesh of a rotor using fourth-order curved triangles.

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.

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. Chevaugéon, 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. 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.
5. T. Hughes. *The Finite Element Method*. Dover, 2003.
6. Ivo Babuška, B. Szabò, and R. L. Actis. Hierarchic models for laminated composites. *International Journal for Numerical Methods in Engineering*, 33:503–535, 1992.
7. G. E. Farin. *Curves and surfaces for CAGD: a practice guide*. Morgan-Kaufmann, 2002.
8. 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.
9. E. Cohen and L. L. Schumacker. Rates of convergence of control polygons. *Computer Aided Geometric Design*, 2:229–235, 1985.
10. J. Lambrechts, R. Comblen, V. Legat, C. Geuzaine, and J.-F. Remacle. Multi-scale mesh generation on the sphere. *Ocean Dynamics*, 58:461–473, 2008.