An efficient automatic mesh generation algorithm for planar isogeometric analysis using high-order rational Bézier triangles

Elias Saraiva Barroso[a], John Andrew Evans[b], Joaquim Bento Cavalcante Neto[a], Creto Augusto Vidal[a], Evandro Parente Junior[c]

[a] *Computer Graphics, Virtual Reality and Animation Group (CRAb), Universidade Federal do Ceará, Fortaleza, Brazil*
[b] *Aerospace Engineering Sciences, University of Colorado at Boulder, United States*
[c] *Laboratório de Mecânica Computacional e Visualização, Universidade Federal do Ceará, Fortaleza, Brazil*

Abstract

Isogeometric Analysis (IGA) is a numerical method that is receiving increasing attention in the last decade. The main goal of IGA is to closely couple geometric modeling with numerical analysis. To that end, in IGA both components use the same geometric representation, e.g., Bézier and NURBS curves and surfaces. However, in many cases, the geometric representation in a CAD system cannot be directly employed in numerical simulation, as only the boundary of the geometry is parametrized. In these cases, an interior parametrization must be constructed before performing isogeometric analysis. This paper presents an algorithm for the generation of unstructured geometrically exact meshes composed of high-order rational Bézier triangles, and applies it to plane models described by NURBS curves. The proposed algorithm respects input discretization and is capable of generating high quality coarse meshes even when high curvature segments are considered. An efficient high-order smoothing step is employed to avoid tangled elements and to improve element quality. The proposed algorithm attains superior performance when compared to a well-known algorithm in the literature, and performs well in the case of complex geometries. High quality meshes were obtained in all examples analyzed. Furthermore, our implementation is efficient, written in C++, and is available as an open-source software.

Keywords: High-order mesh generation, Bézier triangles, Isogeometric Analysis

1. Introduction

Isogeometric Analysis (IGA), proposed by Hughes et al. [1] and, afterwards, studied by several researchers, is a numerical method whose main goal is to use the same underlying mathematical representation for both geometric modeling and numerical analysis. In this context, the geometric models used in *Computer Aided Design* (CAD), e.g., Bézier and Non-Uniform Rational B-Splines (NURBS) curves and surfaces, are the same models used in the formulations of numerical simulators. Thus, the errors caused by the approximate representation of the domain's geometry in numerical methods such as the Finite Element Method (FEM) disappear.

IGA was initially formulated for geometries defined by NURBS [1–4]. Discretization procedures similar to existing p and h refinements for FEM [5] are applied using well-known algorithms from the field of Geometric Modeling. Besides, it uses models with

Email addresses: `elias.barroso@gmail.com` (Elias Saraiva Barroso), `john.a.evans@colorado.edu` (John Andrew Evans), `joaquimb@dc.ufc.br` (Joaquim Bento Cavalcante Neto), `cvidal@dc.ufc.br` (Creto Augusto Vidal), `evandro@ufc.br` (Evandro Parente Junior)

high continuity, thanks to a new discretization strategy that does not exist in the classical finite element formulations: k-refinement [6]. Notice that other geometric entities capable of performing localized refinements that cannot be achieved with NURBS are also used in IGA, e.g., hierarchical B-Splines [7–9], T-Splines [10–12], LR B-Splines [13–15], PHT-Splines [16, 17], and THB-Splines [18, 19].

Despite the advantages of these approaches, they still present limitations when considering the Design-through-Analysis paradigm present in numerical simulation programs. Geometric modelers do not always provide all the input information that numerical analysis programs need. Most CAD systems use the Boundary Representation (B-Rep) paradigm [20, 21], where only the boundary of the models is explicitly parametrized. However, the application of isogeometric formulations for structural or thermal analysis requires an adequate parametrization of the analysis domain (e.g., NURBS patches), which is difficult to obtain in an automatic fashion for complex geometries described using B-Rep.

Another relevant issue arises when trimmed models are considered. In this case, special integration schemes are required due to the models' complex geometry description [22]. The above problems are known in the numerical analysis area in the context of the Finite Element Method, and are solved with the use of mesh generation techniques. There are papers in the literature addressing the generation of NURBS and T-Splines models for geometries described using B-Reps, both in 2D and 3D cases [23–29]. However, those approaches are not robust in some important aspects, as discussed in [30, 31], such as: to handle complex geometries, e.g., objects with arbitrary genus or objects with large size variation; to be automatic, i.e., no user interaction is required; to guarantee the exact geometric representation; to provide suitable parametrization for the analysis.

On the other hand, the use of rational Bézier triangles and tetrahedra for unstructured isogeometric mesh generation has been explored as an alternative to couple IGA with B-Reps. The generation of high-order Bézier triangles is described in [32]. A triangle mesh is used to construct a rational Triangular Bézier Spline (rTBS) with high continuity and exact geometry. The extension of this work to the 3D case is presented in [33], where meshes with Bézier tetrahedra are considered, and rTBS are generated in the case of trimmed surfaces. Triangular Bézier Splines are also used for shell analysis using the Kirchhoff-Love C^1 formulation [34].

Automatic generation of Bézier triangle meshes for plane problems is studied in [30]. The paper focuses on the reuse of existing linear mesh generation packages to create high-order C^0 meshes with exact geometry. The same meshing technique is used in the context of plate analysis [35] and shape optimization [36]. The extension of that mesh generation approach to Bézier tetrahedra, hexahedra, wedges, and pyramids is presented by Engvall and Evans [31], where Bézier projection is used for the reconstruction of NURBS and T-Spline surfaces using Bézier triangles or quadrilaterals. The authors also discuss the use of super parametric elements capable of maintaining exact geometry, a relevant feature since there is no guarantee that Bézier triangles can represent NURBS and T-Spline surfaces of the same degree. A study on quality metrics for rational Bézier triangles and tetrahedrons is presented in [37, 38].

There are also works addressing the generation of high-order meshes in the context of FEM, in the generation and adaptivity of high-order meshes [39–41], in the study of quality metrics and mesh optimization [42–46], and in the field of moving meshes [47, 48]. Quadratic Bézier triangles are also used in the context of moving meshes [49] and in elastostatic and elastodynamic applications [50].

The purpose of this paper is to present an algorithm for automatic generation of high-order isogeometric meshes of rational Bézier triangles for plane models described using B-Rep. The proposed algorithm respects the model's boundary parametrization and does not require additional refinements. This requirement is important when implicit mesh compatibility is to be maintained in models with multiple regions, allowing the use of different mesh generation algorithms with different elements (e.g., Bézier surface patches and triangles), and also when remeshing is applied, such as in crack evolution simulations [51]. Furthermore, the sizing function used in the algorithm allows a smooth variation of element sizes, an important feature in practical problems where there is a high variation in the size of the input discretization.

The procedures described here are generally similar to other works in the literature related to rational Bézier meshes [30], but some contributions are presented as follows. The curved segments are considered in the evaluation of the sizing function used during the linear meshing step, instead of using straight segments. Moreover, topology modifications are adopted to remove singularities from high-order elements. Those singularities possibly occur when an element is adjacent to multiple input curved segments, and cannot be removed without topology changes.

A high-order smoothing is proposed to avoid invalid elements and improve mesh quality. That procedure is based on a mesh deformation scheme via linear elasticity and weighted smoothing applied to rational models. A direct application of those steps in all mesh elements has a high computational cost. To avoid that, those steps are employed locally in a set of disjoint groups of elements, maintaining computational efficiency. We are not aware of any work that has implemented a similar smoothing step on unstructured IGA meshes composed of rational Bézier triangles. Nevertheless, some works have employed mesh regularization in high-order FEM, using deformation schemes [52–54] and mesh optimization [41, 42, 45].

The proposed algorithm is especially useful when high-order meshes with fewer elements are desired. The other alternatives in the literature use automatic discretization strategies, which can lead to excessively fine meshes when the geometry has high-curvature. In such cases, despite improving the accuracy and effectiveness of the numerical simulation, automatic discretization increases the computational cost. The proposed algorithm implementation is available in *PMGen* (Plane Mesh Generator), an open-source program written in C++ language and available at `https://github.com/lmcv-ufc/PMGen`. Its graphical user interface has functionalities for geometric modeling, discretization of isogeometric models, and visualization of the generated meshes and of the quality metrics discussed in this work. Moreover, *PMGen* is capable of generating meshes automatically using curve subdivision procedures, even when the geometry is complex. All files related to the examples presented in this paper are available in the repository.

The remainder of this paper is structured as follows. In Section 2, we discuss the necessary geometric modeling concepts. In Section 3, we present the proposed algorithm and the quality metrics. In Section 3.3, we discuss the high-order smoothing step. In Section 5, we present examples in which the proposed algorithm was applied. Finally, in Section 6, we present the concluding remarks.

2. Geometric Modeling

The 2D models studied in this work are described by NURBS curves using the Boundary Representation paradigm (B-Rep). The topological information consists of a set of vertices and edges, organized in loops with consistent orientation (e.g., regions, holes and constraints). The geometry of each edge is given by a NURBS curve whose endpoints correspond to its corresponding edge vertices. Fig. 1 illustrates examples of B-Rep models.

2.1. Rational Bézier Curve

A rational Bézier curve of degree p is defined as the linear combination of a set of control points ($\mathbf{p}_i$) in which the corresponding combination coefficients $R_i^p(r)$ (blending coefficients) for a given parameter r are rational functions, i.e.,

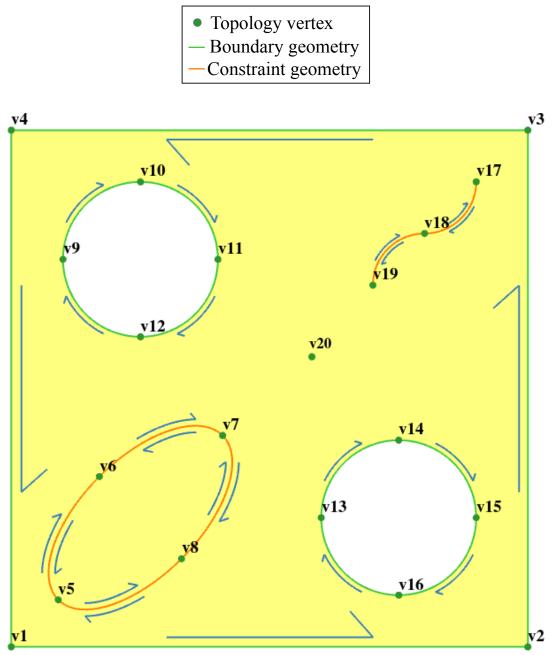
$$\mathbf{C}(r) = \sum_{i=0}^{p} R_i^p(r)\, \mathbf{p}_i. \tag{1}$$

The rational blending functions are defined as

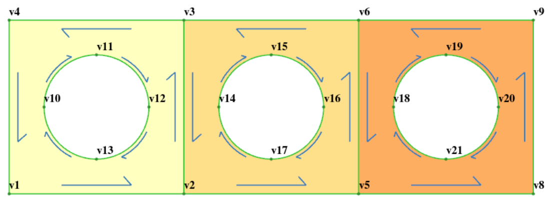
$$R_i^p(r) = \frac{B_i^p(r)\, w_i}{W(r)}, \text{ with } W(r) = \sum_{\hat{i}=0}^{p} B_{\hat{i}}^p(r)\, w_{\hat{i}}, \tag{2}$$

where w_i is the weight associated with the control point $\mathbf{p}_i$, and $B_{\hat{i}}^p(r)$ is the corresponding Bernstein polynomial

$$B_i^p(r) = \frac{i!}{p!\,(p-i)!}\, r^p\,(1-r)^{p-i}. \tag{3}$$



(a) A plate with circular holes and constraints.



(b) A sheet with holes. Three regions are used to set different material properties.

Figure 1: Examples of B-Rep models.

2.2. NURBS

NURBS are widely used in CAD systems since they are capable of representing diverse types of curves and surfaces [55]. A NURBS curve of degree p is written as

$$\hat{\mathbf{C}}(r) = \sum_{i=1}^{m} \hat{R}_i^p(r)\,\mathbf{p}_i, \tag{4}$$

where m is the number of NURBS functions and $\hat{R}_i^p(r)$ are the NURBS blending functions

$$\hat{R}_i^p(r) = \frac{N_i^p(r)\,w_i}{\hat{W}(r)}, \text{ with } \hat{W}(r) = \sum_{\hat{i}=1}^{m} N_{\hat{i}}^p(r)\,w_{\hat{i}}. \tag{5}$$

The B-Spline base functions (N_i^p) require a knot vector, which defines a set of non-negative and non-decreasing parametric values delimited over the parametric range $[r_1, r_{m+p+1}]$, where the curve is defined. Given the knot vector $\mathbf{r} = [r_1, r_2, \ldots, r_{m+p+1}]$, the B-Spline basis functions are defined by the Cox-de Boor recursion formula [2]:

$$N_i^0(r) = \begin{cases} 1, & r_i \le r < r_{i+1} \\ 0, & \text{otherwise,} \end{cases}$$

$$N_i^p(r) = \frac{r - r_i}{r_{i+p} - r_i}\, N_i^{p-1}(r) + \frac{r_{i+p+1} - r}{r_{i+p+1} - r_{i+1}}\, N_{i+1}^{p-1}(r), \tag{6}$$

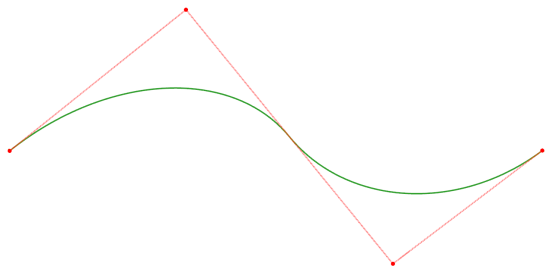
where the number of knots nkin the knot vector is related to the number of control points m and to the degree p of the NURBS through

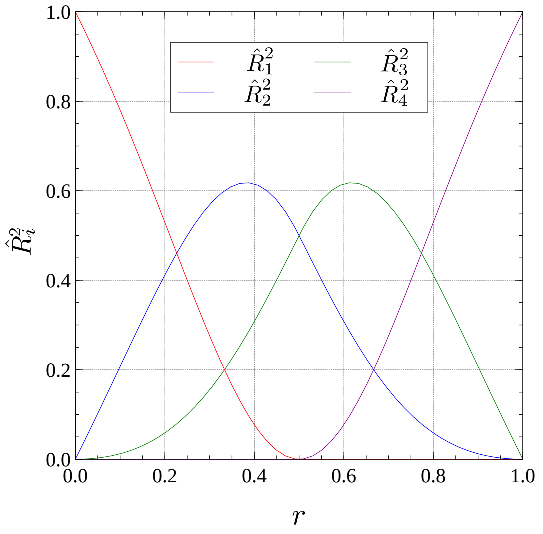
$$nk = m + p + 1. \tag{7}$$

The B-Spline basis has important properties: linear independence; non-negativity ($N_i^p(r) \ge 0$); partition of unity ($\sum_{i=1}^m N_i^p(r) = 1$) and compact support ($N_i^p(r) = 0$ if $r \notin [r_i, r_{i+p+1}]$). The continuity of $\hat{C}(r)$ curve at a given knot j is C^{p-k_j}, where $k_j \le p$ is the multiplicity of knot j in the knot vector. Notice that such properties are also valid for rational basis functions. Fig. 2 shows a cubic NURBS curve with $\mathbf{r} = [0\ 0\ 0\ 0.33\ 0.66\ 1\ 1\ 1]$ and $\mathbf{w} = [1\ 0.5\ 0.5\ 1]$. In this example, the parametric coordinates of knots $1, 2$ and 3 are all equal to 0 (multiplicity equal to 3), which forces the curve to pass by control point $\mathbf{p}_1$. Similarly, the parametric coordinates of knots $6, 7$ and 8 are all equal to 1 (multiplicity equal to 3), which forces the curve to pass by control point $\mathbf{p}_4$.

The representation of a NURBS entity can be changed while preserving its geometry and parametrization. The Knot Insertion and Knot Removal algorithms modify the knot vector, while the Degree Elevation and Degree Reduction algorithms modify the curve degree. These operations are used in the context of geometric modeling [2] and also for applying refinements to isogeometric models [56].

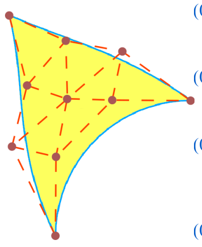
Notice that a NURBS curve can also be represented by a set of rational Bézier curves, and their control points can be evaluated by the Bézier extraction procedure [2, 57, 58].

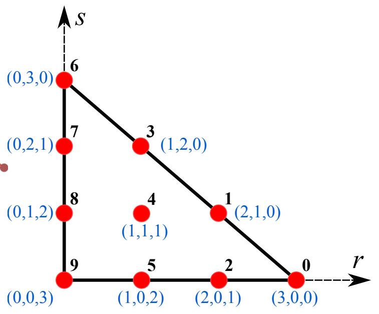

(a) NURBS curve and control polygon.



(b) Bases $\hat{R}_i^p$.

Figure 2: Cubic NURBS curve ($\mathbf{w} = [1\ 0.5\ 0.5\ 1]$).


(a) Physical space.



(b) Parametric space.

Figure 3: Cubic Bézier triangle.

2.3. Bézier Triangles

Bézier triangles are bivariate surfaces defined by a set of control points, arranged in a triangular structure, as shown in Fig. 3. Those triangular surfaces are evaluated by

$$\mathbf{T}(r,s) = \sum_{i+j+k=p} \hat{B}^p_{i,j,k}(r,s)\,\mathbf{p}_{i,j,k}, \tag{8}$$

where r and s are the parametric coordinate, p is the surface degree, $\mathbf{p}_{i,j,k}$ are the control points and $\hat{B}^p_{i,j,k}$ are the bivariate Bernstein polynomials. The triple index i,j,k in basis functions and control points satisfies the constraints $i+j+k=p$ and $i,j,k \geq 0$, as illustrated in Fig. 3 (b). Moreover, the conversion from the triple indexes (i,j,k) to the single index a (depicted in Fig. 3 (b)) is given by

$$a = p - i - j + (p-i+1)(p-i)/2. \tag{9}$$

The number of basis functions (i.e., control points) is computed as

$$N_{cp} = (p+1)(p+2)/2. \tag{10}$$

The bivariate Bernstein polynomials are defined as

$$\hat{B}^p_{i,j,k}(r,s) = \frac{!p}{!i\,!j\,!k} r^i s^j (1-r-s)^k, \tag{11}$$

and satisfy the following recursion formula [59]:

$$\begin{aligned}\hat{B}^p_{i,j,k}(r,s) =& r\,\hat{B}^{p-1}_{i-1,j,k}(r,s) + \hat{B}^{p-1}_{i,j-1,k}(r,s)\\ &+ (1-r-s)\,\hat{B}^{p-1}_{i,j,k-1}(r,s),\end{aligned} \tag{12}$$

where it is assumed that $\hat{B}^0_{0,0,0} = 1$ and $\hat{B}^p_{i,j,k} = 0$ for $i,j,k < 0$. This recursive definition, which resembles *de Casteljau's* algorithm applied to Bézier triangles, is numerically stable [59, 60]. The bivariate Bernstein polynomials also have linear independence, non-negativity, and partition of unity properties.

The first partial derivatives of the bivariate Bernstein polynomials are:

$$\begin{aligned}\frac{\partial}{\partial r}\hat{B}^p_{i,j,k} &= p\,(\hat{B}^{p-1}_{i-1,j,k} - \hat{B}^{p-1}_{i,j,k-1}),\\ \frac{\partial}{\partial s}\hat{B}^p_{i,j,k} &= p\,(\hat{B}^{p-1}_{i,j-1,k} - \hat{B}^{p-1}_{i,j,1-k}).\end{aligned} \tag{13}$$

Efficient algorithms for the evaluation of bivariate Bernstein polynomials and their derivatives are described in Appedix A.1.

2.3.1. Rational Bézier Triangles

The rational version of a Bézier triangle is defined in an analogous manner to the rational tensor product Bézier and NURBS surfaces by also assigning weights to each control point. Those rational entities are important because they can represent quadrics such as circles, ellipses, parabolas, and hyperbolas exactly [59]. The rational Bézier triangle is defined by

$$\mathbf{T}_r(r,s) = \sum_{a=0}^{N_{cp}-1} \tilde{R}_a^p(r,s)\,\mathbf{p}_a, \tag{14}$$

where a is the single index presented in Fig. 3 (b) and the rational Bernstein functions are

$$\tilde{R}_a^p(r,s) = \frac{\hat{B}_a^p(r,s)\,w_a}{\tilde{W}(r,s)}, \tag{15}$$

where $\tilde{W}$ is the weighting function

$$\tilde{W}(r,s) = \sum_{a=0}^{N_{cp}-1} \hat{B}_a^p(r,s)\,w_a. \tag{16}$$

Basis functions derivatives are evaluated using the quotient rule:

$$\begin{aligned}\frac{\partial \tilde{R}_a^p}{\partial r} &= w_a \frac{\tilde{W}\,\dfrac{\partial \hat{B}_a^p}{\partial r} - \dfrac{\partial \tilde{W}}{\partial r}\,\hat{B}_a^p}{\tilde{W}^2},\\ \frac{\partial \tilde{R}_a^p}{\partial s} &= w_a \frac{\tilde{W}\,\dfrac{\partial \hat{B}_a^p}{\partial s} - \dfrac{\partial \tilde{W}}{\partial s}\,\hat{B}_a^p}{\tilde{W}^2},\end{aligned} \tag{17}$$

and the weighting function derivatives are evaluated by

$$\begin{aligned}\frac{\partial \tilde{W}}{\partial r} &= \sum_{a=0}^{N_{cp}-1} \frac{\partial \hat{B}_a^p}{\partial r}\,w_a,\\ \frac{\partial \tilde{W}}{\partial s} &= \sum_{a=0}^{N_{cp}-1} \frac{\partial \hat{B}_a^p}{\partial s}\,w_a.\end{aligned} \tag{18}$$

The Jacobian matrix of the plane rational Bézier triangle is given by

$$J = \begin{bmatrix} \sum_{a=0}^{N_{cp}-1} \dfrac{\partial \tilde{R}_a^p}{\partial r}\,x_a & \sum_{a=0}^{N_{cp}-1} \dfrac{\partial \tilde{R}_a^p}{\partial s}\,x_a \\ \sum_{a=0}^{N_{cp}-1} \dfrac{\partial \tilde{R}_a^p}{\partial r}\,y_a & \sum_{a=0}^{N_{cp}-1} \dfrac{\partial \tilde{R}_a^p}{\partial s}\,y_a \end{bmatrix} \tag{19}$$

where

$$\mathbf{p}_a = (x_a, y_a). \tag{20}$$

It is worth mentioning that these derivatives are required in many numerical problems, such as in the evaluation of stiffness matrices in linear elasticity problems.

3. Proposed mesh generation algorithm

The mesh generation algorithm proposed in this work aims to produce high-order Bézier triangle meshes with good quality, and exact geometry. Moreover, the meshes are conform to the given boundary and interior curves. The input consists of a region of the B-Rep model and its parametrization data, which correspond to the desired global mesh degree; and a set of subdivision curves each of which is stored in a vector of parametric values. Two restrictions should be noted: the mesh's degree must be greater or equal to the highest degree of the boundary curves; and each curve parametrization must have its own NURBS knots.

The generation of the curve subdivision input to the algorithm is not in the scope of this work. However, some algorithms developed by the authors are available in the PMGen program [61]. Generating the curve subdivision is a preprocessing step that impacts the resulting mesh quality. So, it should be done carefully in geometries with complex features, such as regions of high curvatures and narrow regions. Notice that maintaining the boundary parametrization allows a straightforward generation of compatible meshes for models with multiple regions, e.g., the sheet model shown in Fig. 1 (b).

Once parametrization data is available, Bézier segments are obtained as follows: the NURBS curves' degrees are elevated to the mesh's degree; the curves are subdivided using Knot Insertion; and Bézier segments are attained using Bézier extraction. The orientation of each Bézier segment is inherited from its corresponding B-Rep loop.

The mesh generation algorithm consists of three steps:

- **Linear Mesh Generation.** This step constructs the initial mesh that defines the mesh's topology. Here, all the curved segments in the input boundary are represented by their respective chords.
- **High-order Mesh Generation.** This step elevates the degree of the linear mesh to the defined degree, and restores the curved segments of the input boundary. Also, the mesh topology is modified locally to avoid singularities.
- **High-order Mesh Smoothing.** This step improves the quality of the elements near curved edges through the weight and coordinate smoothing procedure discussed in Section 3.3.

Many structured meshing algorithms generate high-order elements directly [62–64]. However, in the unstructured meshing context, it is usual to build initial linear meshes in order to generate high-order ones [30, 39, 45, 65].

In the case of isogeometric mesh generation with Bézier triangles, other algorithms have been proposed [30, 32, 35]. However, our approach has some valuable contributions we want to highlight. We consider the curved segments in the evaluation of the sizing function used in the linear meshing step, which usually does not receive any information from geometric models besides segment chords. Moreover, the algorithm includes a procedure to prevent the generation of high-order elements with singularities, a problem that is not discussed in other works related to plane models. Furthermore, our high-order smoothing step improve mesh quality by acting not only on the control points' coordinates but also on their associated weights, which was done in 3D meshing context [31], but not in 2D cases, where only control point weights were smoothed [30]. Finally, our code implementation is well optimized, written in C++, and outperforms the MATLAB implementation presented in [30] by orders of magnitude in processing time. Performance is particularly important in applications that use a meshing algorithm many times, such as in shape optimization [36]. Fig. 4 illustrates the steps used by the proposed algorithm to generate a cubic mesh. Each step of that algorithm is further discussed in the following.

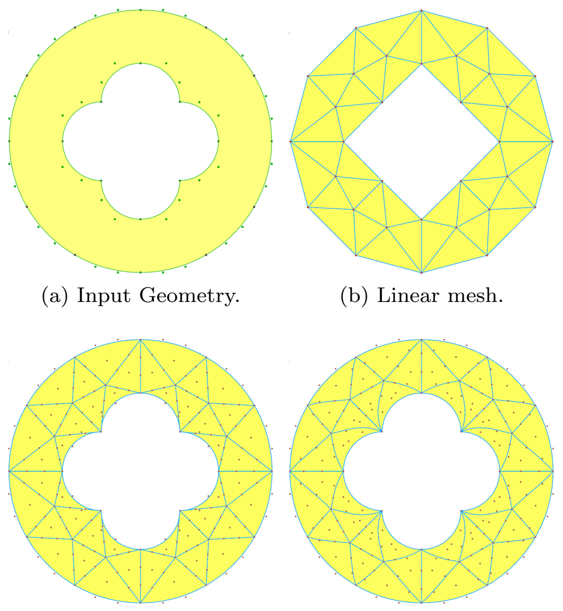

(a) Input Geometry.
(b) Linear mesh.
(c) Degree elevation and boundary recovery.
(d) High-order smoothing.

Figure 4: Procedure for the proposed algorithm.

3.1. Linear mesh generation

For linear mesh generation, we adopt the advancing front algorithm presented in [61]. In this meshing technique, the sizing function is defined by a background quadtree constructed based on the input boundary curve subdivision. For each input Bézier segment, the midpoint $\mathbf{s}_m$ of its chord is evaluated. The quadtree is refined until $q_{side} \leq s_l$, where q_{side} is the side of the quadtree's cell containing point $\mathbf{s}_m$, and s_l is the length of the segment's chord.

This refinement criterion may be inadequate in the case of high-order meshes because the difference between the chord and the arc of some segments increases. Therefore, here the discretization refinement condition is modified as follows. The quadtree is refined if

$$q_{side} > (l_{ce} = s_l - \beta\, d_l), \tag{21}$$

where $s_l = \|\mathbf{p}_2 - \mathbf{p}_1\|$, β is an input coefficient used to control the influence of d_l, which is the signed distance from $\hat{\mathbf{C}}(t_m)$ to the chord $\overline{\mathbf{p}_1\mathbf{p}_2}$ computed as

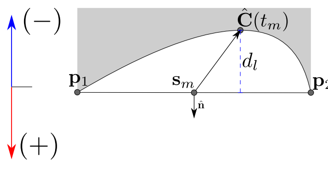
$$d_l = \hat{\mathbf{n}} \cdot (\hat{\mathbf{C}}(t_m) - \mathbf{s}_m). \tag{22}$$

The unit vector $\hat{\mathbf{n}}$ perpendicular to the chord is a 90° clockwise rotation of the unit vector oriented from $\mathbf{p}_1$ to $\mathbf{p}_2$, i.e.,

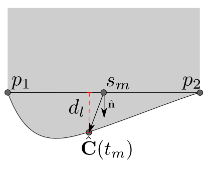
$$\hat{\mathbf{n}} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \frac{\mathbf{p}_2 - \mathbf{p}_1}{||\mathbf{p}_2 - \mathbf{p}_1||}; \tag{23}$$

and t_m is the center of the parametric range of the NURBS curve $\hat{\mathbf{C}}(t)$. We established limits to l_{ce}, i.e., $l_{ce} \in [0.5s_l, 1.5s_l]$, to avoid an excessive variation that may lead to negative values of l_{ce}.

Eq. (21) aims to define sizing functions that take into account the deflection of curved edges. Note that $d_l < 0$ for concave segments, $d_l > 0$ for convex segments and $d_l = 0$ for straight segments, which is the default value used in the original algorithm (see examples in Fig. 5). The segment orientation is given by the loops adjacent to it. In case of constraint edges, where both orientations are considered during mesh generation, $d_l = 0$ is adopted to avoid a low sizing value for the concave segment. The effect of Eq. (21) is further discussed in Section 4.



(a) Concave case.



(b) Convex case.

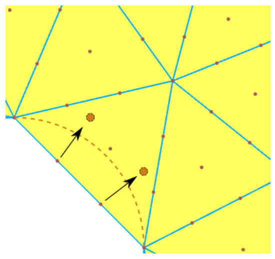
Figure 5: d_l measure in concave and convex segments.

After the initial construction phase, the quadtree is refined by the maximum cell size obtained on boundary cells in the previous phase, and also by enforcing a 1:2 refinement ratio [66, 67]. The remaining procedures related to the linear meshing are described in [61].

Notice that additional criteria could be adopted in the construction of the sizing function, for instance, when a user-defined size map is specified or when a curvature criteria is considered in the case of mesh generation of parametric surfaces. In fact, any linear meshing algorithm capable of producing meshes conforming to a given boundary could be used.

3.2. High-order mesh generation

With a good linear mesh established, the Degree Elevation algorithm is applied to each triangle of the mesh to create an initial high-order mesh. The data structure used for mesh storage should be considered in the implementation of this step. First, the internal control points of the edges, which are Bézier curves of degree p, are determined. Then, the internal control points of each Bézier triangle of degree p are determined, and the incidence of the control points belonging to the element edges is stored. In both cases, the Degree Elevation can be performed by linear interpolation since these geometries are linear. Finally, the input boundary given by the geometric model is assigned to corresponding elements adjacent to the input edges, as depicted in Fig. 6.


(a) Element after degree elevation.

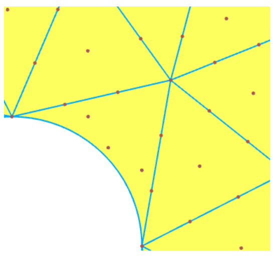

(b) Element after boundary replacement.

Figure 6: Boundary replacement applied to a cubic Bézier Triangle.

After generation of the initial high-order mesh, care must be taken to avoid high-order elements with singularities at corner control points. The Jacobian of the element mapping tends to zero at vertices adjacent to curved edges since the angle between those edges approaches 180°, as shown in Fig. 7 (b). Note that optimal convergence can only be achieved if the determinant of the Jacobian matrix becomes constant under mesh refinement [30, 68], which cannot be guaranteed in those cases.

A threshold angle $\theta_c = 155°$ is adopted. This value was tested in many examples and was adequate to detect this issue. Elements adjacent to two input edges are grouped with the neighboring element of their internal edge and split into four elements. The new corner vertex shared by all new triangles is located at the middle of the collapsed edge. Fig. 7 illustrates the application of this procedure. A more unusual case occurs when an element is adjacent to three input edges, which can happen when closed constraint loops are considered. In this case, the element is split into three, as shown in Fig. 8. Notice that this issue cannot be solved without changing mesh topology locally, even if a nodal reallocation optimization is performed. This problem is also discussed in the context of isogeometric mesh generation of Bézier tetrahedra, hexahedra, wedges and pyramids in [31].

3.3. High-Order Mesh Smoothing

Mesh validity is a primary concern in the high-order mesh generation field. Invalid or bad quality elements may be generated during a boundary replacement step, where curved segments are assigned to straight edges. This problem tends to be worse as the curvature of the edge increases and as the element degree rises. Therefore, mesh deformation schemes are adopted in many works to tackle this issue.

In Elasticity Smoothing (ES), an elastic analysis is performed considering prescribed displacements on the boundary nodes (or control points) to move them from a straight configuration to a curved configuration represented in a geometric model. Linear analyses are usually employed [69, 70], but the use of incremental linear analyses [54], nonlinear elastic analyses [52], and thermo-elastic analyses [53] is also reported in the literature. The last two methodologies, in comparison with the linear analysis approach, are more effective and produce meshes with better quality, but have higher computational cost. Fig. 9 illustrates an example of the application of linear elasticity.

Control point weight smoothing is an analogous approach that aims to reduce the variation of the weights over the domain [30]. In this case, a steady-state heat conduction analysis is performed using weights of the boundary control points as prescribed temperature along the boundary of the domain, smoothing internal weights of the mesh. Although an excessive variation of weights does not result in tangled elements, it directly affects the element's parametric mapping and quality, and can inhibit optimal convergence rates in finite element applications [38].

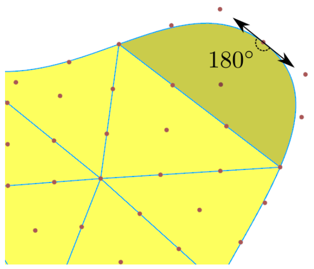
The high-order smoothing (HOS) adopted here consists of two steps. First, the control weights are smoothed using the described heat-transfer approach. Then, a linear elastic analysis is performed over the mesh with smoothed weights in order to smooth the control points' coordinates. Linear analysis is chosen instead of more complex alternatives to keep the HOS step efficient while preventing the occurrence of tangled elements, and improving mesh quality.

Even if linear analysis is adopted, the computational cost of the HOS step is notably high in comparison with the linear mesh step. It may limit the use of the meshing algorithm in some contexts, as in shape optimizations, especially when heuristic algorithms (e.g., genetic algorithm) are used.

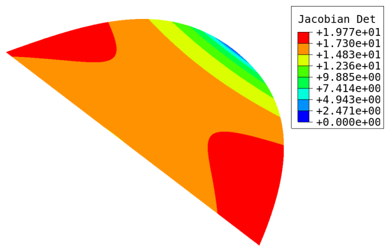
On the other hand, a considerable number of the mesh's elements have negligible displacements. For instance, Fig. 10 shows an example of the application of linear elasticity and the obtained displacement field. Thus, only elements close to the curved edges of the boundaries have significant displacements, while the majority of elements presents small or even negligible values. This behavior is analogous to that observed in control point weight smoothing. Therefore, we propose to apply both analysis locally, which can greatly reduce the computation cost of the HOS step, as will be discussed in the following.

3.3.1. Localized Linear Analysis

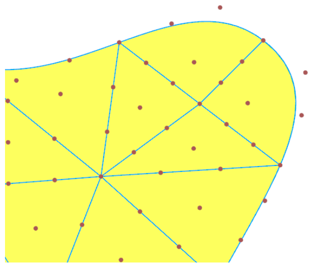
We seek to find a set of elements (F) of the mesh to process linear analyses locally. A subset of the input edges is evaluated, and the elements adjacent to their vertices form an initial set of elements F^i (In



(a) Detection of singularity.

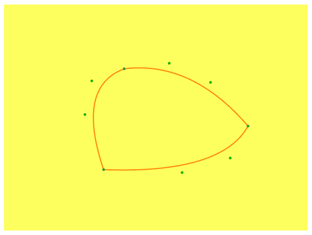


(b) Jacobian determinant of an element with singularity.

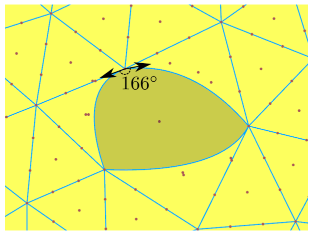


(c) New elements without singularity.

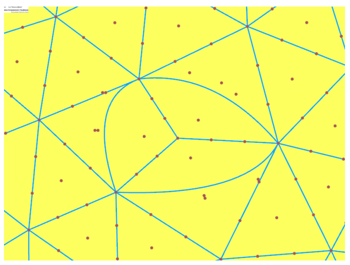
Figure 7: Four-element split procedure used on elements with two adjacent input curved edges.



(a) Constraint input edges.

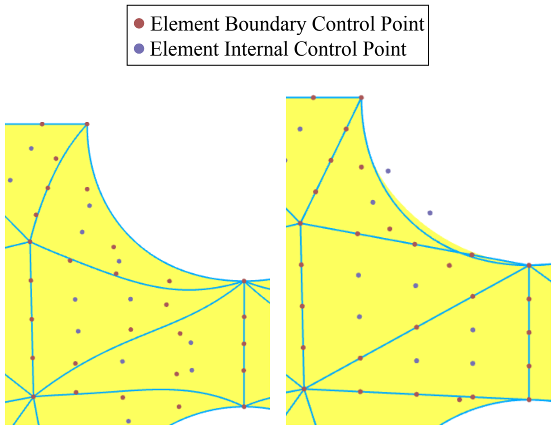


(b) Detection of singularity.



(c) New elements without singularity.

Figure 8: Three-element split procedure applied to elements with three adjacent input curved edges.


● Element Boundary Control Point
● Element Internal Control Point

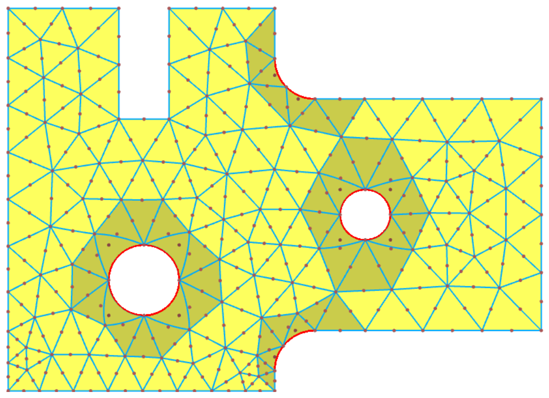
(a) With ES. (b) Without ES.

Figure 9: Boundary replacement of fourth degree: (a) with elasticity smoothing, and (b) without elasticity smoothing.

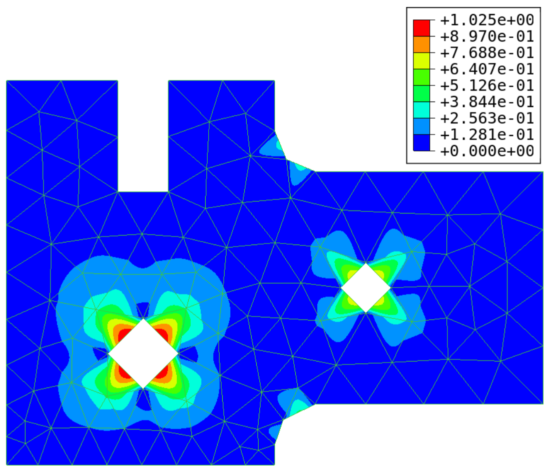
Fig. 10 (a), F^i corresponds to the marked elements). Next, F^i is expanded by adding all adjacent elements of its vertices. This process is repeated $(\alpha - 1)$ times $(\alpha = 1 \implies F = F^i)$, where α is the degree of adjacency, an input parameter.

In the case of linear elastic analysis, curved edges are selected to form F^i. An edge is classified as curved if the length of its control polygon is not equal to its chord. A threshold value of 1% is adopted to check that condition. In the thermal analysis, only rational edges are selected, in other words, edges with $w_i \neq 1$.

Fig. 11 depicts an application of the steps discussed above to find the elements considered in HOS. The set of elements F for each analysis usually has disjoint groups of elements, which should be analyzed separately. The evaluation of each group is carried out with a sub-mesh and sub-mesh-edge data struc-



(a) Mesh after elasticity smoothing.



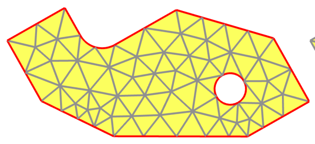
(b) Displacement magnitude.

Figure 10: An application of the elasticity smoothing and the displacements obtained from linear analysis. Only the neighborhood of curved edges presents relevant displacements.

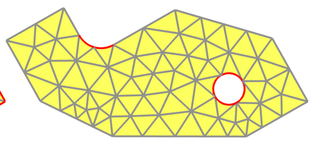
ture. Those structures are directly related to the face and edge topological structure of the mesh.

3.3.2. Implementation discussion

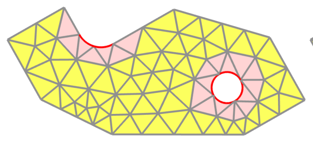
The *sSubMesh* class stores the necessary information for processing the linear analysis: A list of elements (*ElemList*), which contains the elements used in the local simulation; and a list of sub-mesh edges (*ExtEdge*), where the boundary conditions are defined. The class *sSubMeshEdge* has references to adjacent elements in clockwise (cw_{sm}) and counter-



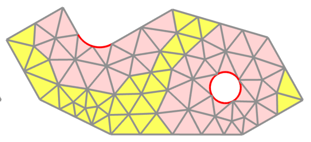
(a) Boundary edges E_b.



(b) Selected curved edges E_c.



(c) Element set F^i.



(d) Element set F.

Figure 11: Steps to find the element set processed in HOS.

clockwise (ccw_{sm}) orientations, similar to the well-known *winged-edge* data structure [20, 21], and also a reference to corresponding mesh edge. Fig. 12 shows the class diagram of the sub-mesh data structure.

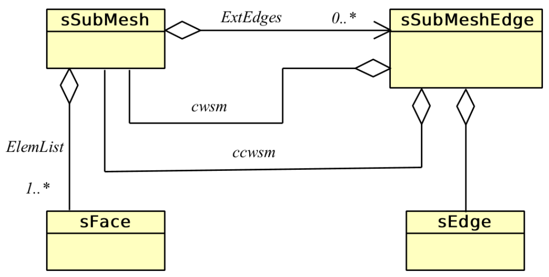


Figure 12: Class diagram of the sub-mesh data structure.

The evaluation of existing sub-meshes is performed using the same strategy used for the disjoint set data structure [71]. The *MakeSet* operator initializes a sub-mesh object for each element in F, inserting each *sSubMeshEdge* object of the element edge into *ExtEdge*, as presented in the algorithm illustrated in Fig. 13.

After the initial construction, the inner edges that divide two sub-meshes are identified if both references ccw_{sm} and cw_{sm} exist. The Union operator is applied to each inner *sSubMshEdge*, resulting in a set of disjoint sub-meshes. Fig. 14 shows the Union operator algorithm, where the *ElemList* and *ExtEdge* lists of the output sub-mesh are updated appropriately. Notice that a *sSubMeshEdge* can store the same object

```
Input:  Mesh element $elm$;
        Adjacent sub-mesh edges $E_{adj}$;
Output: Sub-mesh $sm$ initialized;

$sm.ElemList.add(elm)$;
foreach $e_i$ in $E_{adj}$ do
    $sm.ExtEdge.add(e_i)$;
    if $IsCounterClockwise(e_i.edg, elm)$ then
        $e_i.ccw_{sm} \leftarrow sm$;
    else if $IsClockwise(e_i^{adj}.edg, elm)$ then
        $e_i.cw_{sm} \leftarrow sm$;
end
```

Figure 13: Pseudocode of the *MakeSet* operator.

in both ccw_{sm} and cw_{sm} references (this situation happens in Fig. 16 (g)). In this case, the Union operation is interrupted.

```
Input:  Sub-meshes $sm_1$, $sm_2$;
        Adjacent sub-mesh edge $e_a$;
Output: Union $sm_1 \cup sm_2$ stored in $sm_1$;

$sm_1.ExtEdge.remove(e_a)$;
$sm_2.ExtEdge.remove(e_a)$;
if $e_i.ccw_{sm} = e_i.cw_{sm}$ then return;
$sm_1.ElmList.add(sm_2.ElmList)$;
foreach $e_i$ in $sm_2.ExtEdg$ do
    if $e_i.ccw_{sm} = sm_2$ then
        $e_i.ccw_{sm} \leftarrow sm_1$
    else if $e_i.cw_{sm} = sm_2$ then
        $e_i.cw_{sm} \leftarrow sm_1$
end
$sm_1.ExtEdge.add(sm_2.ExtEdge)$;
$sm_2.ElmList.clear()$;
```

Figure 14: Pseudocode of the *Union* operator.

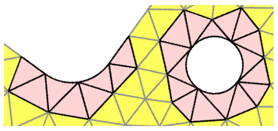
The complete algorithm for the evaluation of sub-meshes is presented in Fig. 15. The *SelectEdges* function evaluates a set of edges in accordance with the criteria discussed early. Note that any set of elements can be passed as input. An example of the application of the algorithm is shown in Fig. 16.

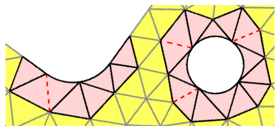
Once the sub-meshes are found, linear analysis is conducted on each one. Note that this step can be carried out in parallel. The boundary conditions adopted in each sub-mesh are prescribed values applied to the control points of the edges in *ExtEdge*

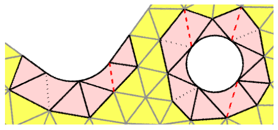
```
Input:  List of boundary edges $E_b$;
Output: Lists of all sub-meshes $M_{sub}$;
        Lists of all sub-meshe edges $E_{sub}$;

// Form initial element set $F^i$.
$E_c \leftarrow SelectEdges(E_b)$;
$F \leftarrow E_c.vertices.AdjacentElements$;
// Expand $F^i$ to $F$.
for $i = 1$ to $\alpha$ do $F.add(F.AdjacentElements)$;
// Initialize $M_{sub}$ and $E_{sub}$ lists.
foreach $elm$ in $F$ do
    $E_{adj} \leftarrow elm.AdjacentSubEdges$;
    $E_{sub}.add(E_{adj})$;
    $M_{sub}.add(MakeSet(elm, E_{adj}))$;
end
// Merge all sub-meshes.
foreach $e_i$ in $E_{sub}$ do
    if $e_i.ccw_{sm}$ and $e_i.cw_{sm}$ exists then
        Union($e_i.ccw_{sm}$,$e_i.cw_{sm}$);
        if $e_i.ccw_{sm}.ElmList$ is $\varnothing$ then
            $M_{sub}.remove(e_i.ccw_{sm})$
        else if $e_i.cw_{sm}.ElmList$ is $\varnothing$ then
            $M_{sub}.remove(e_i.cw_{sm})$
end
```

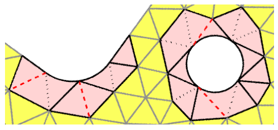
Figure 15: Pseudocode of the algoritm to evaluate sub-meshes.

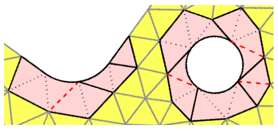
list. The input constraint edges and vertices from the geometric model must also be considered. An academic finite element analysis software developed at Laboratório de Mecânica Computacional e Visualização is employed to process both elastic and thermal analysis, and its routines are available in PMGen.

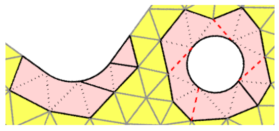

(a) Initial sub-meshes.

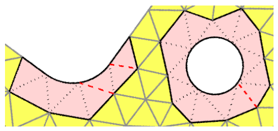

(b) 4 unions processed.


(c) 8 unions processed.


(d) 12 unions processed.


(e) 16 unions processed.


(f) 20 unions processed.


(g) 23 unions processed.

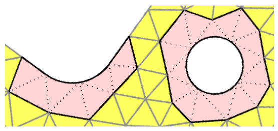

(h) Final sub-meshes.

Figure 16: Application of the algorithm shown in Fig. 15 to a set of 24 elements, resulting in two sub-meshes with 8 and 16 elements.

4. Quality metrics

Quality metrics are measures used to assess the quality of mesh elements. Although it is important to consider the numerical results derived from the application of a mesh, it is accepted in the literature that elements with excessive distortion, such as triangles with high angles, decrease the accuracy and effectiveness of numerical simulations. Therefore, metrics based on element geometry are commonly used in the context of mesh generation and mesh improvement.

In the case of high-order elements, the quality metrics are more complex due to the nonlinear mapping between the element's reference space and the element's physical space, with the presence of localized distortions. The Jacobian (i.e., the determinant of the Jacobian matrix) is commonly used in defining metrics for high-order elements because of the geometrical information it contains.

The ratio Q_j^e of the minimum Jacobian to the maximum Jacobian in the element, known as the *scaled Jacobian*, is a metric used in many works [30, 39, 53, 69, 70]. So,

$$Q_j^e = \frac{J_{min}^e}{J_{max}^e}, \tag{24}$$

where J_{min}^e and J_{max}^e are the smallest and the largest value of the Jacobian inside the element, and $Q_j^e = 0$ if $J_{min}^e \leq 0$. Large variations in the Jacobian diminishes the performance of an element, even if $J > 0$. However, an extremely distorted element may show small or no variation in J [38].

On the other hand, there are Jacobian-based metrics for linear elements that can be used to define high-order metrics and can better detect distortion [42, 45, 72]. Consider the triangle shape metric

$$T_s = \frac{\sqrt{3}}{4}\frac{L_m^2}{A_T}, \tag{25}$$

where L_m is the mean edge length

$$L_m = \sqrt{\frac{1}{3}\sum_{i=1}^{3} L_i^2}, \tag{26}$$

in which L_i is the length of edge i of the triangle and A_T is the triangle's area. T_s varies in the range [0,1], attaining its maximum value for equilateral triangles. Knupp showed that this linear metric can be evaluated using the Jacobian matrix [73], i.e.,

$$T_j = \frac{\sqrt{3}\,\gamma}{\lambda_{11} + \lambda_{22} - \lambda_{12}}, \tag{27}$$

where λ_{11}, λ_{22} and λ_{12} are the components of the metric tensor $A = \mathbf{J}^T\,\mathbf{J}$ and $\gamma = \sqrt{\lambda_{11}\,\lambda_{22} - \lambda_{12}^2}$.

Equation (27) can be used in high-order triangle elements to evaluate its quality at each parametric coordinate (r, s). Thus, one possible element-wise quality metric consists of the smallest value of T_j obtained in the element, i.e.,

$$J_{ts}^e = \min_{(r,s)}(T_j). \tag{28}$$

Two metrics for global evaluation of meshes are defined from J^e_{ts}, the smallest J^e_{ts} in the mesh,

$$J_{ts} = \min_e \left(J^e_{ts}\right), \tag{29}$$

and the average J^e_{ts} in the mesh,

$$J^m_{ts} = \frac{\sum_{e=1}^{\hat{n}} J^e_{ts}}{\hat{n}}, \tag{30}$$

where $\hat{n}$ is the number of elements.

Notice that J^e_{ts} is equal to T_s for linear elements. Thus, J_{ts} is suitable for both high-order and linear elements, and, for this reason, it is not necessary to combine two distinct metrics to make a general one, as done in [40]. Moreover, for a given mesh, J_{ts} does not change under uniform refinement and represents a lower bound for any mesh obtained in this fashion. In contrast, the expected behavior of the *scaled Jacobian* is to converge to 1 under uniform refinement (only for h-refinements). Finally, notice that, in the literature, there are approaches that define high-order metrics by integrating linear metrics, rather than evaluating them at a set of points [42]. Those approaches have been successfully applied in the context of mesh optimization [45] and moving meshes [48].

In advancing front algorithms, the mesh generation process is guided by sizing functions that aim to create high quality triangles, especially at the boundary. However, in the case of high-order meshes, the triangle's height established by the sizing function does not take into account curved edges, because the linear mesh generation step is usually performed separately. In this work, the sizing function used in the linear mesh algorithm is modified to take into account curved edges, as discussed in Section 3.1. To show the importance of that modification, consider triangles formed by two lines and a straight base, two lines and a concave circular base, and two lines and a convex circular base. Fig. 18 depicts the variation of the J_{ts} metric for each base with respect to the location of the vertex opposite the base, and the curved triangles with the largest metric. Note that there is a big difference in the height (h_b) that maximizes the J_{ts} metric in those cases.

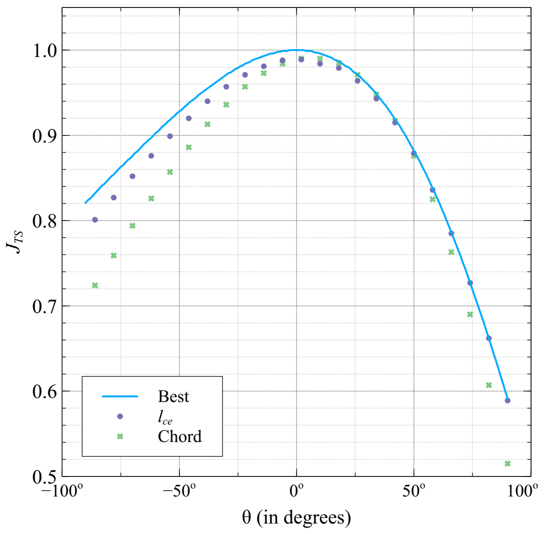
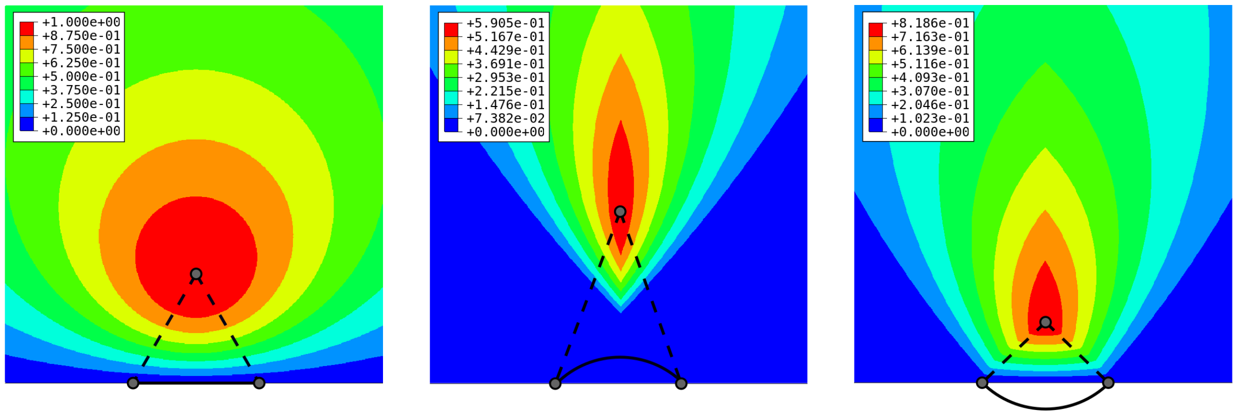


Figure 17: Best J^e_{ts} values by curvature θ of curved circular edges.

Figure 17 displays the variation of the metric J^e_{ts} for isosceles triangles with a circular arc of θ base and three different rules for evaluating the triangle's height. The first rule is the best height for each θ, the second rule is the l_{ce} length defined in Eq. (21), and the third rule is the curve chord, which is commonly adopted by linear mesh generation algorithms, such as those reported in [61]. The results for the height equal to l_{ce} are better in comparison with the chord case. Thus, higher quality meshes are expected when, in the construction of the *quadtree*, the lengths l_{ce} are used, rather than the lengths used in the standard algorithm. However, notice that this modification affects only one of the criteria used in the sizing function's definition, so it is difficult to assess the impact of this modification in practical cases.



(a) Linear edge ($h_b \simeq 0.87 * chord$). (b) Concave edge ($h_b \simeq 1.38 * chord$). (c) Convex edge ($h_b \simeq 0.48 * chord$).

Figure 18: J_{ts} of triangles with a straight base and two circular bases.

5. Examples

The proposed algorithm is evaluated on several examples. The degree of adjacency $\alpha = 2$ and the sizing function's coefficient $\beta = 1.6$ are used in all examples. These values were chosen based on our experience in many examples, including the ones presented in this work. The machine used to run all examples has an Intel Core 2 Quad Q6600 CPU with 2.40GHz and 8GB RAM.

The triangular integration scheme presented in [74] is used here in the linear elastic and thermal analyses. Quadrature rules with order equal to $2\,(p-1)$ are used for a mesh with degree p.

5.1. Effectiveness evaluation

The performance of the proposed algorithm is compared with the algorithm employed in the TriGA program [75], which implements the Dynamic Quadtree algorithm presented in [30]. The input parametrization considered in the proposed algorithm is the same used by TriGA, and cubic degree elements are adopted.

Two examples are considered here, a plate originally presented in [67, 76] and a Torque Arm inspired by the example presented in [36]. The meshes produced by the proposed algorithm are presented in Fig. 20. Table 1 shows the J_{ts} and J_{ts}^m metrics achieved by the proposed algorithm (P) and by TriGA (T). The results show that the proposed algorithm obtains similar quality in terms of the J_{ts} metric, but superior quality in terms of J_{ts}^m, in both cases. In addition, the proposed algorithm yields better quality elements, as shown in the histogram in Fig. 19. This result is expected since no smoothing step is performed on control point coordinates by the TriGA algorithm. The PMGen implementation also presented superior efficiency.

Table 1: Mesh quality obtained in Example 5.1.

Example	J_{ts} (P)	J_{ts} (T)	J_{ts}^m (P)	J_{ts}^m (T)
Plate	0.6419	0.6461	0.9365	0.9258
Torque Arm	0.5517	0.5588	0.9339	0.9038

5.2. Evaluation in complex geometries

In this section, the proposed algorithm is applied to complex geometries. Two examples are studied, a guitar and the fluid between two rotors, where the geometry is based on the problem presented in [63]. In the case of complex geometries, the use of curve

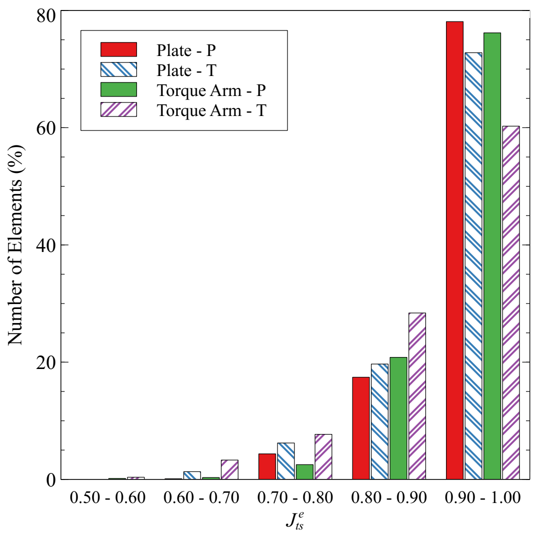
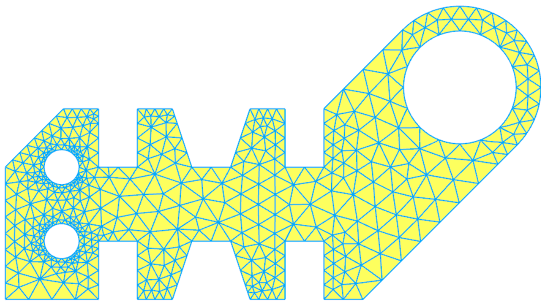
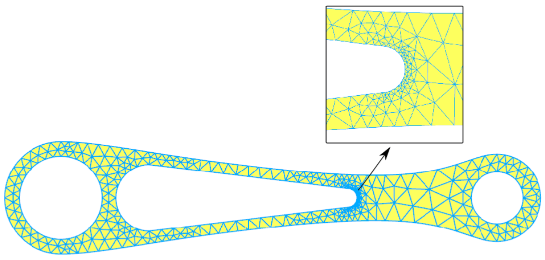


Figure 19: Mesh elements per quality range.



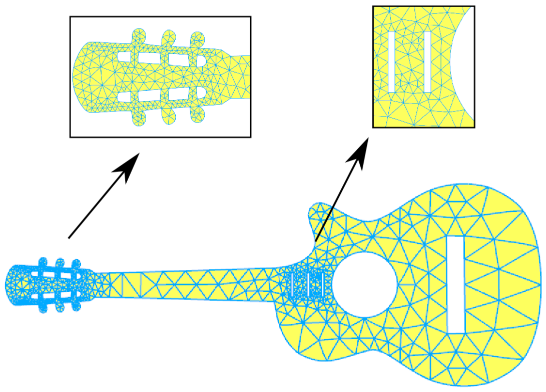
(a) Plate.



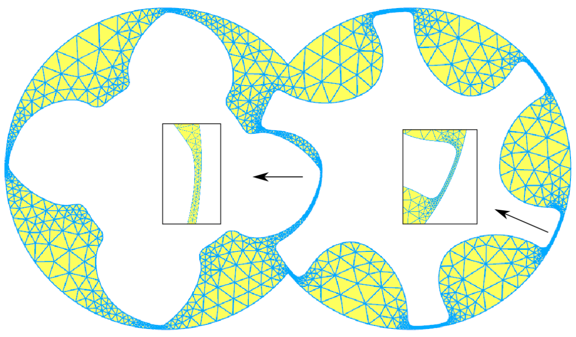
(b) Torque Arm.

Figure 20: Cubic meshes generated by the proposed algorithm in Example 5.1 using discretization provided by TriGA [30].

subdivision is appropriate due to the presence of complex features, such as high curvature and narrow regions. For the examples considered here, curve subdivision is performed by the algorithm developed in [61], which requires three input parameters: the maximum (L_{max}) and the minimum (L_{min}) edge lengths, and the maximum curvature (θ_{max}). The implementation of this algorithm is available in the *PMGen* program. For both models, cubic meshes are used.



(a) Guitar.



(b) Rotors.

Figure 21: Generated meshes of complex geometries.

Table 2 shows the curve subdivision input parameters adopted in each example. The efficiency of the proposed algorithm is compared with Gmsh program [77], using the same curve subdivision adopted in PMGen, cubic T10 finite elements, and a high-order smoothing optimization algorithm available in Gmsh [41, 43, 44]. The parameters used in Gmsh are presented in Table B.1. Notice that the two algorithms are not strictly comparable since Gmsh does not exactly represent rational geometries. However, the capacity to produce valid high-order meshes in complex geometries remains relevant for both programs. Besides, Gmsh is a mature finite element software that has an efficient implementation and is used in many CAE programs. It is important to assess the efficiency of the proposed algorithm in addition to mesh quality. Thus, It is reasonable to compare both programs, considering the same input subdivision and the same final mesh's degree, in terms of efficiency and quality.

Table 2: Curve subdivision parameters in Example 5.2.

Example	L_{max}	L_{min}	θ_{max}
Guitar	8	0	75°
Rotors	30	0	60°

The meshes produced by the proposed algorithm are shown in Fig. 21. Table 3 presents mesh quality results (P for PMGen and G for Gmsh), and Table 4 shows time measurement results, where t_h is the elapsed time in HOS, including the degree elevation step, and t_l is the elapsed time during linear meshing step. The proposed algorithm generates high quality meshes for both examples, but with slightly inferior metrics in comparison with Gmsh. Nevertheless, the proposed algorithm is faster and produces meshes with fewer elements. The HOS step represents a considerable part of the total elapsed time for the proposed algorithm, but the time spent on HOS is comparable to the time spent on the linear meshing step. Notice that, although the mesh optimization procedure in Gmsh has a considerably higher computational cost than that of the proposed HOS, the quality of the mesh it produces is slightly better than the quality delivered by the proposed HOS procedure.

Table 3: Mesh size and mesh quality obtained in Example 5.2.

Example	$\hat{n}$ (P)	$\hat{n}$ (G)	J_{ts} (P)	J_{ts} (G)	J_{ts}^m (P)	J_{ts}^m (G)
Guitar	930	1240	0.464	0.600	0.922	0.948
Rotors	2117	2427	0.565	0.653	0.939	0.950

Table 5 shows the number of sub-meshes (n_{sm}) and the number of elements in sub-meshes ($\hat{n}_{sm}$) obtained in each case. The thermal analysis is applied to a single group of elements in both cases, while elastic analysis is applied several times in many small disjoint groups of elements. The group of elements chosen in thermal analysis is located at rational curves. These models have only one loop with rational curves, the circular hole in the Guitar model and the external boundary on the Rotors model. The loops composed only of rational curves always result in a single group of elements, as observed in the Rotors case, where roughly one half of the mesh is considered in the thermal analysis step. This aspect affects the computational costs of the meshing algorithm and may be considered in some models to exclude the thermal step execution.

Table 4: Time measurements reported in Example 5.2.

Example	t_l^* (P)	t_l^* (G)	t_h^* (P)	t_h^* (G)	$t_h + t_l$ (P)	$t_h + t_l$ (G)
Guitar	63	46	67	471	130	517
Rotors	164	95	190	302	354	397

*Elapsed time measured in milliseconds.

Table 5: Results obtained in Example 5.2.

Example	n_{sm} Elastic	n_{sm} Thermal	$\hat{n}_{sm}$ Elastic	$\hat{n}_{sm}$ Thermal
Guitar	8	1	470	91
Rotors	17	1	469	1102

The impact caused by the localized strategy in HOS is presented in Table 6, where J_{ts} and J_{ts}^m metrics and t_h/t_l are shown for each example studied here, considering global (*Glob*) and local (*Loc*) analysis. The impact on quality metrics is negligible, while excellent improvements in the computational efficiency of HOS is observed.

5.3. Effect of HOS on mesh quality

The effect of HOS on mesh quality is studied in this section. The geometry consists of a mechanical part presented in [78]. The curve subdivision is performed using the same algorithm adopted in the previous section, with $L_{min} = 0$, $\theta_{max} = 90°$, and

Table 6: Quality metrics and computational costs of HOS in global and local analysis.

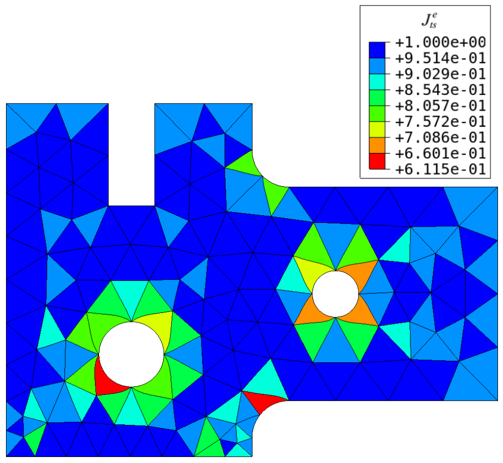
Example	J_{ts}		J_{ts}^m		t_h/t_l	
	Loc	*Glob*	*Loc*	*Glob*	*Loc*	*Glob*
Plate	0.64	0.64	0.94	0.94	0.89	3.72
Torque Arm	0.55	0.55	0.93	0.93	1.70	2.98
Guitar	0.46	0.46	0.92	0.92	1.08	4.75
Rotors	0.56	0.57	0.94	0.94	1.16	3.77

varying the value of L_{max}. Quadratic, cubic, and quartic meshes are generated by the proposed algorithm in each case, with and without the HOS step. In this example, the J_{ts} and J_{ts}^m metrics are evaluated considering only the elements affected in the HOS step since there is no effect on other elements.

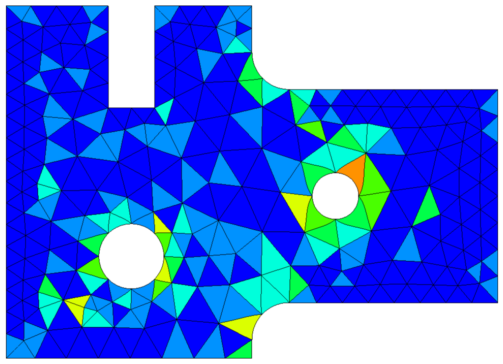
Table 7 shows the mesh quality obtained in each case. Both minimum (J_{ts}) and average (J_{ts}^m) metrics increased in all cases and tend to be far better for higher degrees. The overall element quality is good, even without HOS, but poor quality or tangled elements are generated in some cases. Moreover, the distribution of elements by quality improves in all cases, where more elements in higher quality ranges are reported, as shown in Fig. 23.

Notice that the effect of coordinate smoothing is reduced as the curvature of mesh elements decreases, and as a consequence, the magnitudes of the prescribed displacements become small. Here this behavior is observed as the discretization parameter L_{max} decreases, i.e., for $L_{max} = 2$, where good meshes are generated even without the HOS step. We emphasize that the group of elements affected by the proposed HOS adapts in accordance with the discretization, resulting in an empty set for the elasticity analysis step. Hence, only weight smoothing is applied at very fine levels of curve subdivision. Figure 22 depicts the quartic meshes (with HOS) obtained for each discretization and their quality distributions.

The elapsed times are also reported here. The HOS step takes longer in comparison with the linear meshing step as higher degrees are used for the same level of curve discretization. Notice that, in these cases, the size of the global linear system increases, as well as the computational cost of the numerical integra-



(a) $L_{max} = 6$.


(b) $L_{max} = 4$.

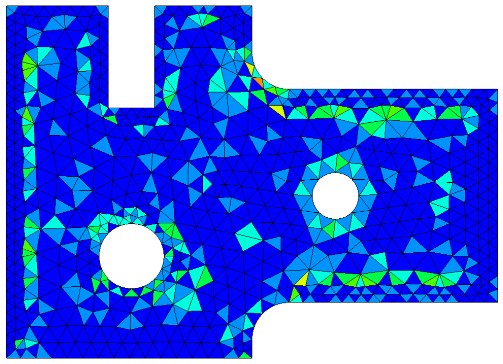
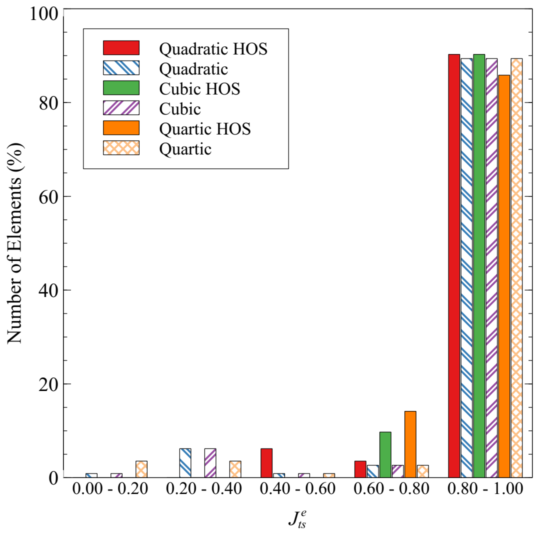
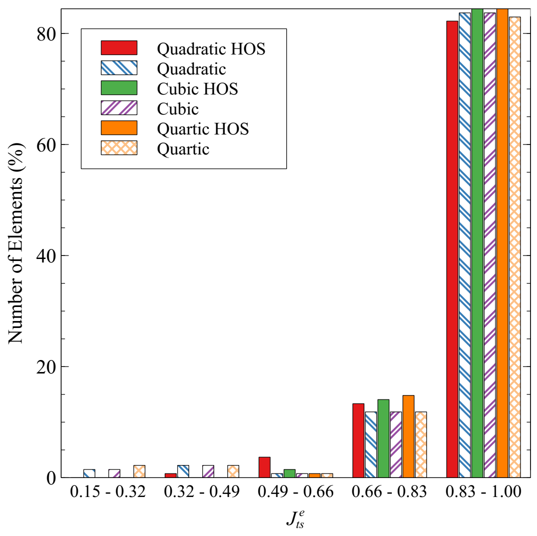

(c) $L_{max} = 2$.

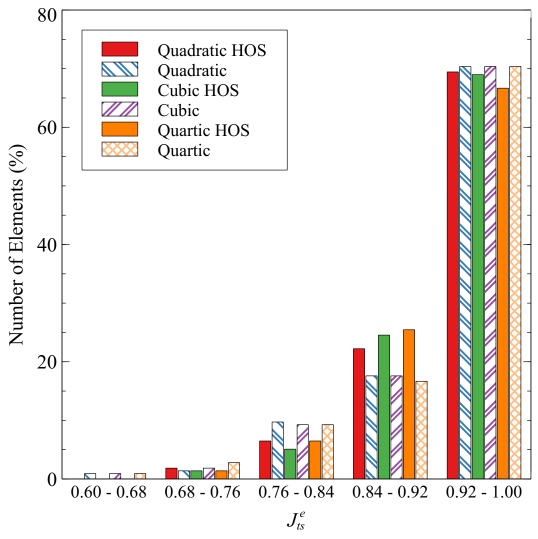
Figure 22: Quartic meshes generated by the proposed algorithm (with HOS) in Example 5.3.



(a) $L_{max} = 6$.



(b) $L_{max} = 4$.



(c) $L_{max} = 2$.

Figure 23: Histogram of element quality for Example 5.3.

Table 7: Influence of HOS on mesh quality and timings.

L_{max}	Degree	J_{ts}	J_{ts} (HOS)	J_{ts}^{m}	J_{ts}^{m} (HOS)	$\hat{n}$	$^*n_{sm}$	$^*\hat{n}_{sm}$	t_h/t_l
	2	0.0315	0.4472	0.9134	0.9226				1.10
6	3	0.0000	0.5650	0.9129	0.9255	200	1 (1)	115 (115)	2.27
	4	0.0000	0.6115	0.9054	0.9202				5.20
	2	0.2467	0.4632	0.9095	0.9101				0.72
4	3	0.2467	0.6168	0.9094	0.9172	399	1 (1)	133 (133)	1.61
	4	0.1439	0.6822	0.9019	0.9162				3.65
	2	0.6642	0.6933	0.9344	0.9366				0.41
2	3	0.6572	0.6994	0.9339	0.9364	1248	4 (4)	214 (214)	0.84
	4	0.6259	0.6957	0.9313	0.9336				1.41

*Values reported in elastic and (thermal) analyses.

tion of element matrices. On the other hand, for the same degree the ratio t_h/t_l decreases as the curve subdivision level rises.

6. Conclusion

This work presented an algorithm for the automatic generation of unstructured isogeometric meshes composed of high-order rational Bézier triangles. The algorithm conforms to an input parametrization of the domain, which is defined by B-Rep using NURBS curves. The algorithm is divided into three steps: 1) generation of a linear mesh using an advancing front algorithm, where mesh topology is defined (the curved segments are considered in the evaluation of the sizing function used during this step); 2) generation of a high-order mesh by degree elevation and boundary replacement, maintaining the exact geometry of the model (singularities in high-order elements are removed in this step); and 3) smoothing of the control points' weights and coordinates in order to improve the quality of the elements in the vicinity of the curved edges of the model (this last step is applied locally, which improves the performance of the proposed algorithm, and avoids tangled and bad quality elements).

The algorithm was compared with TriGA, an academic software capable of generating unstructured isogeometric meshes composed of rational Bézier triangles, and superior results were obtained in two examples. In addition, the algorithm performs well in the case of complex geometries. The high-order smoothing step increased the quality of the meshes as higher curvature and mesh degree are considered in the input boundary representation. Moreover, the local strategy adopted in the proposed HOS yields remarkable efficiency, while it maintains the quality gains obtained by the HOS step.

The efficiency of the proposed algorithm was compared with Gmsh, a mature finite element mesh generator capable of producing and optimizing cubic T10 meshes, and similar results in terms of mesh quality and execution time were observed. Hence, the proposed algorithm implementation presented competitive efficiency and it is capable to generate high-order meshes of any degree and exact geometry.

For future work, we intend to use optimization techniques to further improve the robustness of the algorithm, ensuring that tangled elements are not generated regardless of the input boundary parametrization. In this case, the sub-mesh evaluation algorithm can be used to define small regions for the application of optimization techniques. In addition, we intend to adapt the proposed algorithm in order to use it in the context of surface mesh generation, where trimmed NURBS may be meshed in parametric space. Lastly, we intend to extend our algorithm to handle 3D cases, where Bézier tetrahedra, hexahedra, wedges and pyramids can be generated.

Acknowledgements

The authors acknowledge the financial support provided by CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) in PDSE (Programa de Doutorado Sanduíche no Exterior) process 88881.190187/2018-01, and by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico).

References

[1] T. J. R. Hughes, J. A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Computer Methods in Applied Mechanics and Engineering 194 (39-41) (2005) 4135–4195. doi:10.1016/j.cma.2004.10.008.

[2] L. Piegl, W. Tiller, The NURBS Book, Monographs in Visual Communications, Springer Berlin Heidelberg, Berlin, Heidelberg, 1995. doi:10.1007/978-3-642-97385-7.

[3] F. Auricchio, L. B. da Veiga, A. Buffa, C. Lovadina, A. Reali, G. Sangalli, A fully "locking-free" isogeometric approach for plane linear elasticity problems: A stream function formulation, Computer Methods in Applied Mechanics and Engineering 197 (1-4) (2007) 160–172. doi:10.1016/J.CMA.2007.07.005.

[4] D. J. Benson, Y. Bazilevs, M. C. Hsu, T. J. Hughes, Isogeometric shell analysis: The Reissner-Mindlin shell, Computer Methods in Applied Mechanics and Engineering 199 (5-8) (2010) 276–289. doi:10.1016/j.cma.2009.05.011.

[5] R. D. Cook, D. S. Malkus, M. E. Plesha, R. J. Witt, Concepts and applications of finite element analysis, 4th Edition, Wiley, 2001.

[6] J. A. Cottrell, T. J. R. Hughes, A. Reali, Studies of refinement and continuity in isogeometric structural analysis, Computer Methods in Applied Mechanics and Engineering 196 (41-44) (2007) 4160–4183. doi:10.1016/j.cma.2007.04.007.

[7] A.-V. Vuong, C. Giannelli, B. Jüttler, B. Simeon, A hierarchical approach to adaptive local refinement in isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 200 (49-52) (2011) 3554–3567. doi:10.1016/J.CMA.2011.09.004.

[8] D. Schillinger, L. Dedè, M. A. Scott, J. A. Evans, M. J. Borden, E. Rank, T. J. Hughes, An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces, Computer Methods in Applied Mechanics and Engineering 249-252 (0) (2012) 116–150. doi:10.1016/j.cma.2012.03.017.

[9] E. M. Garau, R. Vázquez, Algorithms for the implementation of adaptive isogeometric methods using hierarchical B-splines, Applied Numerical Mathematics 123 (2018) 58–87. doi:10.1016/J.APNUM.2017.08.006.

[10] T. W. Sederberg, J. Zheng, A. Bakenov, A. Nasri, T-splines and T-NURCCs, ACM Transactions on Graphics 22 (3) (2003) 477. doi:10.1145/882262.882295.

[11] J. A. Cottrell, J. A. Evans, S. Lipton, M. A. Scott, T. W. Sederberg, Isogeometric analysis using T-splines, Computer Methods in Applied Mechanics and Engineering 199 (5-8) (2010) 229–263. doi:10.1016/J.CMA.2009.02.036.

[12] Z. Liu, J. Cheng, M. Yang, P. Yuan, C. Qiu, W. Gao, J. Tan, Isogeometric analysis of large thin shell structures based on weak coupling of substructures with unstructured T-splines patches, Advances in Engineering Software 135 (2019) 102692. doi:10.1016/J.ADVENGSOFT.2019.102692.

[13] T. Dokken, T. Lyche, K. F. Pettersen, Polynomial splines over locally refined box-partitions, Computer Aided Geometric Design 30 (3) (2013) 331–356. doi:10.1016/J.CAGD.2012.12.005.

[14] K. A. Johannessen, T. Kvamsdal, T. Dokken, Isogeometric analysis using LR B-splines, Computer Methods in Applied Mechanics and Engineering 269 (2014) 471–514. doi:10.1016/J.CMA.2013.09.014.

[15] M. Occelli, T. Elguedj, S. Bouabdallah, L. Morançay, LR B-Splines implementation in the Altair RadiossTM solver for explicit dynamics IsoGeometric Analysis, Advances in Engineering Software 131 (2019) 166–185. doi:10.1016/J.ADVENGSOFT.2019.01.002.

[16] J. Deng, F. Chen, X. Li, C. Hu, W. Tong, Z. Yang, Y. Feng, Polynomial splines over hierarchical T-meshes, Graphical Models 70 (4) (2008) 76–86. doi:10.1016/J.GMOD.2008.03.001.

[17] P. Wang, J. Xu, J. Deng, F. Chen, Adaptive isogeometric analysis using rational PHT-splines, Computer-Aided Design 43 (11) (2011) 1438–1448. doi:10.1016/J.CAD.2011.08.026.

[18] C. Giannelli, B. Jüttler, H. Speleers, THB-splines: The truncated basis for hierarchical splines, Computer Aided Geometric Design 29 (7) (2012) 485–498. doi:10.1016/J.CAGD.2012.03.025.

[19] C. Giannelli, B. Jüttler, S. K. Kleiss, A. Mantzaflaris, B. Simeon, J. Špeh, THB-splines: An effective mathematical technology for adaptive refinement in geometric design and isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 299 (2016) 337–365. doi:10.1016/J.CMA.2015.11.002.

[20] M. Mäntylä, Introduction to Solid Modeling, Computer Science Press, Inc., New York, NY, USA, 1988.

[21] I. Stroud, Boundary Representation Modelling Techniques, Springer, London, 2006. doi:10.1007/978-1-84628-616-2.

[22] P. Kang, S. K. Youn, Isogeometric analysis of topologically complex shell structures, Finite Elements in Analysis and Design 99 (2015) 68–81. doi:10.1016/j.finel.2015.02.002.

[23] M. Brovka, J. I. López, J. M. Escobar, J. M. Cascón, R. Montenegro, A new method for T-spline parameterization of complex 2D geometries, Engineering with Computers 30 (4) (2014) 457–473. doi:10.1007/s00366-013-0336-8.

[24] W. Wang, Y. Zhang, L. Liu, T. J. R. Hughes, Trivariate solid T-spline construction from boundary triangulations with arbitrary genus topology, CAD Computer Aided Design 45 (2) (2013) 351–360. doi:10.1016/j.cad.2012.10.018.

[25] L. Liu, Y. Zhang, T. J. R. Hughes, M. A. Scott, T. W. Sederberg, Volumetric T-spline construction using Boolean operations, Engineering with Computers 30 (4) (2013) 425–439. doi:10.1007/s00366-013-0346-6.

[26] J. M. Escobar, R. Montenegro, E. Rodríguez, J. M. Cascón, The meccano method for isogeometric solid modeling and applications, Engineering with Computers 30 (3) (2014) 331–343. doi:10.1007/s00366-012-0300-z.

[27] S. Zeng, E. Cohen, Hybrid volume completion with higher-order Bézier elements, Computer Aided Geometric Design 35-36 (2015) 180–191. doi:10.1016/j.cagd.2015.03.008.

[28] H. Al Akhras, T. Elguedj, A. Gravouil, M. Rochette, Isogeometric analysis-suitable trivariate NURBS models from standard B-Rep models, Computer Methods in Applied Mechanics and Engineering 307 (2016) 256–274. doi:10.1016/j.cma.2016.04.028.

[29] J. I. López, M. Brovka, J. M. Escobar, R. Montenegro, G. V. Socorro, Spline parameterization method for 2D and 3D geometries based on T-mesh optimization, Computer Methods in Applied Mechanics and Engineering 322 (2017) 460–482. doi:10.1016/J.CMA.2017.05.005.

[30] L. Engvall, J. A. Evans, Isogeometric triangular Bernstein-Bézier discretizations: Automatic mesh generation and geometrically exact finite element analysis, Computer Methods in Applied Mechanics and Engineering 304 (2016) 378–407. `doi:10.1016/j.cma.2016.02.012`.

[31] L. Engvall, J. A. Evans, Isogeometric unstructured tetrahedral and mixed-element Bernstein–Bézier discretizations, Computer Methods in Applied Mechanics and Engineering 319 (2017) 83–123. `doi:10.1016/j.cma.2017.02.017`.

[32] N. Jaxon, X. Qian, Isogeometric analysis on triangulations, CAD Computer Aided Design 46 (1) (2014) 45–57. `doi:10.1016/j.cad.2013.08.017`.

[33] S. Xia, X. Qian, Isogeometric analysis with Bézier tetrahedra, Computer Methods in Applied Mechanics and Engineering 316 (2017) 782–816. `doi:10.1016/j.cma.2016.09.045`.

[34] M. Zareh, X. Qian, Kirchhoff–Love shell formulation based on triangular isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 347 (2019) 853–873. `doi:10.1016/J.CMA.2018.12.034`.

[35] N. Liu, A. E. Jeffers, A geometrically exact isogeometric Kirchhoff plate: Feature-preserving automatic meshing and C1 rational triangular Bézier spline discretizations, International Journal for Numerical Methods in Engineering 115 (3) (2018) 395–409. `doi:10.1002/nme.5809`.

[36] J. López, C. Anitescu, N. Valizadeh, T. Rabczuk, N. Alajlan, Structural shape optimization using Bézier triangles and a CAD-compatible boundary representation, Engineering with Computers (2019) 1–16`doi:10.1007/s00366-019-00788-z`.

[37] L. Engvall, Geometrically Exact and Analysis Suitable Mesh Generation Using Rational Bernstein–Bezier Elements, Ph.D. thesis, University of Colorado (jan 2018).

[38] L. Engvall, J. A. Evans, Mesh quality metrics for isogeometric Bernstein–Bézier discretizations, Computer Methods in Applied Mechanics and Engineering 371 (2020) 113305. `doi:https://doi.org/10.1016/j.cma.2020.113305`.

[39] S. Dey, R. M. O'Bara, M. S. Shephard, Towards curvilinear meshing in 3D: The case of quadratic simplices, CAD Computer Aided Design 33 (3) (2001) 199–209. `doi:10.1016/S0010-4485(00)00120-2`.

[40] Q. Lu, M. S. Shephard, S. Tendulkar, M. W. Beall, Parallel mesh adaptation for high-order finite element methods with curved element geometry, Engineering with Computers 30 (2) (2014) 271–286. `doi:10.1007/s00366-013-0329-7`.

[41] C. Geuzaine, A. Johnen, J. Lambrechts, J. F. Remacle, T. Toulorge, The generation of valid curvilinear meshes, Notes on Numerical Fluid Mechanics and Multidisciplinary Design 128 (2015) 15–39. `doi:10.1007/978-3-319-12886-3_2`.

[42] X. Roca, A. Gargallo-Peiro, J. Sarrate, Defining quality measures for high-order planar triangles and curved mesh generation, in: Proceedings of the 20th International Meshing Roundtable, IMR 2011, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 365–383. `doi:10.1007/978-3-642-24734-7-20`.

[43] T. Toulorge, C. Geuzaine, J. F. Remacle, J. Lambrechts, Robust untangling of curvilinear meshes, Journal of Computational Physics 254 (2013) 8–26. `doi:10.1016/j.jcp.2013.07.022`.

[44] A. Johnen, J. F. Remacle, C. Geuzaine, Geometrical validity of curvilinear finite elements, Journal of Computational Physics 233 (1) (2013) 359–372. `doi:10.1016/j.jcp.2012.08.051`.

[45] A. Gargallo-Peiró, X. Roca, J. Peraire, J. Sarrate, Distortion and quality measures for validating and generating high-order tetrahedral meshes, Engineering with Comput-

ers 31 (3) (2015) 423–437. doi:10.1007/s00366-014-0370-1.

[46] A. Johnen, C. Geuzaine, T. Toulorge, J. F. Remacle, Efficient computation of the minimum of shape quality measures on curvilinear finite elements, CAD Computer Aided Design 103 (2018) 24–33. doi:10.1016/j.cad.2018.03.001.

[47] X. J. Luo, M. S. Shephard, L. Q. Lee, L. Ge, C. Ng, Moving curved mesh adaptation for higher-order finite element simulations, Engineering with Computers 27 (1) (2011) 41–50. doi:10.1007/s00366-010-0179-5.

[48] E. Ruiz-Gironés, A. Gargallo-Peiró, J. Sarrate, X. Roca, Automatically imposing incremental boundary displacements for valid mesh morphing and curving, Computer-Aided Design 112 (2019) 47–62. doi:10.1016/J.CAD.2019.01.001.

[49] D. Cardoze, A. Cunha, G. L. Miller, T. Phillips, N. Walkington, A bézier-based approach to unstructured moving meshes, in: Proceedings of the twentieth annual symposium on Computational geometry - SCG '04, ACM Press, New York, New York, USA, 2004, p. 310. doi:10.1145/997817.997864.

[50] C. Kadapa, Novel quadratic Bézier triangular and tetrahedral elements using existing mesh generators: Applications to linear nearly incompressible elastostatics and implicit and explicit elastodynamics, International Journal for Numerical Methods in Engineering 117 (5) (2019) 543–573. doi:10.1002/nme.5967.

[51] A. C. Miranda, M. A. Meggiolaro, J. T. Castro, L. F. Martha, T. N. Bittencourt, Fatigue life and crack path predictions in generic 2D structural components, Engineering Fracture Mechanics 70 (10) (2003) 1259–1279. doi:10.1016/S0013-7944(02)00099-1.

[52] P.-O. Persson, 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, American Institute of Aeronautics and Astronautics, Reston, Virigina, 2013. doi:10.2514/6.2009-949.

[53] D. Moxey, D. Ekelschot, U. Keskin, S. J. Sherwin, J. Peiró, High-order curvilinear meshing using a thermo-elastic analogy, CAD Computer Aided Design 72 (2016) 130–139. doi:10.1016/j.cad.2015.09.007.

[54] R. Poya, R. Sevilla, A. J. Gil, A unified approach for a posteriori high-order curved mesh generation using solid mechanics, Computational Mechanics 58 (3) (2016) 457–490. doi:10.1007/s00466-016-1302-2.

[55] L. Piegl, On NURBS: A Survey, IEEE Computer Graphics and Applications 11 (1) (1991) 55–71. doi:10.1109/38.67702.

[56] J. A. Cottrell, T. J. Hughes, Y. Bazilevs, Isogeometric Analysis: Toward Integration of CAD and FEA, John Wiley & Sons Ltd, Chichester, UK, 2009. doi:10.1002/9780470749081.

[57] M. A. Scott, M. J. Borden, C. V. Verhoosel, T. W. Sederberg, T. J. R. Hughes, Isogeometric finite element data structures based on Bézier extraction of T-splines, International Journal for Numerical Methods in Engineering 88 (2) (2011) 126–156. doi:10.1002/nme.3167.

[58] D. C. Thomas, M. A. Scott, J. A. Evans, K. Tew, E. J. Evans, Bézier projection: A unified approach for local projection and quadrature-free refinement and coarsening of NURBS and T-splines with particular application to isogeometric design and analysis, Computer Methods in Applied Mechanics and Engineering 284 (2015) 55–105. doi:10.1016/j.cma.2014.07.014.

[59] G. Farin, Curves and Surfaces for CAGD A Practical Guide, 5th Edition, Vol. 3, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[60] E. Mainar, J. M. Peña, Evaluation algorithms for multivariate polynomials in Bernstein-Bézier form, Journal of Approximation Theory 143 (1) (2006) 44–61. `doi:10.1016/j.jat.2006.05.007`.

[61] E. S. Barroso, J. A. Evans, J. B. Cavalcante Neto, C. A. Vidal, E. Parente Junior, An algorithm for automatic discretization of isogeometric plane models, in: XL Iberian Latin-American Congress on Computational Methods in Engineering, Natal, 2019.

[62] R. Haber, M. S. Shephard, J. F. Abel, R. H. Gallagher, D. P. Greenberg, A general two-dimensional, graphical finite element preprocessor utilizing discrete transfinite mappings, International Journal for Numerical Methods in Engineering 17 (7) (1981) 1015–1044. `doi:10.1002/nme.1620170706`.

[63] J. Hinz, M. Möller, C. Vuik, Elliptic grid generation techniques in the framework of isogeometric analysis applications, Computer Aided Geometric Design 65 (2018) 48–75. `doi:10.1016/j.cagd.2018.03.023`.

[64] S. Gondegaon, H. K. Voruganti, An efficient parametrization of planar domain for isogeometric analysis using harmonic functions, Journal of the Brazilian Society of Mechanical Sciences and Engineering 40 (10) (2018) 493. `doi:10.1007/s40430-018-1414-z`.

[65] M. S. Shephard, J. E. Flaherty, K. E. Jansen, X. Li, X. Luo, N. Chevaugeon, J. F. Remacle, M. W. Beall, R. M. O'Bara, Adaptive mesh generation for curved domains, Applied Numerical Mathematics 52 (2-3 SPEC. ISS.) (2005) 251–271. `doi:10.1016/j.apnum.2004.08.040`.

[66] A. C. O. Miranda, J. B. Cavalcante Neto, L. F. Martha, An algorithm for two-dimensional mesh generation for arbitrary regions with cracks, in: Proceedings - 12th Brazilian Symposium on Computer Graphics and lmage Processing, SIBGRAPI 1999, IEEE Comput. Soc, 1999, pp. 29–38. `doi:10.1109/SIBGRA.1999.805605`.

[67] M. O. Freitas, P. A. Wawrzynek, J. B. Cavalcante-Neto, C. A. Vidal, L. F. Martha, A. R. Ingraffea, A distributed-memory parallel technique for two-dimensional mesh generation for arbitrary domains, Advances in Engineering Software 59 (2013) 38–52. `doi:10.1016/j.advengsoft.2013.03.005`.

[68] Y. Bazilevs, L. Beirão Da Veiga, J. A. Cottrell, T. J. Hughes, G. Sangalli, Isogeometric analysis: Approximation, stability and error estimates for h-refined meshes, Mathematical Models and Methods in Applied Sciences 16 (7) (2006) 1031–1090. `doi:10.1142/S0218202506001455`.

[69] Z. Q. Xie, R. Sevilla, O. Hassan, K. Morgan, The generation of arbitrary order curved meshes for 3D finite element analysis, Computational Mechanics 51 (3) (2013) 361–374. `doi:10.1007/s00466-012-0736-4`.

[70] R. Abgrall, C. Dobrzynski, 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) (2014) 246–266. `doi:10.1002/fld.3932`.

[71] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, 3rd Edition, The MIT Press, 2009. `arXiv:arXiv:1011.1669v3`, `doi:10.2307/2583667`.

[72] P. Knupp, Label-invariant mesh quality metrics, in: Proceedings of the 18th International Meshing Roundtable, IMR 2009, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 139–155. `doi:10.1007/978-3-642-04319-2_9`.

[73] P. M. Knupp, Algebraic mesh quality metrics for unstructured initial meshes, Finite Elements in Analysis and Design 39 (3) (2003) 217–241. `doi:10.1016/S0168-874X(02)00070-7`.

[74] D. A. Dunavant, High degree efficient symmetrical Gaussian quadrature rules for the triangle, International Journal for Numerical Methods in Engineering 21 (6) (1985) 1129–1148. `doi:10.1002/nme.1620210612`.

[75] L. Engvall, TriGA: Triangular IGA, 2015.

[76] P. O. Persson, Mesh size functions for implicit geometries and PDE-based gradient limiting, Engineering with Computers 22 (2) (2006) 95–109. doi:10.1007/s00366-006-0014-1.

[77] C. Geuzaine, J. F. Remacle, Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering 79 (11) (2009) 1309–1331. doi:10.1002/nme.2579.

[78] O. C. Zienkiewicz, R. L. R. L. Taylor, The finite element method, Butterworth-Heinemann, 2000.

Appendix

A - Bivariate Bernstein Polynomials and Derivatives

The *dynamic programming* concept is used to define an efficient algorithm to evaluate all basis functions appearing in Eq. (12), avoiding waste of resources. The array indexing scheme used to store basis functions is shown in Fig. 3 (b).

During the evaluation of B^p basis, the terms $B^{p-1}_{i-1,j,k}$, $B^{p-1}_{i,j-1,k}$ and $B^{p-1}_{i,j,k-1}$ vanish, respectively, when $i = 0$, $j = 0$ and $k = 0$. These terms are located in the i^{th} column, where $B^{p-1}_{i-1,j,k}$ is in the same position of the current element, and in the $(i+1)^{th}$ column, where $B^{p-1}_{i,j,k-1}$ and $B^{p-1}_{i,j-1,k}$ are to the right of the current element. Hence, for each intermediary degree, processing the evaluation of the basis functions in reverse order (i from 0 to p) does not require auxiliary memory. These considerations guide the evaluation of basis functions in the algorithm presented in Fig. A.1. The inputs are a triangle of degree p and barycentric coordinates r and s and the output are the basis functions stored in the **b** array.

The algorithm for the evaluation of the first derivatives is presented in Fig. A.2, where the derivatives are stored in **dr** and **ds** arrays. The same considerations used to construct the former algorithm are used here. Note that the same memory used to store partial derivatives in the s direction (array **ds**) can be used to store and access basis functions (array **b**).

B - Gmsh parameters

```
Basis(p,r,s,b)
{
  t    = 1 - r - s;
  b[0] = 1;
  for(int d = 1; d <= p; d++)
  {
    n = (d+1)*(d+2)/2;

    // Evaluate basis at column i = 0.
    b[n-d-1] = s * b[n-2*d-1];
    for(q = n-d; q < n-1; ++q)
      b[q] = s * b[q-d] + t * b[q-d-1];
    b[n-1]     = t * b[n-d-2];

    // Loop over each i column.
    for(q = n-2-d, i = 1; i < d; ++i, --q)
    {
      b[q] = r * b[q] + t * b[q-d+i-1];
        for(j = 1, --q; j < d-i; ++j, --q)
          b[q] = r * b[q] + s * b[q-d+i] +
                 t * b[q-d+i-1];
      b[q] = r * b[q] + s * b[q-d+i];
    }
    b[0] = r * b[0];
  }
}
```

Figure A.1: Bézier triangle basis function code.

Table B.1: Gmsh parameters used in Example 5.2.

Parameter	Value
Linear meshing algorihtm	Frontal-Delaunay
Regularization algorithm	Optimization
Target jacobian range	0.8 - 1.2
Number of layers	6
Distance factor	12
Boundary nodes	Fixed
Weight on node displacement	1
Maximum number of iterations	100
Max. number of barrier updates	25
Strategy	Disjoint strong

```
FirstDerv(p,r,s,dr,ds)
{
  // Evaluate basis functions (degree p-1).
  Basis(p-1,r,s,b);
  n = (p+1)*(p+2)/2;

  // Evaluate derivatives at column i = 0.
  dr[n-p-1] = 0.0;
  ds[n-p-1] = p * b[n-2*p-1];
  for(q = n-p; q < n-1; ++q)
  {
    dr[q] = -p * b[q-p-1];
    ds[q] =  p * (b[q-p] - b[q-p-1]);
  }
  dr[n-1] = ds[n-1] = -p * b[n-p-2];

  // Loop over each i column.
  for(q = n-2-p, i = 1; i < p; ++i, --q)
  {
    dr[q] =  p * (b[q] - b[q-p+i-1]);
    ds[q] = -p * b[q-p+i-1];
    for(j = 1, --q; j < (p-i); ++j, --q)
    {
      dr[q] = p * (b[q] - b[q-p+i-1]);
      ds[q] = p * (b[q-p+i] - b[q-p+i-1]);
    }
    dr[q] = p * b[q];
    ds[q] = p * b[q-p+i];
  }
  dr[0] = p * b[0];
  ds[0] = 0;
}
```

Figure A.2: Bivariate Bernstein polynomials first derivative code.