XL CILAMCE
IBERO-LATIN AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGINEERING
NOVEMBER 11-14, 2019
Praiamar Natal Hotel & Convention
Natal, RN-BRAZIL

AN ALGORITHM FOR AUTOMATIC DISCRETIZATION OF ISOGEOMETRIC PLANE MODELS

Elias Saraiva Barroso
elias.barroso@gmail.com
Computer Graphics, Virtual Reality and Animation Group. Universidade Federal do Ceará.
Campos do Pici, 60440-900, Fortaleza, Ceará, Brazil.
John Andrew Evans
Computational Mechanics and Geometry Laboratory. University of Colorado at Boulder
john.a.evans@colorado.edu
3775 Discovery Dr, 80303, Boulder, Colorado, United States.
Joaquim Bento Cavalcante Neto
Creto Augusto Vidal
joaquimb@dc.ufc.br
cvidal@dc.ufc.br
Computer Graphics, Virtual Reality and Animation Group. Universidade Federal do Ceará.
Campos do Pici, 60440-900, Fortaleza, Ceará, Brazil.
Evandro Parente Junior
evandro@ufc.br
Laboratório de Mecânica Computacional e Visualização, Universidade Federal do Ceará
Campos do Pici, 60440-900, Fortaleza, Ceará, Brazil.

Abstract. Mesh generation is an important component of Computer Aided Engineering (CAE) systems. To generate a high quality mesh, suitable discretization must be assigned to a model's boundaries. Unfortunately, many CAE software packages require considerable user interaction to define appropriate discretization when models have complex features, such as high curvatures and narrow regions. This work presents an algorithm for the automatic discretization of boundary curves of plane models capable of handling complex features. Moreover, the algorithm is capable of producing curvilinear boundary discretization by taking into account the parameterization of the boundary curves, and these curvilinear boundary discretizations may be used to generate high quality isogeometric meshes consisting of Bézier triangles. The algorithm has three input parameters, the maximum and minimum edge length and the maximum curvature angle. The proposed algorithm is applied in complex geometric models, demonstrating its ability to balance mesh size and quality.

Keywords: Boundary Discretization, Mesh Generation, Rational Bézier Triangles

1 Introduction

Mesh generation is an important topic in many fields, such as Engineering and Computing. In Engineering, for instance, numerical simulations are important in several areas such as the Automotive, Naval and Aerospace industries. Numerical simulations are also important in the context of physically realistic animations, which is nowadays very relevant in Computer Graphics related applications. Computer Aided Engineering (CAE) systems integrate numerical simulations of physical problems with modeling and post-processing visualization tools. In the case of CAE programs based on the Finite Element Method, the preprocessing step consists of the geometric modeling, mesh generation and definition and assigment of physical attributes for numerical simulation.

There are specialized programs for performing some of these steps, such as Gmsh [1] for mesh generation and Paraview [2] for post-processing. In the mesh generation step, the geometric model is represented by a mesh of elements. Mesh features such as size, distribution and element quality directly influence both simulation results and computational cost.

It is common for mesh generation systems to adopt a partitioning of the geometric model, defining regions and discretization of its boundaries. Partition into regions is important when different materials are present in the model. Moreover, different mesh generation algorithms can be used in each region, with different elements. Suitable *a priori* discretization enables implicit compatibility between meshes of different regions. It is tipicaly up to users to create suitable partitions and discretization for the model.

One common way to define boundary discretization is to define a characteristic mesh size that is used for all, or a part, of model curves and faces. In the *Gmesh* program, for example, the sizing function can be defined in many ways, such as defining values at the model vertices, the use of distance functions or analytics fields, and the use of background mesh, among others [1]. It is important to note that the sizing function can be used both in the definition of the boundary discretization and during mesh generation. Robust sizing function algorithms applied in the context of mesh adaptation and mesh generation are studied in [3]. Although there are robust solutions available in the literature for definition of sizing functions, many software packages requires considerable user interaction to define the initial discretization in the case of models with complex features.

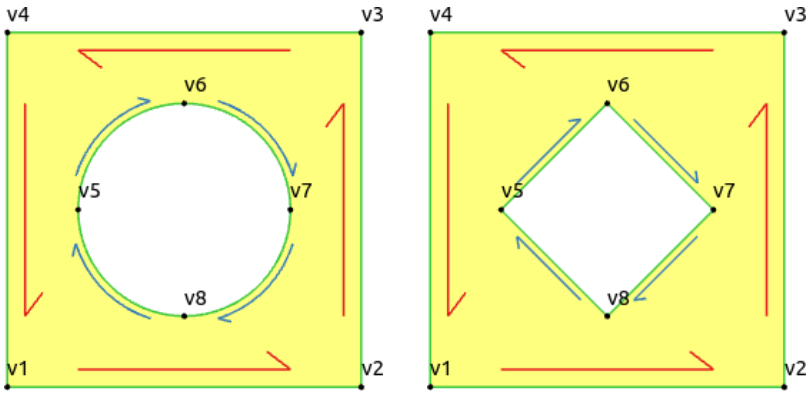
In this paper, we present an algorithm for the automatic discretization of plane models using 3 parameters. The output discretization is used in the generation of high-order isogeometric (exactly geometry) meshes composed by rational Bézier triangles. Aspects as the curvature, the proximity between curves, parametrization of geometry curves and transition between edge lengths are considered. The proposed algorithm is applied in complex geometric models, where high-order meshes with good quality are obtained.

Both discretization and mesh generation algorithms were implemented in the academic program Plane Mesh Generator (PMGen), available in repository `https://github.com/lmcv-ufc/PMGen`. The program has a graphical user interface for problem modeling, application of discretizations and generation of isogeometric and finite element meshes.

This paper is organized as follows. Section 2 review some concepts from the geometry modeling field. The proposed algorithm is discussed in Section 3. The mesh generation algorithm is presented in Section 4. Examples of application of the algorithm are presented in Section 5. The conclusions of this work are presented in Section 6.

2 Geometry Modeling

The geometry of the models studied in this work is given by Non-Uniform Rational B-Splines (NURBS) curves, using the boundary representation paradigm [4, 5]. The topological information is described by a set of vertices and edges organized in loops with consistent orientation (e.g. regions and holes). Moreover, each NURBS curve represents an edge of the topological model by its extreme points. The Figure 1 illustrates the geometry and model topology of a hole plate.



(a) Geometry model. (b) Topology model.

Figure 1. Plate with circular hole model.

2.1 Rational Bézier Curve

A rational Bézier curve of degree p is defined by the linear combination between a set of control points ($\mathbf{p}_i$) and rational bases ($R_{i,p}$):

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

The rational basis is defined by:

$$R_{i,p} = \sum_{i=0}^{p} \frac{B_{i,p}\,w_i}{\sum_{\hat{i}=0}^{p} B_{\hat{i},p}\,w_{\hat{i}}}, \tag{2}$$

where w_i is the weight associated with the control point $\mathbf{p}_i$. The Bernstein polynomials ($B_{i,p}$) are defined by:

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

2.2 NURBS

NURBS are parametric representations widely used in CAD systems, as they offer a mathematical form capable of representing both standard mathematical models (e.g. quadrics, conics, surfaces of revolution) and freeform models using the same database for storage [6]. A NURBS curve of degree p is given by:

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

where m is the number of bases and $\hat{R}_{i,j}$ are the rational bases:

$$\hat{R}_{i,p} = \sum_{i=1}^{m} \frac{N_{i,p}\,\mathbf{w}_i}{\sum_{\hat{i}=1}^{m} N_{\hat{i},p}\,\mathbf{w}_{\hat{i}}}. \tag{5}$$

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 $\Xi = [r_1, r_2, \ldots, r_{m+p+1}]$, the B-Spline base functions are defined by the Cox-de Boor recursive formula [7]:

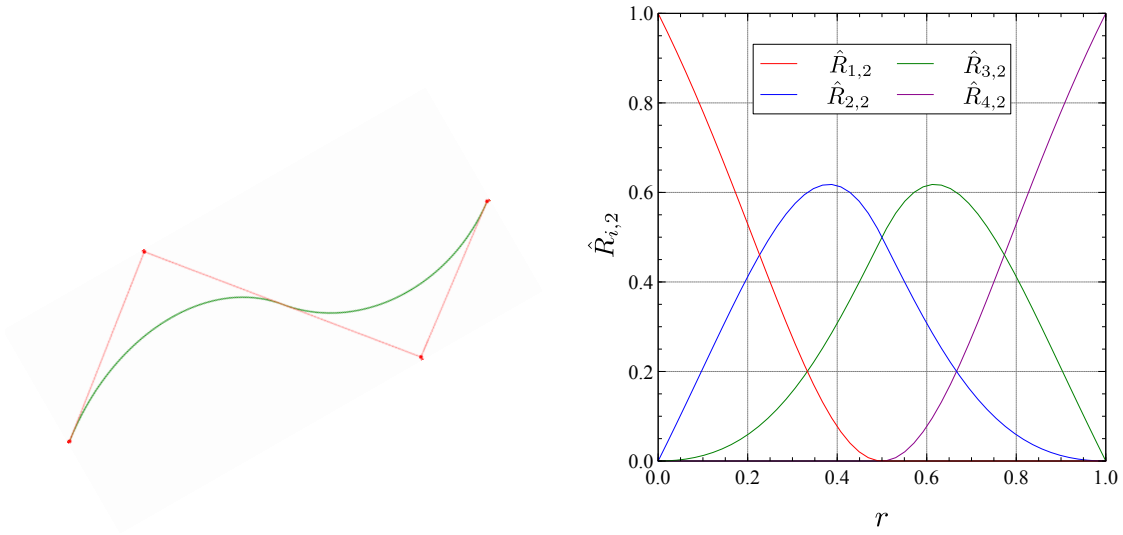
$$N_{i,0} = \begin{cases} 1, & r_i \leq r < r_{i+1}, \\ 0, & \text{otherwise}, \end{cases}$$

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

the number of bases (m) can be calculated using the knot vector size ks and the curve degree p:

$$m = ks - p - 1. \tag{7}$$

The $N_{i,p}$ basis have important properties, such as linear independence, non-negativity ($N_{i,p} \geq 0$), partition of unity ($\sum N_{i,p} = 1$) and compact support ($r \notin [r_i, r_{i+p+1}] \implies N_{i,p} = 0$). The continuity of $\hat{C}(r)$ curve in the knot r_i is C^{p-k_i}, where k_i is the multiplicity of the knot r_i in the knot vector. It is important ponting out that such properties are also valid for rational basis. The Figure 2 shows a quadratic NURBS curve with Ξ = [0 0 0 0.33 0.66 1 1 1] and $\mathbf{w}$ = [1 0.5 0.5 1].



(a) NURBS curve.

(b) Bases $R_{i,p}$.

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

Some algorithms can change the representation of a NURBS entity without modifying its geometry and parameterization. The Knot Insert and Knot Removal modify the knot vector, while the Degree Elevation and Degree Reduction modify the curve degree. These operations are used in the context of geometric modeling [7], but also for applying refinements to isogeometric models [8].

A NURBS curve can be represented by a set of rational Bézier curves, and their control points can be evaluated by the Bézier extraction procedure [7, 9, 10].

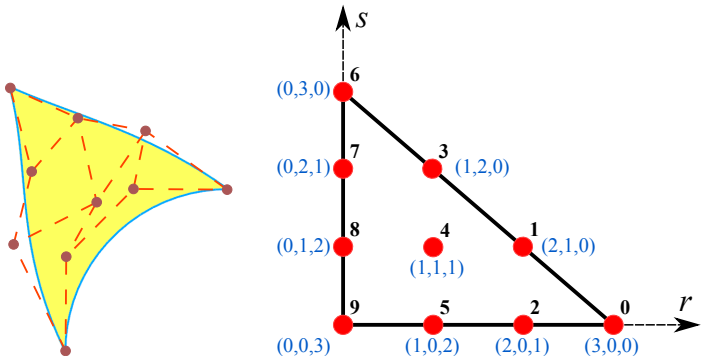
2.3 Bézier Triangle

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

$$T(\boldsymbol{\lambda}) = \sum_{i+j+k=p} B^p_{i,j,k}\, \mathbf{p}_{i,k,l}, \tag{8}$$

where $\boldsymbol{\lambda} = (r, s, t), \forall\, r + s + t = 1$ is a barycentric coordinate, p is the degree, $\mathbf{p}_{i,j,k}$ are the control points and $B^p_{i,j,k}$ are the bivariate Bernstein polynomials. The triple index i, j, k in basis functions and control points respects $i + j + k = p$ and $i, j, k \geq 0$, as illustrated in Figure 3.(b). The number of basis functions (i.e. control points) are computed by:

$$N_{cp} = (p + 1)(p + 2)/2. \tag{9}$$



(a) Physical space. (b) Parametric space.

Figure 3. Cubic Bézier triangle.

The bivariate Bernstein polynomials are defined by:

$$B^p_{i,j,k} = \frac{p!}{i!\,j!\,k!} r^i s^j t^k. \tag{10}$$

Moreover, basis $B^p_{i,j,k}$ satisfy following recursion [11]:

$$B^p_{i,j,k} = r\,B^{p-1}_{i-1,j,k} + s\,B^{p-1}_{i,j-1,k} + t\,B^{p-1}_{i,j,k-1}, \tag{11}$$

which assume $B^0_{0,0,0} = 1$ and $i,j,k < 0 \implies B^p_{i,j,k} = 0$. This recursive definition resemble *de Casteljau* algorithm applied to Bézier Triangles, that is numerically stable [11, 12].

The bivariate Bernstein polynomials have the following properties:

- Linear independency;
- Non-Negativity: $B^p_{i,j,k} \geq 0$;
- Partition of Unity: $\sum_{i+j+k=p} B^p_{i,j,k} = 1$;
- Symmetry.

The first partial derivatives of the bivariate Bernstein polynomials are:

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

The rational version of Bézier triangle is defined considering weights in each control point, analogously to rational tensor product Bézier surfaces and NURBS surfaces. These rational entities are important because they can represent quadrics exactly [11]. The rational basis are defined by:

$$R^p_a = \frac{B^p_a\, w_i}{\sum_{a=0}^{N_{cp}-1} B^p_a\, w_i}, \tag{13}$$

where a is the single index presented in the Figure 3.(b). The rational Bézier triangle is defined by:

$$T_r(\boldsymbol{\lambda}) = \sum_{a=0}^{N_{cp}-1} R^p_a\, \mathbf{p}_a. \tag{14}$$

The first partial derivative is evaluated using quocient rule:

$$\frac{\partial}{\partial r} R_a^p = w_a \frac{W \ \dfrac{\partial}{\partial r} B_a^p - \dfrac{\partial}{\partial r} W \ B_a^p}{W^2},$$
$$\frac{\partial}{\partial s} R_a^p = w_a \frac{W \ \dfrac{\partial}{\partial s} B_a^p - \dfrac{\partial}{\partial s} W \ B_a^p}{W^2}, \tag{15}$$

where W is the weigthing function:

$$W = \sum_{a=0}^{N_{cp}-1} B_a^p \, w_a, \tag{16}$$

the weighting function derivatives are evaluated by:

$$\frac{\partial}{\partial r} W = \sum_{a=0}^{N_{cp}-1} \frac{\partial}{\partial r} B_a^p \, w_a,$$
$$\frac{\partial}{\partial s} W = \sum_{a=0}^{N_{cp}-1} \frac{\partial}{\partial s} B_a^p \, w_a. \tag{17}$$

The jacobian matrix of the plane rational Bézier triangle is given by:

$$\mathbf{J} = \begin{bmatrix} \displaystyle\sum_{a=0}^{N_{cp}-1} \frac{\partial}{\partial r} R_a^p \, \mathbf{x}_a & \displaystyle\sum_{a=0}^{N_{cp}-1} \frac{\partial}{\partial r} R_a^p \, \mathbf{y}_a \\ \displaystyle\sum_{a=0}^{N_{cp}-1} \frac{\partial}{\partial s} R_a^p \, \mathbf{x}_a & \displaystyle\sum_{a=0}^{N_{cp}-1} \frac{\partial}{\partial s} R_a^p \, \mathbf{y}_a \end{bmatrix}. \tag{18}$$

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

2.4 Boundary discretization

Uniform discretization is a simple way to define a boundary discretization. Given a number of divisions (n_d), the length of each segment of a parametric curve $\mathbf{c}(t)$ defined in the interval $[t_a, t_b]$ is given by:

$$L_u = Arc(t_a, t_b) \, / \, n_d, \tag{19}$$

where the arc length is evaluated by:

$$Arc(t_a, t_b) = \int_{t_a}^{t_b} |\mathbf{c}'(t)| \, dt. \tag{20}$$

The arc length can be evaluated numerically:

$$Arc(t_a, t_b) = \sum_{n=1}^{N_{int}} |\mathbf{c}'(\xi_n)| \, w_n \, J, \tag{21}$$

where N_{int} is the number of integration points, ξ_n and w_n are respectively the integration points and the weight associated with each point, and J is the Jacobian transformation between the numerical quadrature system and the interval $[t_a, t_b]$ of the parametric curve $\mathbf{c}(t)$. In this work, Gauss quadrature is used to evaluate the arc lengths.

The set of parametric values $[t_1, \ldots, t_{n_d}]$ that divides the interval $[t_a, t_b]$ in segments with equal lengths in physical space is evaluated using the algorithm shown in Figure 4. The tolerance adopted in this work is $\delta = 10^{-3}$.

Input: Interval $[t_1, t_2]$, length l and tolerance δ.
Output: $t_m \implies Arc(t_1, t_m) = l$.

$t_i \leftarrow t_1$;
repeat
  $t_m \leftarrow (t_1 + t_2)/2$;
  $l_m \leftarrow Arc(t_i, t_m)$;
  if $l < l_m$ **then**
    $t_2 \leftarrow t_m$
  else
    $t_1 \leftarrow t_m$
until $|l - l_m| \leq l * \delta$;
return t_m

Figure 4. Algorithm to find the parametric coordinate that integrates a given arc length.

Alternatively, it is also possible to assume a length L_{ut} so the whole model has a similar edge length. Therefore, the number of divisions adopted for each edge is evaluated by:

$$n_d = \text{round}(Arc(t_a, t_b) \,/\, L_{ut}). \tag{22}$$

Discretization criteria based on curvature are presented in several works [3,7−9]. A naive way to control the curvature of the edges is by limiting the ratio between its arc and chord:

$$\frac{Arc(t_a, t_b)}{Chord(t_a, t_b)} \leq AC_m, \tag{23}$$

where $Chord(t_a, t_b) = |\mathbf{c}(t_b) - \mathbf{c}(t_a)|$ and AC_m is the maximum allowable arc to chord ratio. Figure 5 shows some circular arcs with different AC_m values. In the case of circular arcs, AC_m can be evaluated by the maximum allowable curvature angle (θ_{max}):

$$AC_m = \frac{\theta_{max}}{2\,\sin(\theta_{max}/2)}. \tag{24}$$

Hence:

$$\frac{Arc(t_a, t_b)}{Chord(t_a, t_b)} \leq \frac{\theta_{max}}{2\,\sin(\theta_{max}/2)}. \tag{25}$$

Moreover, the curvature is approximated by the tangent vectors at the segment extreme points [14]:

$$\frac{T_g^a \cdot T_g^b}{|T_g^a|\,|T_g^b|} \geq \cos\theta_{max}, \tag{26}$$

where T_g^a is the tangent vector evaluated at t_a and T_g^b is the tangent vector at t_b.

These approaches are sufficient in several cases. However, they may require considerable user intervention in complex cases. The algorithm presented in the next section allows automatic discretization of complex models using only three parameters.

3 Proposed recursive strategy

The proposed discretization algorithm consists of performing discretization of the model recursively. An initial discretization is adopted on the knots of the NURBS curves. It is worth pointing out that all points of C^0 continuity along each curve is necessarily located on the knots with multiplicity p, where p is the curve degree. Then, each interval $[t_1, t_2]$ is refined according to the following criteria:

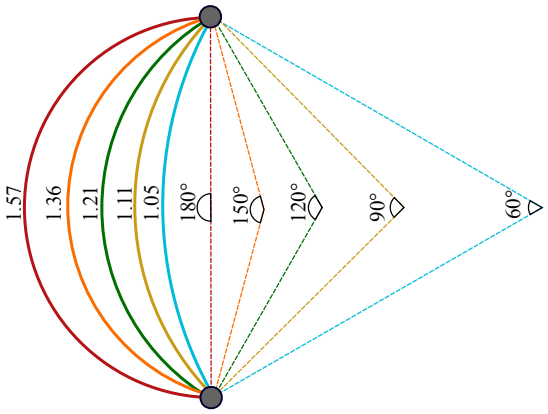


Figure 5. Circular arcs with different arc to chord ratio.

- **Maximum length:** The interval is refined if the length of the arc of the segment is greater than L_{max}.
- **Curvature:** The interval is refined if the approximated curvature is greater than θ_{max} (Equation 26) or the arc to chord ratio of the segment is greater than AC_m (Equation 25).
- **Proximity:** The interval is refined if a nearby region is detected, by applying the procedure presented in Section 3.1.
- **Tangent vector variation:** The interval is refined if tangent vector magnitude vary excessively, as discussed in Section 3.2.
- **Disparity:** The interval is refined if the half of the largest Cartesian component of the vector formed by segment chord is greater than the side of the *quadtree* shown in Section 3.3, considering the leaf cells that contain the endpoints and the midpoint of the above.

If any criterion is satisfied, the segment $[t_1, t_2]$ is split at the midpoint t_m (applying the Algorithm presented in Figure 4), and the two intervals ($[t_1, t_m]$ and $[t_m, t_2]$) are processed again. In addition, the following stopping criterion is adopted: $Arc(t_1, t_2) * 0.75 < L_{min}$. Thus, three input parameters are required: L_{max}, θ_{max} and L_{min}.

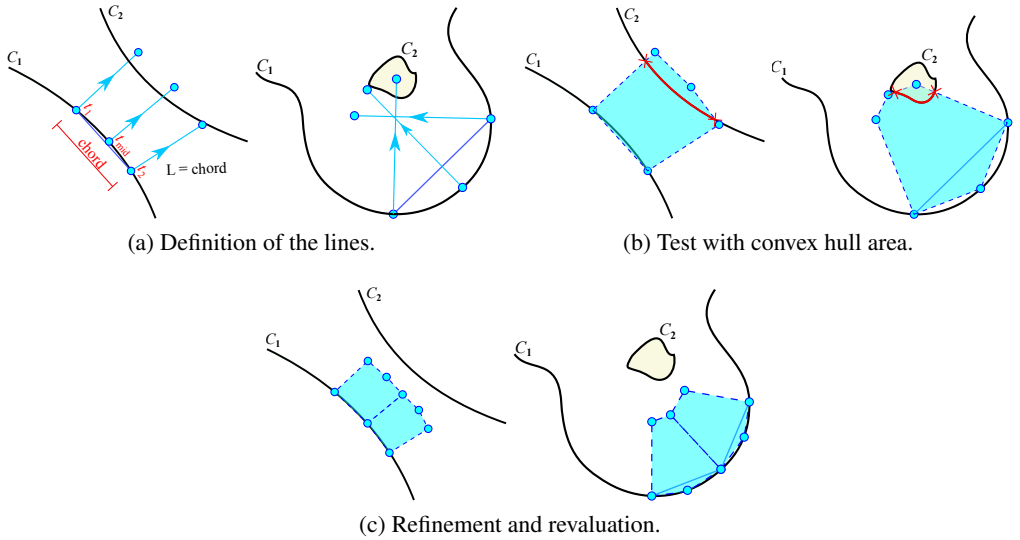
After the application of the recursive discretization, a smoothing step is performed to reduce the differences between the length of neighboring edges. It is worth mentioning that the proposed algorithm can be applied to a subset of edges of the model, so that existing discretizations are maintained.

3.1 Proximity check

In the curve segment $[t_1, t_2]$, three perpendicular lines to the curve are defined starting at the extreme points, t_1 and t_2, and the midpoint t_m. Each line length is set equal to $f_{prox} * Chord(t_1, t_2)$. In this work $f_{prox} = 1.0$ is adopted. The line direction is defined according to the edge adjacent loops. The procedure can be applied in both directions (e.g., when constraint internal edges are considered). The area corresponding to the convex hull of the endpoints of each line is used to test intersections with the rest of the model, except for the $[t_1, t_2]$ interval of the curve. Figure 6 illustrates the application of the proximity check in two cases.

The described procedure presents an infinite loop in models with sharp corners. To avoid this problem, quadrants are formed in each extreme point (t_1 and t_2). If segments near the endpoints are located in these quadrants, then the proximity check is interrupted. Figure 7 illustrates an example of this problem. It is important to note that the other criteria are still evaluated and can refine the segment.

It is worth mentioning that the proximity check can be also performed using medial axis [3, 16].



(a) Definition of the lines.

(b) Test with convex hull area.

(c) Refinement and revaluation.

Figure 6. Application of the proximity test.

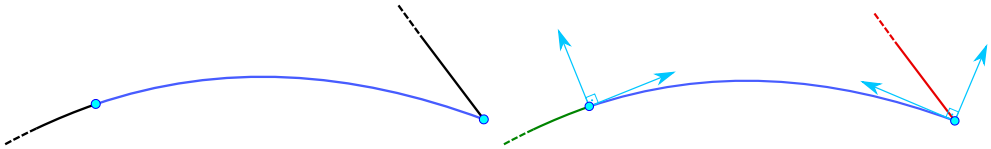


Figure 7. Detection of sharp corners. The right segment stop the execution of the proximity test.

3.2 Tangent vector variation

The parametrization of the model curves influences directly in the quality of the high-order isogeometric elements adjacent to them. Here, the quality of the parametrization is given by the ratio between the minimum (T_g^{min}) and maximum (T_g^{max}) tangent vector modulus along curved segments:

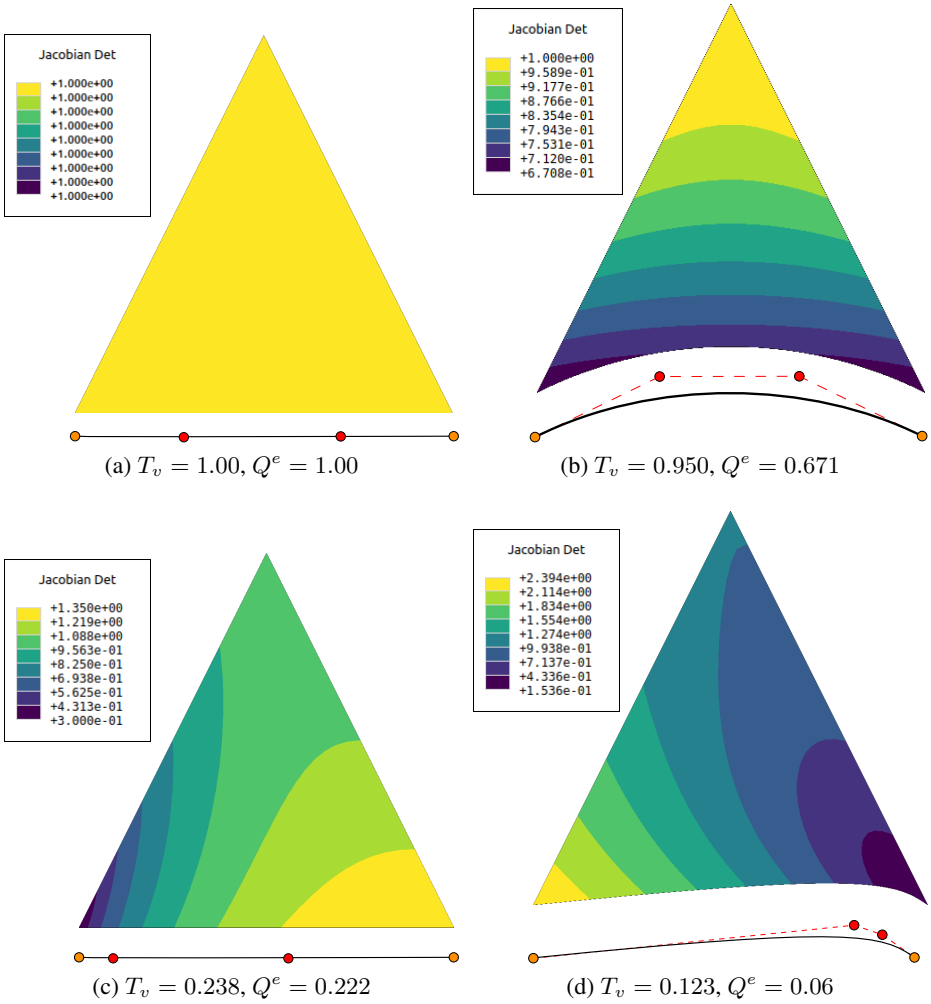
$$T_v = \frac{|T_g^{min}|}{|T_g^{max}|}. \tag{27}$$

Thus, the segment is refined if $T_v < 0.5$. This measure resembles the quality metric scaled Jacobian used in the context of high-order element [17–21], evaluated by:

$$Q^e = \frac{\det(\mathbf{J}^e)_{min}}{\det(\mathbf{J}^e)_{max}}, \tag{28}$$

where $\det(\mathbf{J}^e)_{min}$ and $\det(\mathbf{J}^e)_{max}$ are, respectively, the element minimum and maximum determinant of the Jacobian matrix. A large variation of $|\mathbf{J}^e|$ decreases element performance, even when $|\mathbf{J}| > 0$. Figure 8 illustrates the effect of curved edges parametrization in the variation of the Jacobian matrix determinant over cubic Bézier triangle elements.

It is worth pointing out that this issue does not happen in case of finite element meshes. The parametrization of the geometry model is not inherited by high-order Lagrange edges, where the internal nodes are usually uniformly distributed in physical space (Euclidean distance) of the geometry curve.



(a) $T_v = 1.00$, $Q^e = 1.00$ (b) $T_v = 0.950$, $Q^e = 0.671$

(c) $T_v = 0.238$, $Q^e = 0.222$ (d) $T_v = 0.123$, $Q^e = 0.06$

Figure 8. The effect of curved edges parametrization over cubic Bézier triangle elements. The cases (a) and (b) have good parametrization, while the cases (c) and (d) have bad parametrization.

3.3 Disparity

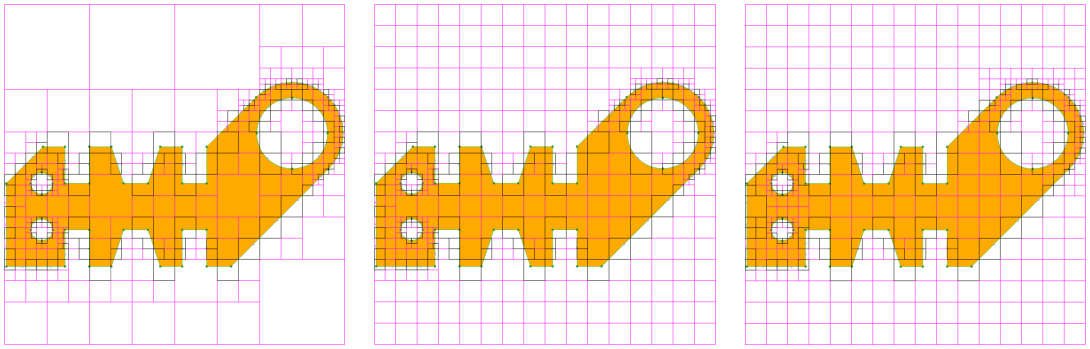
The disparity criterion aims to reduce the differences between the lengths of nearby segments. A quadtree is constructed based on the model edge size, following the same rule presented in [22]:

- **Construction:** The initial size of the quadtree is given by the bounding square of the model. Thus, for each segment of the model, the lengths of its chord s_l and the midpoint of chord s_m are evaluated. The quadtree is refined if $s_l < q_{side}$, where q_{side} is the side of the leaf cell containing the point s_m.
- **Maximum size:** The largest size (q_{max}) of the cells that were refined by the edges of the model, in the previous step, is used to refine the other cells, until q_{max} is the maximum size in all leaf cells.
- **Disparity:** Refinements are applied until the difference between the size of adjacent cells is 1 [11−13].

Figure 9 illustrates the application of these steps in the construction of the quadtree.

Using the processed quadtree, a segment $[t_1, t_2]$ is refined if half of the largest Cartesian component of the vector formed by the segment chord is larger than the smallest side of quadtree cells at the endpoints and midpoint of the segment chord.

The quadtree is constructed after a first application of the maximum length, curvature and proximity criteria. Then the discretization is further refined considering all criteria. It is worth pointing out that the edges of the model previously discretized, but which are not discretized in the execution of the algorithm,



(a) Construction phase. (b) Maximum size phase. (c) Disparity phase.

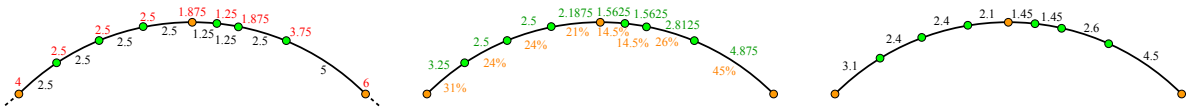
Figure 9. Creation of the quadtree used in the disparity criterion.

also contribute in the construction of the quadtree.

3.4 Smoothing

The discretization carried out in the previous steps does not ensure a smooth transition between adjacent edges. Internally, the discretization obtained in each initial segment can present edges with differences of at least 50% in size (when there is a difference). Larger differences can be observed on the edges adjacent to the vertices prior to the application of the discretization. A poor transition between edges can compromise the mesh generation algorithm, reducing mesh quality. For this reason, a smoothing procedure should be applied.

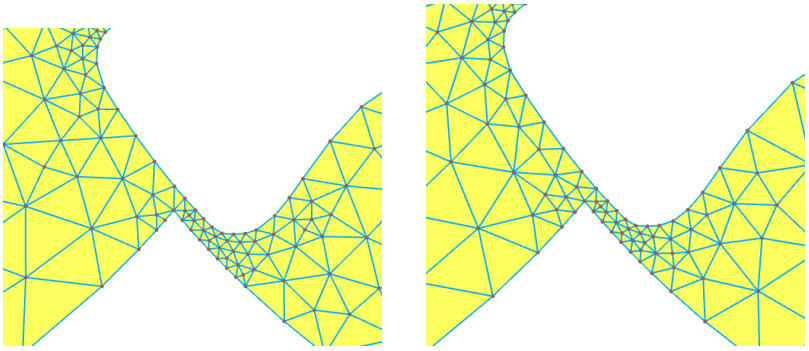
The length of each segment is evaluated. The average edge length is evaluated at each vertex, considering its adjacency, a procedure similar to the evaluation of vertex normals in the context of computer graphics. At each edge, a new edge length (LN_i) is evaluated, given by the average of the lengths on its vertices. The values LN_i do not necessarily maintain the initial curve length. Therefore, the values LN_i are used to calculate a new percentage of the parent curve segment size, that each segment has. Hence, the internal vertices of each segment are moved, reduzing the differences between adjacent segments. Figure 10 shows the aplication of the smoothing procedure.



(a) Evaluation of the edge lengths at vertices. (b) Evaluation of the lengths LN_i. (c) Segments lengths after smoothing process.

Figure 10. Smoothing process. The orange vertices do not move.

The process described is applied twice. It is worth to noting that only vertices generated during the discretization process are moved. The Figure 11 shows the effect of the smoothing step applied to the fourth case of Example 5.2.



(a) Smoothing is not applied. (b) Smoothing is applied.

Figure 11. Effectiveness of the discretization smoothing procedure.

4 Mesh Generation

This section discusses the high-order isogeometric mesh generation algorithm. The input consists of a set of Bézier curves of degree p, with consistent orientation. The input curves are obtained from the geometry model. The Degree Elevation and Knot Insertion are used to evaluate NURBS curves that have the obtained discretization on their knot vectors and degree p. The Bézier curves control points are computed using Bézier extraction.

The algorithm consist of three steps:

- **Linear Mesh Generation:** The initial mesh is constructed by a linear mesh generator, where the mesh topology is defined. Note that any linear mesh that accepts an input front can be used.
- **High-order Mesh:** The high-order mesh is obtained by applying the degree elevation algorithm in the linear mesh.
- **Elasticity Smoothing:** A linear elasticity problem is solved considering prescribed displacements on control points of boundary curved edges. The internal control points are moved in this step.

It is worth pointing out that a similar procedure is presented in other works [21, 25, 26].

4.1 Linear mesh generation

The linear mesh is generated using the advancing front algorithm presented in [22]. The boundary contraction procedure is identical to the one presented in the original work, discussed in the geometry and topological processing stages. The laplacian smoothing and the back-tracking are processed to improve mesh quality. The back-tracking consists of deleting bad elements and their neighborhoods and reprocess the mesh generation algorithm. The quality metric used to detect bad triangles is given by:

$$\alpha = \frac{\gamma_{eq}}{\gamma}, \tag{29}$$

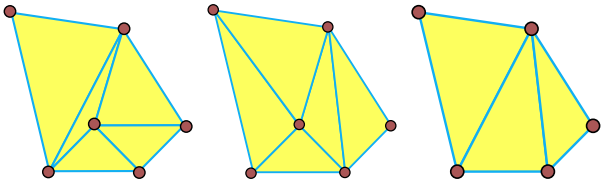
where γ is the ratio between the square of the average edge length of the triangle (L_m) and the triangle area (A_T):

$$\gamma = \frac{L_m^2}{A_T}, \tag{30}$$

$\gamma_{eq} = 4/\sqrt{3}$ is the value obtained for equilateral triangles. The value α varies in the range [0,1], obtaining maximum value for equilateral triangles. A triangle is classified as bad if $\alpha < 2/3$.

The following modification in the original algorithm is adopted. In the back-tracking step, 3 candidate regions are considered to replace the deleted one: The first is the region existing before the deletion; The second is obtained by processing a triangulation, corresponding to the topology processing step of the method; The third is obtained by processing the mesh generation algorithm, using the same quadtree initially evaluated. The mesh with the highest quality value for the worst elements is chosen. Figure 12

illustrates an example of meshes considered in the back-tracking step. It was observed that this modification improved the quality of meshes in complex geometries.

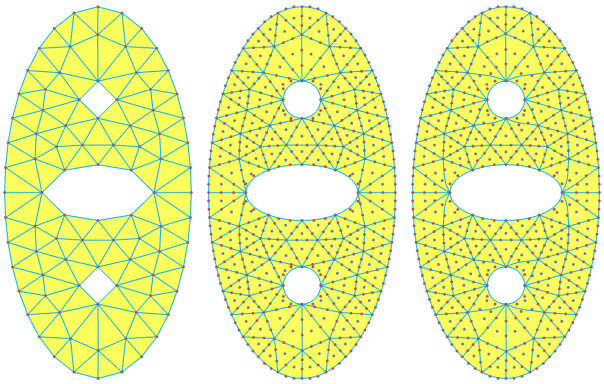


(a) (b) (c)

Figure 12. Meshes considered in the back-tracking stage: (a) Initial region, $\min(\alpha) = 0.26$; (b) advancing front reprocessing, $\min(\alpha) = 0.57$; (c) triangulation, $\min(\alpha) = 0.66$.

4.2 High-order Mesh

The high-order mesh is obtained by applying degree elevation on triangles. The data structure used for mesh storage should be considered in implementing this step. First, the internal control points of the inner edges are evaluated, which are Bézier curves of degree p. The first and last control points of an edge are determined in the linear mesh generation step. Then, the internal control points of each Bézier triangle of degree p are determined, and the incidence of the control points located in the element edges is stored. It is important to note that both degree elevation cases can be performed by linear interpolation since these geometries are linear. Figure 13 illustrates the procedure for obtaining high-order mesh.



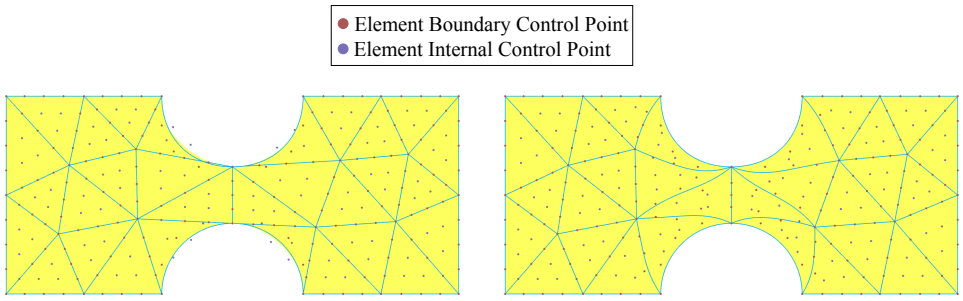
(a) (b) (c)

Figure 13. Procedure to obtaining the high-order mesh: (a) Linear Mesh; (b) Degree Elevation on edges (red control points) and on triangles (purple control points); (c) Final mesh after elasticity smoothing.

4.3 Elasticity Smoothing

The Elasticity smoothing is a technique used to generate a high-order mesh from another corresponding linear one. An elastic analysis is performed applying prescribed displacements to the boundary nodes (or control points), so that the high-order geometry of the geometric model is represented. Linear elastic analysis are usually employed [18, 19]. However, there are also works where incremental linear analyzes [27], nonlinear elastic analyzes [28] and thermo-elastic analyzes [20] are applied.

In this work, linear elastic smoothing is employed to improve the quality of the generated Bézier triangles. Restoring curved mesh edges without any mesh improvement step can result invalid elements. This effect tends to be worse as the curvature of the edge and the element degree increase, as shown in Figure 14. On the other hand, no invalid elements are observed when curvature at elements edge are small.



(a) Without elasticity smoothing. (b) With elasticity smoothing.

Figure 14. Boundary replacement of a quartic degree mesh with and without elasticity smoothing.

5 Examples

In this section the proposed algorithm is applied in three complex geometric models. The output discretization is assessed based on the quality of high-order meshes generated using that as input. Although it is possible to generate elements of any degree, cubic meshes are adopted in all cases. The minimum (Q_{min}) and mean ($\overline{Q}$) values of the scaled Jacobian quality metric, presented in Equation 28), are used to evaluate the high-order meshes. In addition, the minimum (α_{min}) and average ($\overline{\alpha}$) values of the linear metric presented in Equation 29 are also evaluated considering the triangles formed by the control points located on the corners of the high-order triangles. Despite the quality of the meshes depends on used meshing algorithm, the discretization adopted directly influenced its performance.

5.1 Plate

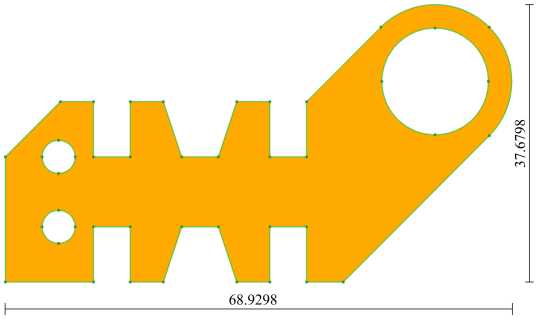
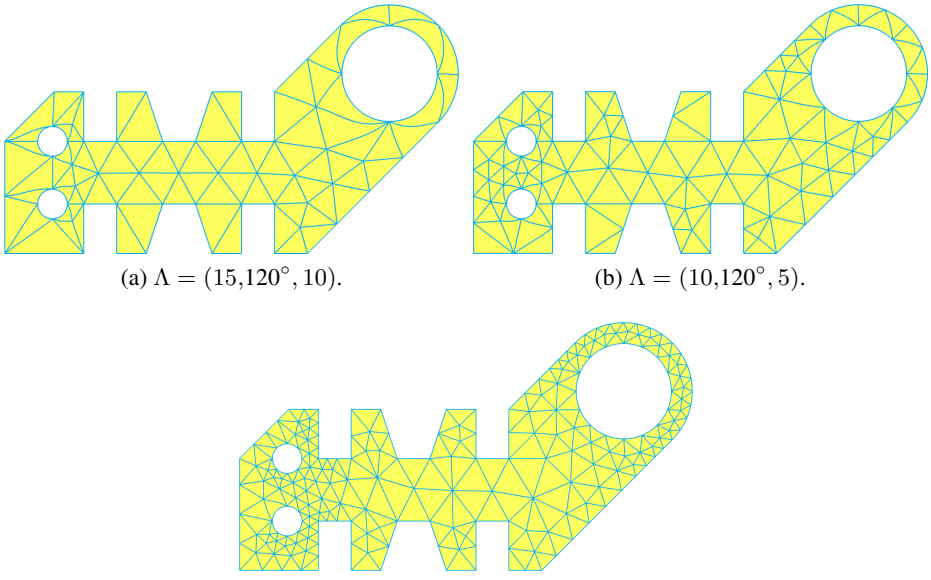


Figure 15. Geometry of Example 5.1.

This example consisting of a plate was originally presented in [3, 24]. The model is represented by a set of linear and quadratic rational Bézier curves, as shown in Figure 15. Three discretizations are processed using different set of parameters $\Lambda = (L_{max}, \theta_{max}, L_{min})$. Table 1 shows the number of elements N_{elm} and mesh quality obtained for each discretization. Figure 16 illustrates the meshes processed for each case.

Table 1. Results obtained in Example 5.1.

L_{max}	θ_{max}	L_{min}	N_{elm}	$\overline{\alpha}$	α_{min}	$\overline{Q}$	Q_{min}
15	120°	10	86	0.8744	0.4202	0.8280	0.0000
10	120°	5	145	0.9253	0.6845	0.8846	0.5425
5	120°	0	333	0.9382	0.6819	0.9145	0.5784



(a) $\Lambda = (15{,}120^\circ, 10)$. (b) $\Lambda = (10{,}120^\circ, 5)$.

(c) $\Lambda = (5{,}120^\circ, 0)$.

Figure 16. Meshes obtained in Example 5.1.

The results show that a regular linear quality was obtained even in coarse meshes, where $\overline{\alpha} = 0.8744$ and $\alpha_{min} = 0.4202$. On the other hand, the first case has invalid high-order elements ($Q_{min} = 0$), indicating that a minimum discretization level is required for high-order mesh. Good quality is obtained on second and third cases, where $Q_{min} = 0.5425$ and $Q_{min} = 0.5784$, respectively. Narrow regions present a higher element density due to the application of the proximity criterion. It is worth noting that $L_{min} = 0$ allows a suitable refinement of all narrow regions of the model.

5.2 Two Rotors

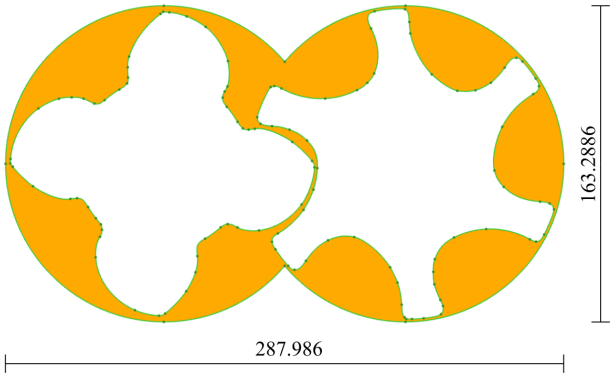


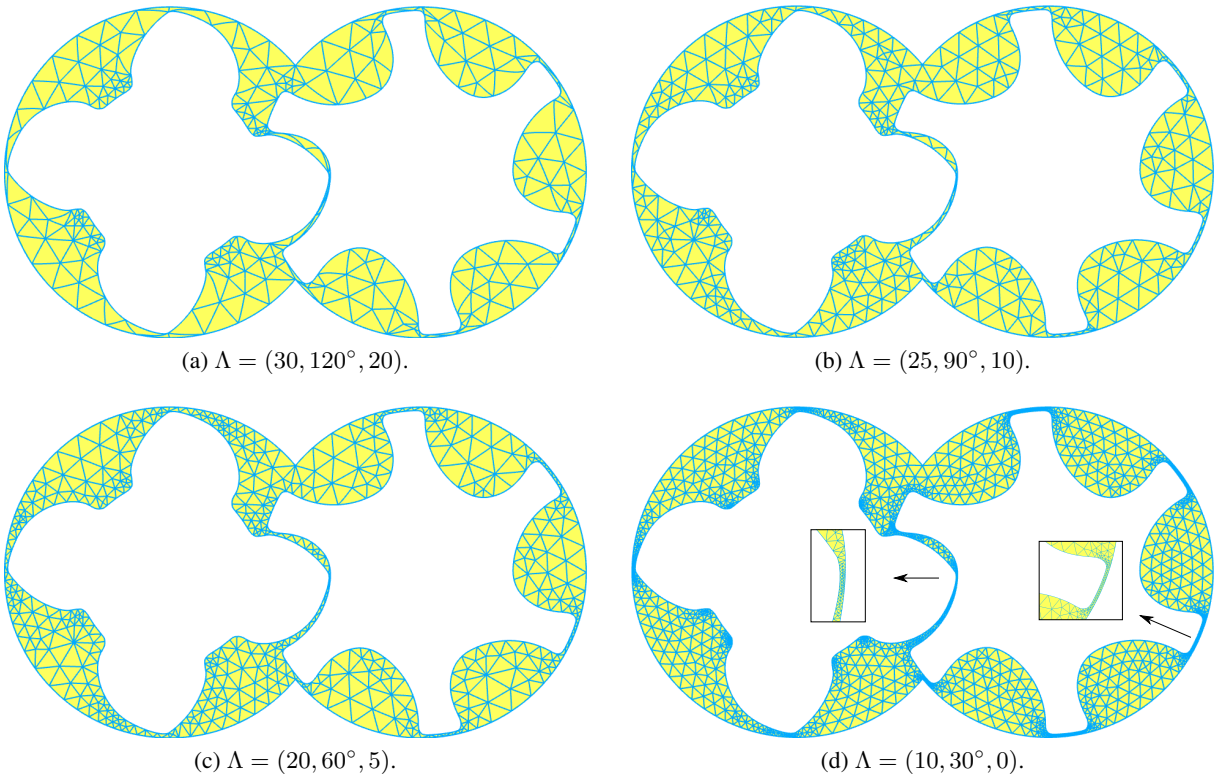
Figure 17. Geometry of Example 5.2.

This example deals with the fluid between two rotors, based on the problem presented in [29]. The geometry of the model shown in Figure 17 is composed of quadratic and cubic Bézier curves. Table 2 presents the results obtained for four sets of parameters, chosen to increase the discretization of the model. Figure 18 illustrates the meshes obtained for each discretization.

Unlike the previous example, poorly refined meshes yield low-quality elements even in linear metric ($\alpha_{min} = 0.0643$). It can be explained by the presence of complex features in the model geometry. Further discretization of the boundary allowed a consistent improvement in α_{min} and Q_{min}, between the first case to the third, increased by 0.2026 and 0.2551, and between the first case to the fourth case, increased by 0.4256 and 0.4972. On the other hand, there is an expressive raise in the number of

Table 2. Results obtained in Example 5.2.

L_{max}	θ_{max}	L_{min}	N_{elm}	$\overline{\alpha}$	α_{min}	$\overline{Q}$	Q_{min}
30	120°	20	372	0.8264	0.0643	0.7748	0.0000
25	90°	10	590	0.8828	0.1366	0.8442	0.0009
20	60°	5	867	0.9120	0.2669	0.8823	0.2551
10	30°	0	2772	0.9456	0.4899	0.9457	0.4972



(a) $\Lambda = (30, 120°, 20)$. (b) $\Lambda = (25, 90°, 10)$.

(c) $\Lambda = (20, 60°, 5)$. (d) $\Lambda = (10, 30°, 0)$.

Figure 18. Meshes obtained in Example 5.2.

elements, increasing in relation to the first mesh by 133% in the third case and by 645% in the fourth case. The choice of the best mesh depends on what level of precision and computational cost is desired in the numerical application.

5.3 Guitar

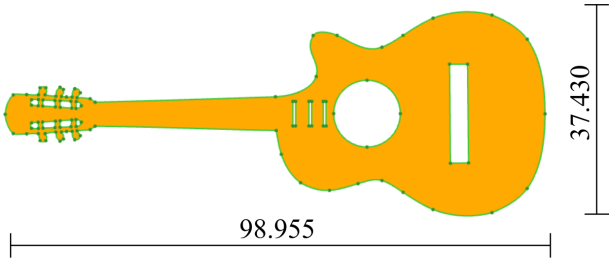


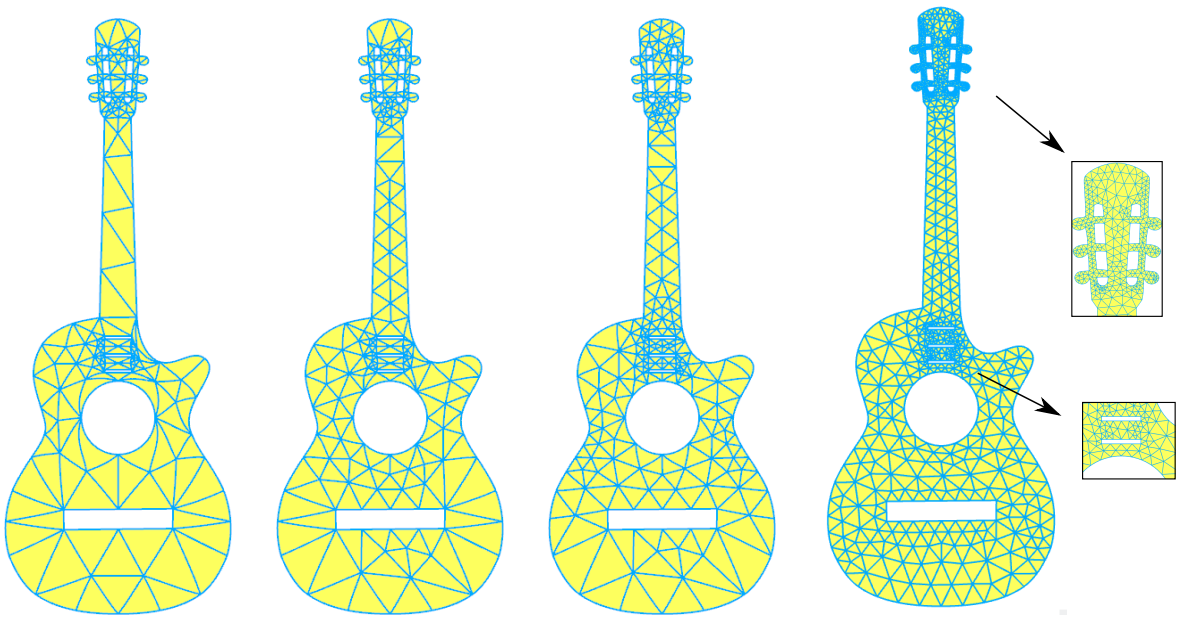
Figure 19. Geometry of Example 5.3.

The last example consists of a guitar, modeled by cubic Bézier curves, as shown in Figure 19. Table 3 presents the results obtained for four set of parameters. The meshes obtained in each discretization are

presented in Figure 20.

Table 3. Results obtained in Example 5.3.

L_{max}	θ_{max}	L_{min}	N_{elm}	$\overline{\alpha}$	α_{min}	$\overline{Q}$	Q_{min}
12	120°	8	251	0.8256	0.1484	0.7903	0.0065
12	90°	4	309	0.8652	0.3801	0.8428	0.0002
8	90°	2	411	0.9046	0.4655	0.8810	0.0001
4	60°	0	1280	0.9455	0.5620	0.9512	0.5370

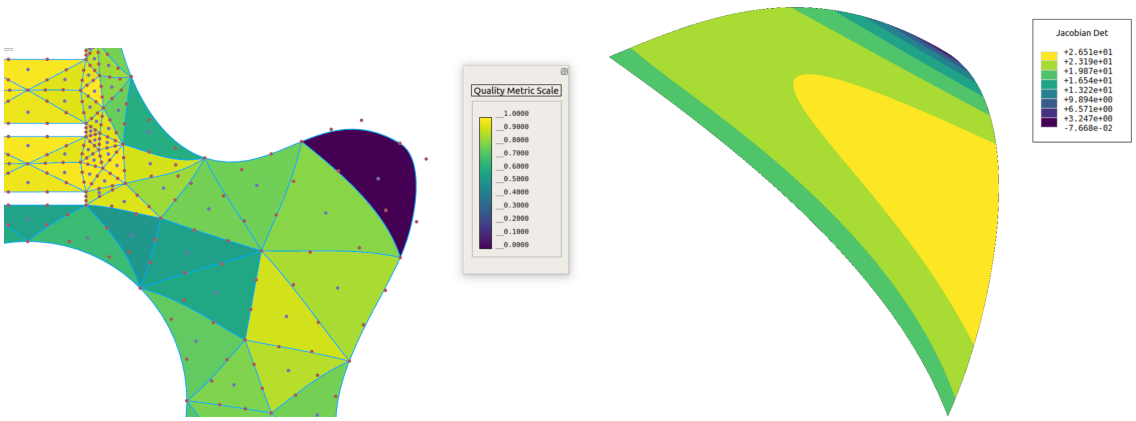


(a) $\Lambda = (12, 120°, 8)$. (b) $\Lambda = (12, 90°, 4)$. (c) $\Lambda = (8, 60°, 2)$. (d) $\Lambda = (4, 60°, 0)$.

Figure 20. Meshes obtained in Example 5.3.

Similar to the previous examples, it is observed improvement on both avarege metric measures ($\overline{\alpha}$ and $\overline{Q}$) as the model discretization increases. The linear metric α_{min} improves between the first and third case by 0.3171, and between the first and fourth case by 0.4136, respectively. The number of element increased by 64% between the first and third case and 410% between the first and fourth case.

The high-order metric Q_{min} kept close to the minimum bound in the three initial cases due to the presence of elements with two curved edges of the boundary, as shown in Figure 21. These elements may have excessive perturbation in their Jacobian transformation on the neighborhood of the common corner. It is not possible to increases the quality consistently in this case without changing mesh topology, where it must be guaranteed that no element is adjacent to more than one curved edge of the boundary. Thus, high-order mesh generation algorithms should avoid this topology feature. This issue has been discussed in the context of isogeometric mesh generation of Bézier tetrahedra and pyramid [30].



(a) Bad quality element ($Q^e = 0$). (b) Variation of $|\mathbf{J}|$.

Figure 21. Bad quality element due to the incidence of two curved edges of the boundary.

6 Conclusions

This work presented an algorithm for automatic, high-quality boundary discretization for isogeometric or finite element mesh generation. The level of discretization is given by the desired edge size, but it can be also based on complex features, as in locations with high curvature, in narrow regions, where boundary curves are close, and also due the parametrization of the model curves. Futhermore, the level of discretization is controlled by three input parameters, allowing to control the number of elements generated during the mesh generation step. It is important to note that the algorithm can be applied in all curves of the model or in a subset of them, in order to preserve pre-existing discretizations, as in the case of models with multiple regions with processed meshes.

The obtained discretization is used in the generation of high-order isogeometric meshes composed by rational Bézier triangles. The use of the algorithm is demonstrated using three examples, which present complex features. The output meshes were used to evaluate the obtained discretizations, based on mesh quality. A set of parameters were used to obtain models with different discretization levels. Better element quality was obtained as the discretization level increases, good quality meshes were obtained in all examples.

In future work, the proposed algorithm can be extended to the three-dimensional case, where the model is given by parametric surfaces. In this case, recursive discretization shall be made on both model surfaces and curves.

Acknowledgements

The authors acknowledge the financial support provided by CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), PDSE (Programa de Doutorado Sanduíche no Exterior) process 88881.190187/2018-01.

References

[1] Geuzaine, C. & Remacle, J. F., 2009. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, vol. 79, n. 11, pp. 1309–1331.

[2] Ahrens, J., Geveci, B., & Law, C., 2005. ParaView: An End-User Tool for Large Data Visualization. In Hansen, C. D. & Johnson, C. R., eds, *the Visualization Handbook*, pp. 962. Elsevier Butterworth-Heinemann.

[3] Persson, P. O., 2006. Mesh size functions for implicit geometries and PDE-based gradient limiting. *Engineering with Computers*, vol. 22, n. 2, pp. 95–109.

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

[5] Stroud, I., 2006. *Boundary Representation Modelling Techniques*. Springer.

[6] Piegl, L., 1991. On NURBS: A Survey. *IEEE Computer Graphics and Applications*, vol. 11, n. 1, pp. 55–71.

[7] Piegl, L. & Tiller, W., 1995. *The NURBS Book*. Monographs in Visual Communications. Springer Berlin Heidelberg, Berlin, Heidelberg.

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

[9] Scott, M. A., Borden, M. J., Verhoosel, C. V., Sederberg, T. W., & Hughes, T. J., 2011. Isogeometric finite element data structures based on Bézier extraction of T-splines. *International Journal for Numerical Methods in Engineering*, vol. 88, n. 2, pp. 126–156.

[10] Thomas, D. C., Scott, M. A., Evans, J. A., Tew, K., & Evans, E. J., 2015. 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*, vol. 284, pp. 55–105.

[11] Farin, G., 2002. *Curves and Surfaces for CAGD A Practical Guide*, volume 3. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition.

[12] Mainar, E. & Peña, J. M., 2006. Evaluation algorithms for multivariate polynomials in Bernstein-Bézier form. *Journal of Approximation Theory*, vol. 143, n. 1, pp. 44–61.

[13] Frey, P. J. & Maréchal, L., 1998. Fast Adaptive Quadtree Mesh Generation. In *Proc. 7Th International Meshing Roundtable, Sandia National Laboratories*, pp. 211—-224, Dearborn.

[14] Miranda, A. C. O., Lira, W. W. M., Cavalcante-Neto, J. B., Sousa, R. A., & Martha, L. F., 2013. A Three-Dimensional Adaptive Mesh Generation Approach Using Geometric Modeling With Multi-Regions and Parametric Surfaces. *Journal of Computing and Information Science in Engineering*, vol. 13, n. 2, pp. 021002.

[15] McLaurin, D. & Shontz, S. M., 2014. Automated Edge Grid Generation Based on Arc-Length Optimization. In *Proceedings of the 22nd International Meshing Roundtable*, pp. 385–403. Springer International Publishing, Cham.

[16] Pinto, F. D. M. & Freitas, C. M. D. S., 2009. Fast medial axis transform for planar domains with general boundaries. In *Proceedings of SIBGRAPI 2009 - 22nd Brazilian Symposium on Computer Graphics and Image Processing*, pp. 96–103. IEEE.

[17] Dey, S., O'Bara, R. M., & Shephard, M. S., 2001. Towards curvilinear meshing in 3D: The case of quadratic simplices. *CAD Computer Aided Design*, vol. 33, n. 3, pp. 199–209.

[18] Xie, Z. Q., Sevilla, R., Hassan, O., & Morgan, K., 2013. The generation of arbitrary order curved meshes for 3D finite element analysis. *Computational Mechanics*, vol. 51, n. 3, pp. 361–374.

[19] Abgrall, R., Dobrzynski, C., & Froehly, A., 2014. A method for computing curved meshes via the linear elasticity analogy, application to fluid dynamics problems. *International Journal for Numerical Methods in Fluids*, vol. 76, n. 4, pp. 246–266.

[20] Moxey, D., Ekelschot, D., Keskin, U., Sherwin, S. J., & Peiró, J., 2016. High-order curvilinear meshing using a thermo-elastic analogy. *CAD Computer Aided Design*, vol. 72, pp. 130–139.

[21] Engvall, L. & Evans, J. A., 2016. Isogeometric triangular Bernstein-Bézier discretizations: Automatic mesh generation and geometrically exact finite element analysis. *Computer Methods in Applied Mechanics and Engineering*, vol. 304, pp. 378–407.

[22] Miranda, A. C., Cavalcante Neto, J. B., & Martha, L. F., 1999. 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*, pp. 29–38. IEEE Comput. Soc.

[23] Cavalcante Neto, J., Carvalho, M., & Martha, L., 1993. Combinação de Técnicas de Quadtree e Delaunay para Geração Automática de Malhas de Elementos Finitos. In *VI Brazilian Symposium on Computer Graphics and Image Processing*, pp. XII Brazilian Symposium on Computer Graphics and I, Recife, Pernambuco.

[24] Freitas, M. O., Wawrzynek, P. A., Cavalcante-Neto, J. B., Vidal, C. A., Martha, L. F., & Ingraffea, A. R., 2013. A distributed-memory parallel technique for two-dimensional mesh generation for arbitrary domains. *Advances in Engineering Software*, vol. 59, pp. 38–52.

[25] Jaxon, N. & Qian, X., 2014. Isogeometric analysis on triangulations. *CAD Computer Aided Design*, vol. 46, n. 1, pp. 45–57.

[26] Liu, N. & Jeffers, A. E., 2018. A geometrically exact isogeometric Kirchhoff plate: Feature-preserving automatic meshing and C1 rational triangular Bézier spline discretizations. *International Journal for Numerical Methods in Engineering*, vol. 115, n. 3, pp. 395–409.

[27] Poya, R., Sevilla, R., & Gil, A. J., 2016. A unified approach for a posteriori high-order curved mesh generation using solid mechanics. *Computational Mechanics*, vol. 58, n. 3, pp. 457–490.

[28] Persson, P.-O. & Peraire, J., 2013. Curved Mesh Generation and Mesh Refinement using Lagrangian Solid Mechanics. In *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, Reston, Virigina. American Institute of Aeronautics and Astronautics.

[29] Hinz, J., Möller, M., & Vuik, C., 2018. Elliptic grid generation techniques in the framework of isogeometric analysis applications. *Computer Aided Geometric Design*, vol. 65, pp. 48–75.

[30] Engvall, L. & Evans, J. A., 2017. Isogeometric unstructured tetrahedral and mixed-element Bernstein–Bézier discretizations. *Computer Methods in Applied Mechanics and Engineering*, vol. 319, pp. 83–123.