Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 00 (2014) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

23rd International Meshing Roundtable (IMR23)

Quadrilateral Mesh Generation with Deferred Constraint Insertion

Chrystiano Araújo, Waldemar Celes

Tecgraf/PUC-Rio Institute
Computer Science Department
Pontifical Catholic University of Rio de Janeiro, Brazil

Abstract

In this paper, we present a new method for quadrilateral mesh generation in complex 2D domains, based on deferred constraint insertion. Our method is especially designed to handle complex geological cross-sections and multi-fractured media. We first introduce a new algorithm for triangular mesh generation: starting with a initial mesh, in general simple and regular, we adaptively modify it in order to accommodate each constraint, despite its complexity. We then propose a set of heuristics to convert the achieved triangular mesh into a quadrilateral one, pursuing the alignment of elements along constraints. As a result, we achieve a very stable, efficient, good-quality, and automated mesh generator. We demonstrate the robustness of our proposal by showing resulting triangular and quadrilateral meshes for a set of geological domains with complex line constraints.

Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

Keywords: quadrilateral mesh; triangular mesh; mesh with constraints;

1. Introduction

Important engineering applications need to make domain discretization using quadrilateral meshes. Quadrilateral elements, when compared to triangular ones, tend to be more numerically stable and to require less mesh refinement. However, quadrilateral mesh generation for complex geological domains remains challenging. Even the generation of triangular meshes may impose difficulties when dealing with highly complex domains. In this work, we are particularly interested in the generation of meshes for multi-layered and multi-fractured geological models.

Methods for quadrilateral mesh generation can be classified in two large groups: indirect and direct methods. Both present advantages and disadvantages. In indirect methods, an intermediate triangular mesh is first generated to be later converted to a quadrilateral one. In direct methods, the quadrilateral mesh is generated in the first place. Direct methods tend to result in meshes with better quality; however, it is hard to conceive a robust direct method to handle complex line constraints. In practice, when applied to complex geological domains as the ones we investigate in this work, direct methods usually rely on user intervention to guide the solution, ending up as a laborious task.

Waldemar Celes. Tel: +55-21-3520-4330
E-mail address: celes@tecgraf.puc-rio.br

Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

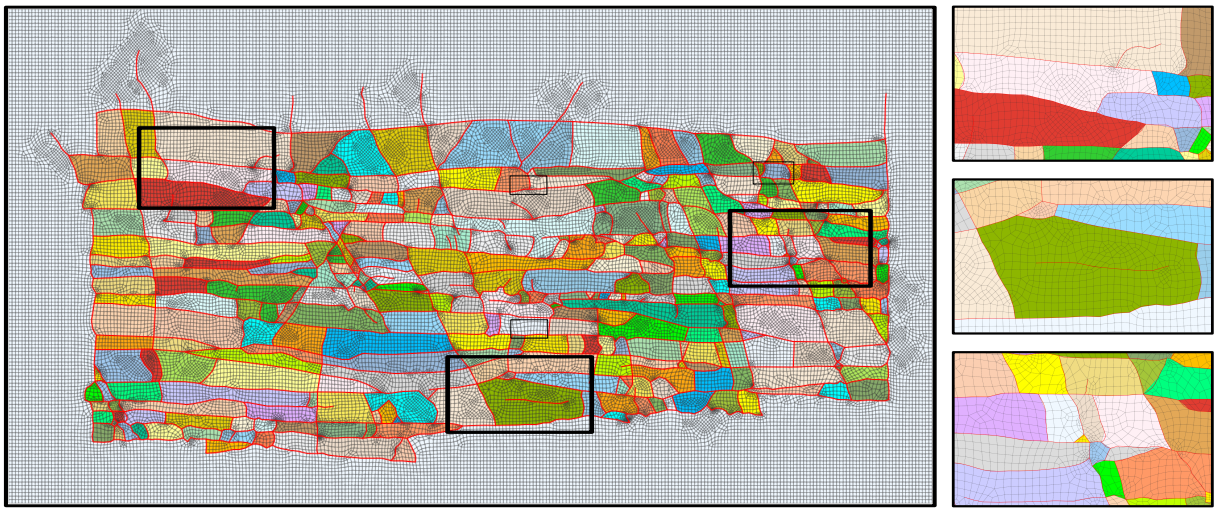


Fig. 1: Fully automated, quadrilateral mesh generation for a fractured domain (48,674 elements).

In this paper, we present a new indirect method for quadrilateral mesh generation. Our algorithm is capable of dealing with complex line constrains. The term *complex line constraints*, in this work, means polylines with complex geometry, including dangling polylines in the interior of domain regions. First, we present a new triangular mesh generation algorithm, based on a deferred constraint insertion strategy. The proposed algorithm is an iterative procedure: we start with an initial valid, in general simple and regular, mesh that is locally adapted in order to accommodate the insertion of constraints, one at a time. As a result, we achieved a robust, efficient, and completely automated algorithm – no user intervention is required, besides the choice of an initial mesh. Most importantly, this strategy preserves the global structure of the initial mesh, significantly favouring conversion to quadrilaterals. The quadrilateral mesh is generated by first processing the achieved triangular mesh in order to align elements along constraints and then applying a triangle-pair clustering strategy. Figure 1 illustrates a generated mesh for an actual geological multi-fracture section. Note that all input constraints are honored, even in local regions with complex geometry.

2. Related Work

In this section, we discuss some previous works on quadrilateral mesh generation, using both direct and indirect methods.

2.1. Direct methods

Direct methods are usually based on two main techniques: domain decomposition and advancing front. In the first group, domains are decomposed into convex or mappable regions, and mapping methods are used to generate a quadrilateral mesh in each region. Baehmann et al. [1] decomposed the domain in simple geometries using a quadtree data structure. The works by Nowottny [2] and Chae and Jeong [3] used a recursive decomposition approach as initially proposed by Talbert and Parkinson [4]: complex regions are recursively subdivided until new simple regions are achieved. However, these methods are all limited to closed boundaries. In order to overcome this limitation, Lin et al. [5] recently proposed a new method for domain decomposition capable of dealing with line constraints (open and closed boundaries). The subregions were generated in a way that line constraints became boundaries of the generated regions. They then used the paving algorithm proposed by Blacker and Stephenson [6] to generate the mesh in each region.

Advancing front methods work in a different way. The boundary edges represent the initial front set, and new elements are inserted in the mesh from the boundary to the interior, advancing the front, until the whole domain are

filled with elements. The paving method, by Blacker and Stephenson [6], is an example of advancing front technique, being later modified by White and Kinney [7]. Cass et al. [8] extended the paving method for 3D surfaces. More recently, Park et al. [9] proposed an advancing front method for handling line constraints, but their proposal is limited to simple domains.

In general, direct methods based on domain decomposition are capable of generating good quality meshes (as a result of mapping procedures); however, for complex constraint geometries, domain decomposition usually requires laborious user intervention. On the other hand, advancing front methods may require less user intervention, but is also challenging for complex domains, in particular to handle auto intersections in the front set.

2.2. Indirect methods

Indirect methods first create a triangular mesh and then convert it into a quadrilateral one. In order to generate the triangular mesh, the conventional constrained Delaunay triangulation is commonly employed. The quadrilateral elements are then generated by combining or subdividing triangles, preserving boundary edges. The simplest strategy for converting a triangular mesh into a quadrilateral one was proposed by Catmull-Clark [10]: each triangle is subdivided into three quadrilaterals by inserting extra nodes, one at the triangle centroid and one at each middle side. The resulting quadrilateral mesh is consistent by construction; however, the quality of the mesh tends to be poor (quadrilaterals with bad aspect ratio and vertices with high valence). Liu et al. [11] proposed a post-processing phase after applying Catmull-Clark, using topological operations and mesh smoothing to improve element quality. More recently, Liu et al. [12] proposed to incorporate geodesic isolines around line constraint, guiding the arrangement of triangles to create better quadrilateral elements along boundaries.

Johnston et al. [13] and Lee and Lo [14] opted for generating the quadrilateral elements by combining existing triangles. Borouchaki and Frey [15] proposed to combine pairs of triangles to generate quadrilaterals; in the end, non-paired triangles are subdivided using Catmull-Clark. In a similar way, Velho [16] proposed to generate quadrilaterals by the application of a hybrid 4-8 subdivision and clustering. First, adjacent triangles are pair-clustered, generating four quadrilaterals with the insertion of mid-side nodes. Then, isolated triangles are subdivided using Catmull-Clark.

The method named Q-morph, presented by Owen et al. [17], is probably the best known indirect method in the literature. The authors proposed to generate quadrilaterals by combining triangles guided by an advancing front strategy. The original proposal was limited to closed boundary. Later, Lee et al. [18] extended Q-morph to deal with line constraints. More recently, the use of cross-field to perform surface parametrisation has emerged [19–21], introducing new strategies for converting triangular to quadrilateral meshes.

We have also opted for an indirect method for quadrilateral mesh generation. However, in order to handle complex line constraints while preserving the global structure of a given initial mesh, we introduce a new algorithm for triangular mesh generation, based on deferred constraint insertion. We employ the triangle-pair clustering strategy to convert the achieved triangular mesh to a quadrilateral one, presenting a set of heuristics to minimize the occurrence of isolated triangles close to constraints, thus favouring quadrilaterals following region boundaries.

3. Proposed method

Our method expects as input a set of constraints $C = \{C_1, C_2, ..., C_n\}$ that models the domain. The method does not differentiate external from internal constraints. Each constraint is represented by a polygonal line, being delimited by two endpoints $\vec{e}_i$ with zero or more midpoints $\vec{m}_i$: $C = \{\vec{e}_0, \vec{m}_1, \vec{m}_2, ..., \vec{m}_k, \vec{e}_1\}, k \geq 0$. There should be no crossing constraints, i.e., any two constraints are disjoint or meet at endpoints. The current version of the algorithm does not handle closed polygonal lines. Closed lines must be represented by at least two polygonal constraints.

Each constraint must be honored in the final mesh. The endpoints must be mapped to vertices (nodes) in the mesh, and the polygonal geometry defined by the midpoints must be covered by a sequence of edges in the mesh, within an approximation error controlled by a given threshold. The input is assumed valid; so, despite how close constraints are from each other, the proposed method does honor them, generating consistent mesh in the delimited domain regions. In geological domains, as we shall see, several very small regions are usually defined in between constraint lines, and we want our proposal to handle such a configuration smoothly.

As an indirect method, our proposal first generates a triangular mesh that is then converted to a quadrilateral one. We use the TopS data structure [22, 23] for mesh representation and edition. The following sections describe our proposal in detail.

3.1. Triangular mesh generation

We propose to generate the triangular mesh using a strategy based on *deferred constraint insertion*. Our algorithm starts with a valid initial (in general simple and regular) mesh. Each constraint is then inserted, and the mesh is locally modified to accommodate the constraint. In modifying the mesh, our proposal ensures that the mesh quality is preserved, as long as possible, by locally applying controlled topological refinement and smoothing operations.

The pseudo-code shown in Algorithm 1 provides an overview of the proposed method.

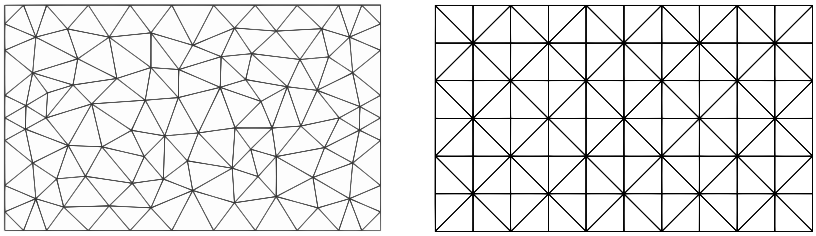
```
Data: Set of constraints: C = {C_1, C_2, ..., C_n}
Result: Triangular mesh: M_final = (V, E, T)
Set an initial mesh M_initial = (V, E, T);
for each constraint C_k ∈ C do
    for each endpoint e⃗_i ∈ {e⃗_0, e⃗_1} do
        v⃗_best ← best mesh candidate vertex for attraction;
        while attraction would degrade quality do
            Refine mesh locally;
            v⃗_best ← best candidate for attraction;
        end
        v⃗_i ← attract v⃗_best to e⃗_i
    end
    v⃗_curr ← v⃗_0;
    while v⃗_curr ≠ v⃗_1 do
        Find first intersection between C'_k = {v⃗_curr, ..., e⃗_1} and mesh;
        v⃗_best ← best mesh candidate vertex for attraction;
        while attraction would degrade quality do
            Refine mesh locally;
            Find first intersection on modified mesh;
            v⃗_best ← best candidate for attraction;
        end
        v⃗_curr ← attract v⃗_best to C'_k
    end
end
```

Algorithm 1: Overview of the proposed algorithm

3.1.1. Initial mesh

The algorithm can start with *any* valid mesh, as long as the mesh fills the entire domain. For instance, one can choose to generate an initial mesh defined in the domain bounding box. Even if the domain has complex boundary defined by a set of constraints, we can later identify and discard elements in the mesh that are outside the domain (for instance, applying a flood-fill algorithm on the generated mesh).

Nonetheless, the initial mesh does influence the final result, and we should choose it with discretion. If the ultimate goal is to generate a triangular mesh, we recommend to use a good-quality initial mesh; for instance, we can use a constraint-free Delaunay triangulation. However, if the goal is to generate a quadrilateral mesh, we argue, as we shall discuss, that a good choice is to start with a triangular 4-8 mesh, using a crisscross pattern [16]. This mesh aligns triangles in a way that favours pair clustering for quadrilateral generation. Moreover, the initial mesh orientation is freely choosen, and one can align the crisscross pattern according to the desired result. Figure 2 illustrates initial mesh patterns, Delaunay and 4-8.



(a) Delaunay (b) 4-8

Fig. 2: Initial mesh patterns.

The discretization of the initial mesh must reflect the desired minimal discretization. In fact, the algorithm will refine the mesh to accommodate the constraints. Regions close to complex constraints may become quite refined, but regions free of constraints or close to simple constraints will basically keep the original discretization. One can also employ adaptive discretization in the initial mesh in order to fulfil application criteria (for instance, based on a density function defined over the domain). As mentioned, any initial valid mesh can be employed.

3.1.2. Insertion of endpoints

Given an initial mesh, the proposed algorithm inserts each constraint at a time, adjusting the mesh as needed. In the end, the generated triangular mesh honors all the constraints. For each constraint to be inserted, the first step consists in inserting the two endpoints in the mesh; each endpoint must be represented by a mesh vertex after insertion.

Both constraint endpoints are inserted following the same procedure. For each endpoint, first we locate the mesh triangle that contains the point. Then, the three incident vertices of this triangle are eligible to be attracted to the endpoint position. The *best* candidate vertex is chosen, among the three incident vertices, except vertices already attracted to previous constraints, based on a criterion that minimises mesh distortion due to the attraction. In our proposal, mesh distortion is measured by first computing the lowest quality value among the triangles adjacent to the vertex ($Adj(\vec{v}_i)$), before ($\phi^{(0)}_{min}$) and after ($\phi^{(1)}_{min}$) attraction (we may have values from different triangles):

$$\phi^{(0,1)}_{min} = \min_{T_j \in Adj(\vec{v}_i)^{(0,1)}} \phi(T^{(0,1)}_j) \tag{1}$$

In this work, we use Lo parameter to measure triangle quality:

$$\phi(T_j) = 4\sqrt{3}\frac{A}{\sum_{k=1}^{3} l_k^2} \tag{2}$$

where A represents the triangle area and l_k is the length of each triangle edge. This parameter value varies from 0.0 (zero-area triangle) to 1.0 (equilateral triangle). The *mesh distortion* associated to attracting the vertex is then measured by how the lowest triangle quality is degraded:

$$d(\vec{v}_i) = \frac{\phi^{(0)}_{min} - \phi^{(1)}_{min}}{\phi^{(0)}_{min}} \tag{3}$$

The best candidate is then the one with minimum distortion:

$$\vec{v}_{best} = \vec{v}_i \mid d(\vec{v}_i) = \min_{\vec{v}_j \in V_{cand}} d(\vec{v}_j) \tag{4}$$

where V_{cand} represents the set of candidate vertices (in this case, the vertices of the triangle that contains the given endpoint). More importantly, the best candidate ($\vec{v}_{best}$) can only be attracted if the following conditions apply:

- The quality of any adjacent triangle, after attraction, cannot be less than a given threshold.

$$\min_{T_j \in Adj(\vec{v}_{best})^{(1)}} \phi(T_j^{(1)}) \geq q_{min} \tag{5}$$

- The mesh distortion due to the attraction cannot exceed a given threshold:

$$d(\vec{v}_{best}) \leq d_{max} \tag{6}$$

If the attraction of the best candidate does not obey these conditions, the mesh is locally refined by splitting the triangle that contains the endpoint. This is performed by applying a recursive use of the longest-edge bisection operation. However, in order to minimise refinement spread, we slightly modified the recursive procedure. Instead of only splitting the longest edge, we allow *close-to-longest* edges to be split: if the length of an edge is greater then 80% of the longest edge length of the triangle, it is valid to split the edge. This modification tends to localise refinement while preserving good mesh quality.

The attraction evaluation and local mesh refinement are repeated until a valid candidate is found. Sometimes, however, the set of constraints imposes very complex local subdivision. In such scenarios, we have observed that the maximum distortion criterion may not be possible to be honored, or would impose an arbitrarily high degree of refinement. To handle this, we have opted to relax the criterion of maximum distortion at each iteration of this search for a valid candidate: the threshold d_{max} is decreased by a given *relaxation factor*, r_{factor}, at each iteration. Once a valid candidate is found, the original threshold value is restored for subsequent mesh operations.

Based on computational experiments, considering different domains, in the current version of the algorithm, we have fixed these parameter values as: $q_{min} = 0.1$, $d_{max} = 0.35$, and $r_{factor} = 0.9$. The main purpose of the minimum quality criterion (q_{min}) is to avoid degenerated or inverted triangles. As we shall discuss, triangle quality is later improved by a local smoothing operation.

3.1.3. Insertion of polygonal line

Once the two constraint endpoints are inserted in the mesh (and mapped to mesh vertices), the goal now is to adjust the mesh in order to have a sequence of mesh edges representing the constraint path, within an approximation error limited by a threshold. This is implemented by an iterative procedure.

Starting with the first vertex ($\vec{v}_{curr} \leftarrow \vec{v}_0$), we find the closest intersection point between the constraint line and the set of opposite mesh edges of the current vertex. This point of intersection, as illustrated in Figure 3a, represents the point to where a mesh vertex must be attracted to. The candidates are the vertices of the crossed edge, except if they are already part of a constraint. Similar to the procedure used in the insertion of endpoints, we compute mesh distortion associated to each candidate. If the attraction of the best candidate does not violate the quality criteria (Equations 5 and 6), it is moved to the point of intersection and becomes the current vertex. If the best candidate fails, the crossed edge is split, again using the modified recursive close-to-longest-edge bisection operation. The search for a valid attraction is then repeated.

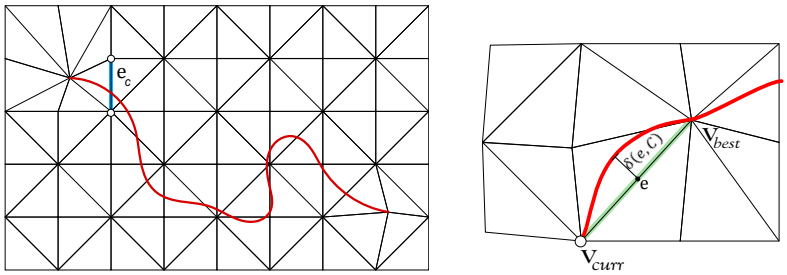
Once a vertex is attracted, the algorithm goes through the same procedure again, finding a next current vertex, until the target vertex is reached. In the end, we have a set of mesh edges covering the constraint path.

Note however that the quality criteria in use do not ensure that the sequence of edges in the final mesh will represent the polygonal line within an approximation tolerance. In order to fix that, we added a third quality criterion when attracting vertex to the polygonal line:

- The greatest distance between the constraint segment, from the current vertex to the intersection point, and the corresponding mesh edge after attraction (the line segment connecting the two points) must be limited by a given threshold (see Figure 3b):

$$\delta(e, C) \leq \delta_{max} \tag{7}$$

Besides split-edge, there exists one scenario that we need to make use of a different topological operator; otherwise, the proposed procedure would not converge. This special scenario may happen at the endpoints. Suppose that the



(a) Next closest intersection (b) Approximation error

Fig. 3: Procedures guiding mesh modification to accomodate constraint.

two edges that connect the endpoint to the candidate vertices are part of previous constraints and that the opposite edge is not a close-to-longest edge. In such a case, the proposed procedure would refine the triangle indefinitely. We then detect such occurrence and refine the triangle not using the split-edge but the split-face operation, as illustrated in Figure 4.

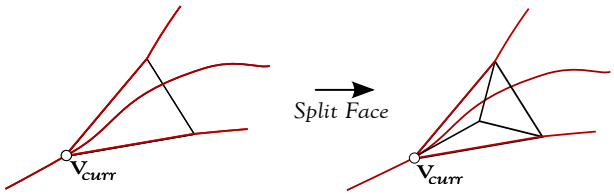


Fig. 4: Use of split-face topological operator.

3.1.4. Local improvement of quality

Although vertex attractions are guided by quality criteria, mesh quality is degraded after a sequence of attraction operations. In order to locally improve mesh quality after each attraction, we apply a smoothing operation on the vertices adjacent to the one just attracted, which is also smoothed. We have opted to use the well known Laplacian smoothing operator, which defines the new vertex smoothed position as being:

$$\vec{v}_i' = \vec{v}_i + \lambda L(\vec{v}_i) \tag{8}$$

where λ represents the velocity of displacement and $L(\vec{v}_i)$ the Laplacian of vertex $\vec{v}_i$ that, in its simplest form known as *umbrella operator*, is expressed as:

$$L(\vec{v}_i) = \frac{1}{\sum_{j \in Adj(\vec{v}_i)} w_{ij}} \sum_{j \in Adj(\vec{v}_i)} w_{ij}(\vec{v}_j - \vec{v}_i) \tag{9}$$

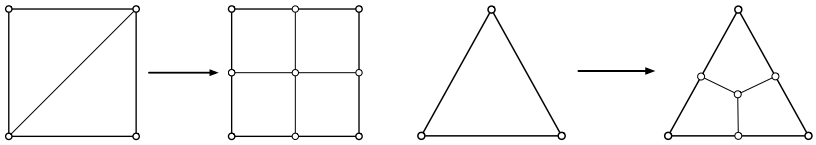
In this work, we have opted for using the weight of each adjacent vertex, w_{ij}, according to the *scale dependent umbrella operator* [24]: $w_{ij} = \|\vec{v}_j - \vec{v}_i\|^{-1}$.

The smoothing is performed by an iterative procedure. At each iteration, the new smoothed position of each vertex is computed and the vertex is displaced. If the vertex belongs to a constraint, the new position is reprojected along the constraint. Laplacian operations in general improve mesh quality significantly; however, it may also degrades triangles [25]. In order to avoid this, we only apply the operation if triangle quality after displacement remains above q_{min}. In the current version, we perform 10 smoothing iterations after each attraction.

If the generation of a triangular mesh is the final goal, the mesh undergoes a global smoothing operation. The same Laplacian operator is applied to the whole mesh to improve quality (in this version, we again use 10 iterations). However, if the goal is to have a quadrilateral mesh, the achieved triangular mesh goes through the conversion process.

3.2. Conversion to quadrilateral mesh

The conversion to a quadrilateral mesh follows the proposal by Velho [16]. First, adjacent triangles are combined two by two forming triangle-pair clusters. Isolated (non-clustered) triangles will naturally remain. Each cluster is then split into four quadrilaterals, by removing the shared edge and inserting mid-side vertices. Conversely, the isolated triangles are subdivided using Catmull-Clark pattern, also with the insertion of mid-side vertices, as illustrated in Figure 5. Quadrilaterals from clusters tend to present better shape quality than quadrilaterals from isolated triangles.



(a) Clustered triangles (b) Isolated triangle

Fig. 5: Conversion from triangular to quadrilateral mesh.

Velho [16] proposed the use of a greedy strategy to form the clusters. The triangle longest edges are processed in decreasing order of length. For each edge, if the adjacent triangle is yet not clustered, a new cluster is created. This procedure ensures that all clusters form convex quadrilateral blocks, validating the subdivision into four quadrilateral elements.

We propose the use of this procedure with some important modifications. First, the interior (shared) edge of a cluster cannot be part of a constraint, since it will be removed to create the quadrilateral subdivision. We then do not allow the creation of such clusters. Second, and more involving, we want to reduce the number of isolated triangles close to the constraints, aiming better mesh alignment. The triangular mesh undergoes a pre-processing phase prior to clustering; the following procedure is performed: First, all longest edges that lay on constraints are collected; then, the following steps are repeated a given number of iteration:

- For each edge in the collection:
  - Label the edge vertices.
  - Split the edge (to align cluster to constraints).
- Process the remaining edges to renew the collection, using labels just assigned to vertices:
  - Collect all longest edges whose one opposite vertex is labeled.
  - Swap and collect all longest edges whose one adjacent vertex is labeled, and opposite vertices are not.

This iterative procedure works as an advancing front and tends to collect edges aligned to line constraints. By splitting such edges, we force the creation of cluster following the lines. The number of iteration dictates how far from the constraints we want to spread the alignment. In the current version, we have opted to set this number to vary from 4 to 5: for each edge under inspection, the number of iteration is randomly chosen between 4 and 5, in order to perturb the transition zone.

4. Results

As mentioned, we are particularly interested in mesh generation for complex geological and multi-fractured models. Figure 1 illustrates one achieved quadrilateral mesh for a multi-fractured domain. In order to test the quality of the generated mesh, we set an experiment considering an actual geological model, defined by the constraints illustrated in Figure 6. We then applied our proposal to generate triangular and quadrilateral meshes for this domain using two different initial meshes: a Delaunay triangular mesh and a 4-8 triangular mesh. We then compared the achieved triangular mesh against one generated by the constrained Delaunay triangulation algorithm available in *Triangle* [26]. We also compared the resulting quadrilateral mesh against one generated using the SIGMA system [27]. The mesh in SIGMA was generated by a conventional user-guided domain decomposition method. The mesh in each decomposed

region was generated using mapping procedures (that in general generates good quality meshes), whenever possible, or Delaunay triangulation followed by conversion-to-quadrilateral procedure.

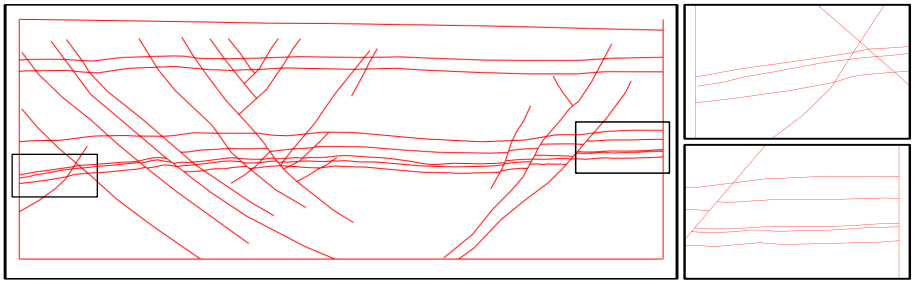
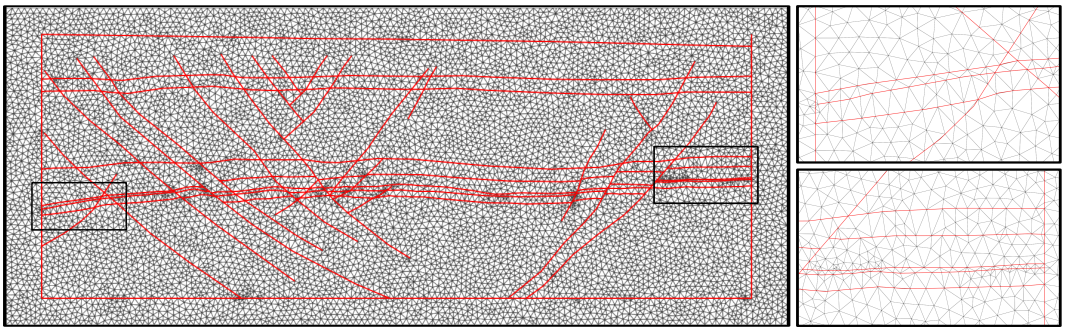
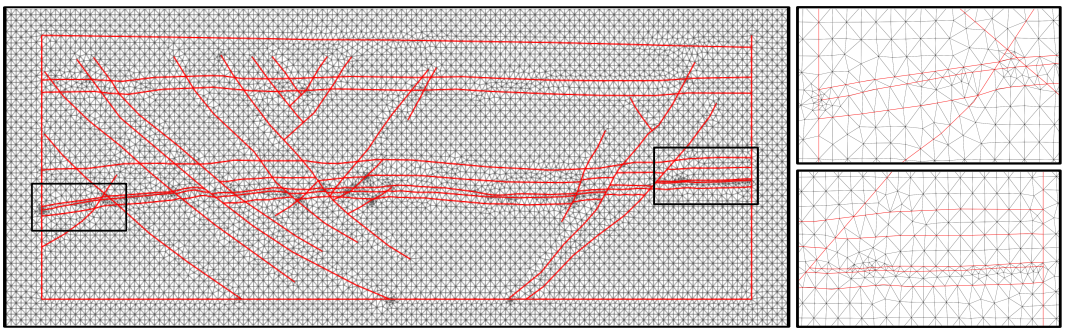


Fig. 6: Geological section domain

For comparison, all triangular meshes have approximately the same number of elements; this also applies for the quadrilateral meshes. Figure 7 shows the achieved triangular meshes and Figure 8 shows the mesh achieved by Triangle [26], generated using a no-small-angle parameter set to 30 degrees.



(a) Delaunay based (18,549 triangles)



(b) 4-8 based (18,864 triangles)

Fig. 7: Generated triangular meshes using different initial meshes.

We analyse the quality of the generated triangular meshes measuring the Lo parameter and vertex valence. Figure 9 depicts the achieved results. Note that both meshes generated using our proposal (from Delaunay or 4-8 pattern) present good overall quality; only a few number of elements present Lo parameter below 0.7, and less then 0.5% of the vertices has valence 9. Naturally, the 4-8-based mesh resulted in mesh with several elements with valence 4 and 8, while the Delaunay-based mesh presents better vertex valence statistics, presenting final numbers quite close to the

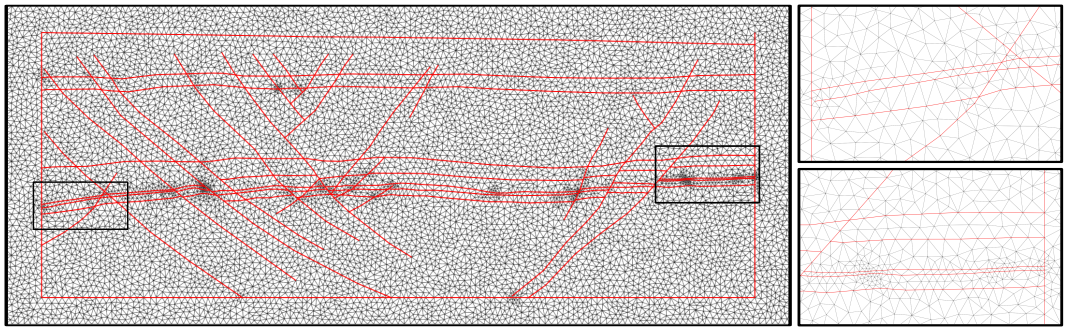
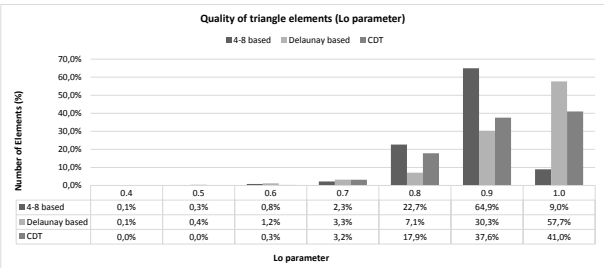


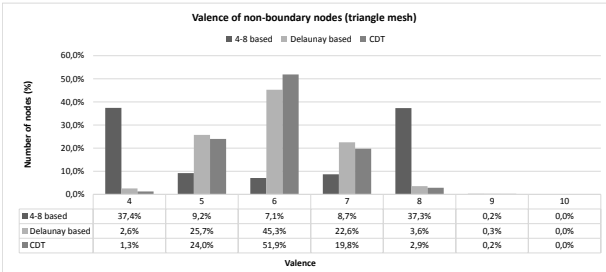
Fig. 8: Constrained Delaunay triangulation generated by Triangle [26] (18,693 triangles).

constrained Delaynay triangulation (CDT) generated by Triangle [26]. However, by preserving the global structure of the given initial meshes, our proposal favours conversion to quadrilateral meshes.



	0.4	0.5	0.6	0.7	0.8	0.9	1.0
4-8 based	0,1%	0,3%	0,8%	2,3%	22,7%	64,9%	9,0%
Delaunay based	0,1%	0,4%	1,2%	3,3%	7,1%	30,3%	57,7%
CDT	0,0%	0,0%	0,3%	3,2%	17,9%	37,6%	41,0%

(a) Lo parameter



	4	5	6	7	8	9	10
4-8 based	37,4%	9,2%	7,1%	8,7%	37,3%	0,2%	0,0%
Delaunay based	2,6%	25,7%	45,3%	22,6%	3,6%	0,3%	0,0%
CDT	1,3%	24,0%	51,9%	19,8%	2,9%	0,2%	0,0%

(b) Vertex valence

Fig. 9: Quality analysis of triangular meshes.

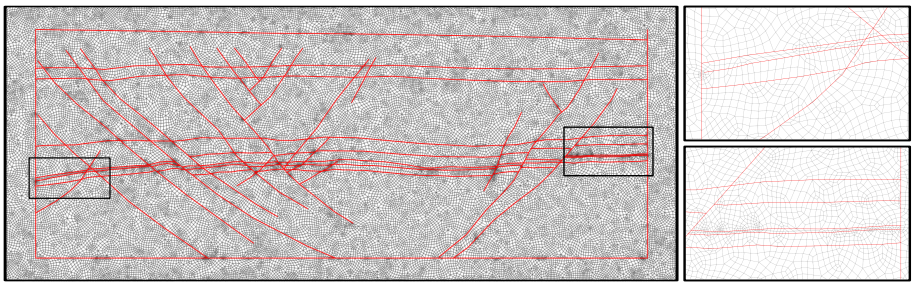
The generated triangular meshes were then converted to quadrilaterals. Figure 10 shows the achieved quadrilateral meshes, and Figure 11 illustrates the mesh generated by the SIGMA system [27]. Note that the 4-8-based mesh presents significantly better alignment than the others. This happens because the triangular mesh preserves the initial mesh structure and the clustering strategy searches to align clusters close to constraints.

As depicted in Figure 12, we analysed the quality of quadrilateral meshes measuring the minimal internal angle of the elements and the vertex valence. As can be noted, the achieved 4-8-based mesh presents better statistics, and even the Delaunay-based mesh is competitive.

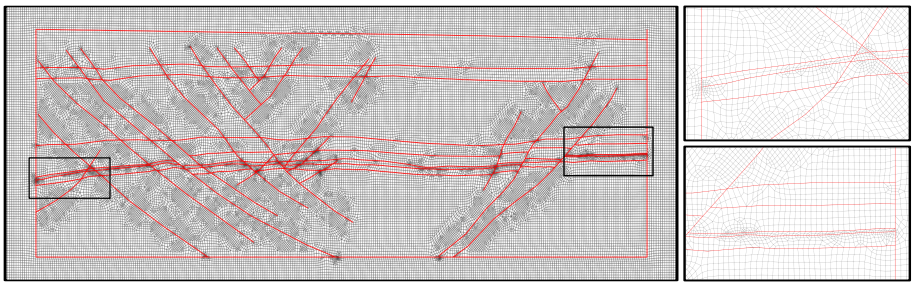
The proposed algorithm converges to a valid solution, with reasonable quality mesh, even for a very coarse initial mesh. Figure 13a illustrates a extreme coarse mesh, with only four triangles covering the whole domain. Figure 13b shows the achieved quadrilateral mesh, and Figures 13c and 13d its corresponding quality analysis. Finally, Figure 14 illustrates a result achieved for another complex geological domain.

5. Conclusion

In this paper, we introduced a new method for 2D mesh generation with deferred insertion of constraints. Starting with *any* initial valid triangular mesh, our proposal is capable of accommodating all constraints, despite their complexity, applying local topological and smoothing operations. This strategy preserves the global structure of the given initial mesh, which favours the conversion to quadrilaterals. We convert the triangular mesh using triangle-pair clustering in a way to reduce the number of non-aligned clusters close to constraints. As a result, we have achieved a



(a) Delaunay-based (43,882 quadrilaterals)



(b) 4-8-based (45,122 quadrilaterals)

Fig. 10: Generated quadrilateral meshes using our proposal with different initial meshes.

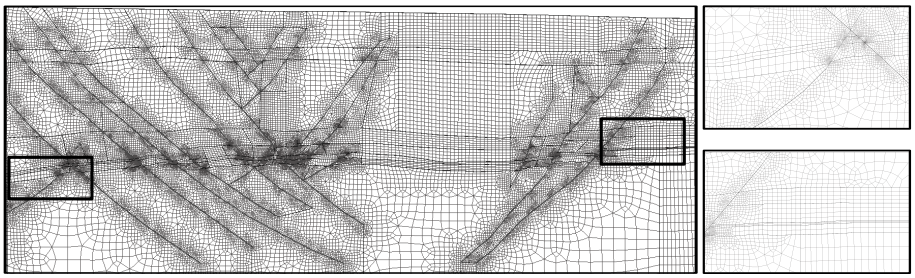


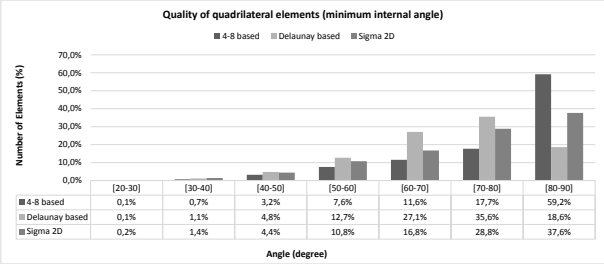
Fig. 11: Quadrilateral mesh generated by SIGMA (45,082 quadrilaterals).

very efficient, robust, good-quality, and automated mesh generator. Needless to say, our proposal is quite appropriate to modify an existing mesh due to the insertion of new constraints.

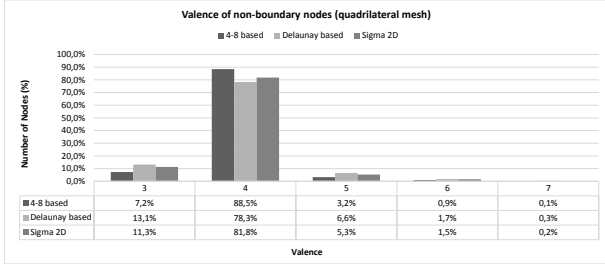
Our future plans include the investigation of other topological operators for reducing mesh distortion. Also, as demonstrated by previous works, a post-processing phase may be employed in order to improve mesh quality. Our main goal is the generation of 3D meshes, and we believe the deferred constraint insertion strategy presented in this paper is also promising for 3D domains.

Acknowledgment

The authors thank the support provided by the Tecgraf Institute at PUC-Rio, which is mainly funded by the Brazilian oil company, Petrobras. The second author also thank the Brazilian National Council for Scientific and Technological Development (CNPq) for the financial support to conduct this research. The constrained Delaunay triangulation was generated using *Triangle*, created at Carnegie Mellon University as part of the Quake project (http://www.cs.cmu.edu/ quake/triangle.html).

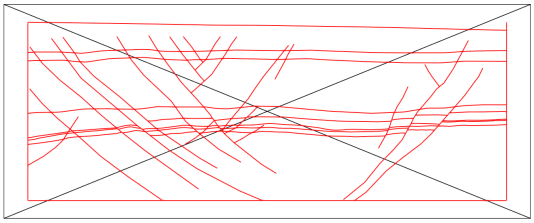


(a) Minimum angle

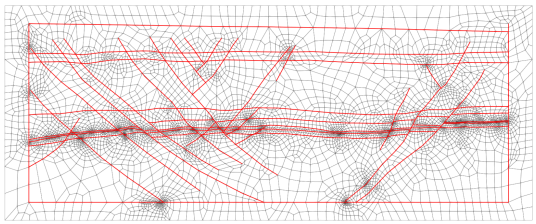


(b) Vertex valence

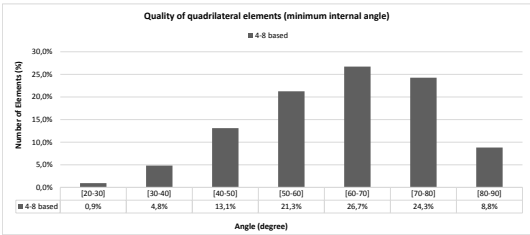
Fig. 12: Quality analysis of quadrilateral meshes.



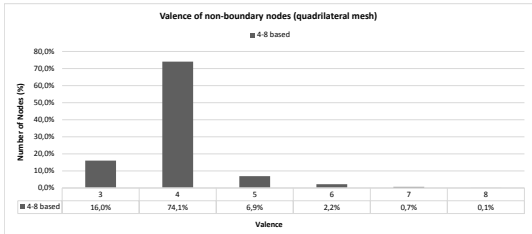
(a) Initial coarse mesh (4 triangles)



(b) Achieved result (8,017 quadrilaterals)



(c) Minimum angle



(d) Vertex valence

Fig. 13: Convergence with extreme coarse initial mesh.

References

[1] P. L. Baehmann, S. L. Wittchen, M. S. Shephard, K. R. Grice, M. A. Yerry, Robust, geometrically based, automatic two-dimensional mesh generation, International Journal for Numerical Methods in Engineering 24 (1987) 1043–1078.

[2] D. Nowottny, Quadrilateral mesh generation via geometrically optimized domain decomposition, in: Proceedings, 6th International Meshing Roundtable, 1997, pp. 309–320.

[3] S.-W. Chae, J.-H. Jeong, Unstructured surface meshing using operators, in: Proceedings, 6th International Meshing Roundtable, 1997, pp. 281–291.

[4] J. A. Talbert, A. R. Parkinson, Development of an automatic, two-dimensional finite element mesh generator using quadrilateral elements and bezier curve boundary definition, International Journal for Numerical Methods in Engineering 29 (1990) 1551–1567.

[5] W. Lin, Y. Tang, C. Zhao, X. Liu, G. Zhu, F. Jiang, An algorithm for automatic 2d finite element mesh generation with line constraints, Computer-Aided Design 43 (2011) 1803–1813.

[6] T. D. Blacker, M. B. Stephenson, Paving: A new approach to automated quadrilateral mesh generation, International Journal for Numerical Methods in Engineering 32 (1991) 811–847.

[7] D. R. White, P. Kinney, Redesign of the paving algorithm: Robustness enhancements through element by element meshing, in: Proceedings, 6th international meshing roundtable, 1997, pp. 323–335.

[8] R. J. Cass, S. E. Benzley, R. J. Meyers, T. D. Blacker, Generalized 3-d paving: An automated quadrilateral surface mesh generation algorithm, International Journal for Numerical Methods in Engineering 39 (1996) 1475–1489.

[9] C. Park, J.-S. Noh, I.-S. Jang, J. M. Kang, A new automated scheme of quadrilateral mesh generation for randomly distributed line constraints,

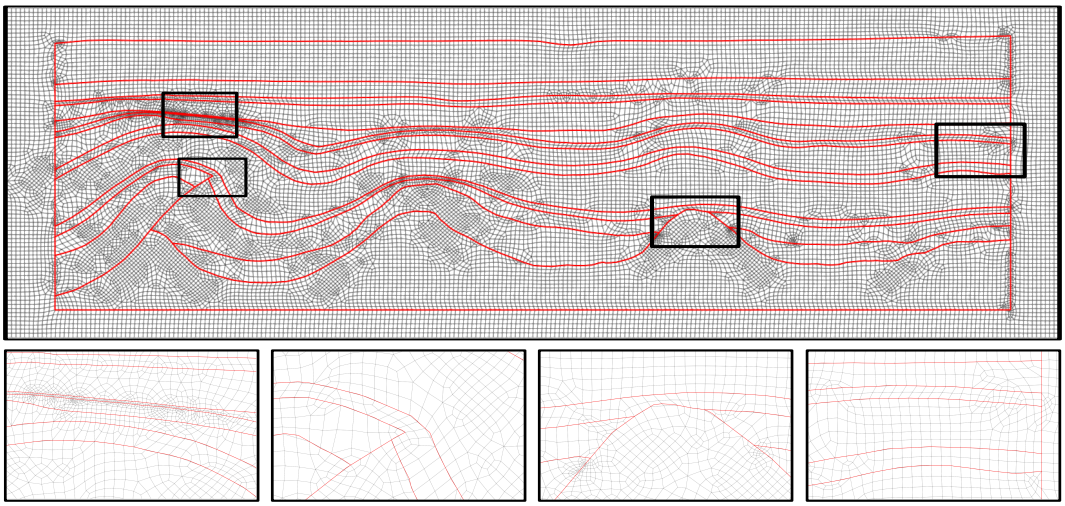


Fig. 14: Quadrilateral mesh generated for another complex geological domain (23,778 elements).

Computer-Aided Design 39 (2007) 258–267.

[10] E. Catmull, J. Clark, Recursively generated b-spline surfaces on arbitrary topological meshes, Computer-Aided design 10 (1978) 350–355.

[11] Y. Liu, H. Xing, Z. Guan, An indirect approach for automatic generation of quadrilateral meshes with arbitrary line constraints, International Journal for Numerical Methods in Engineering 87 (2011) 906–922.

[12] Y. Liu, H. Xing, A boundary focused quadrilateral mesh generation algorithm for multi-material structures, Journal of Computational Physics 232 (2013) 516–528.

[13] B. P. Johnston, J. M. Sullivan, A. Kwasnik, Automatic conversion of triangular finite element meshes to quadrilateral elements, International Journal for Numerical Methods in Engineering 31 (1991) 67–84.

[14] C. Lee, S. Lo, A new scheme for the generation of a graded quadrilateral mesh, Computers & structures 52 (1994) 847–857.

[15] H. Borouchaki, P. J. Frey, Adaptive triangular-quadrilateral mesh generation, International Journal for Numerical Methods in Engineering (1998) 915–934.

[16] L. Velho, Quadrilateral meshing using 4-8 clustering, in: Proceedings CILANCE 2000. Symposium on Mesh Generation and Self-Adaptivity, 2000, pp. 61–64.

[17] S. J. Owen, M. L. Staten, S. A. Canann, S. Saigal, Q-morph: an indirect approach to advancing front quad meshing, International Journal for Numerical Methods in Engineering 44 (1999) 1317–1340.

[18] K.-Y. Lee, I.-I. Kim, D.-Y. Cho, T.-w. Kim, An algorithm for automatic 2d quadrilateral mesh generation with line constraints, Computer-Aided Design 35 (2003) 1055–1068.

[19] F. Kälberer, M. Nieser, K. Polthier, Quadcover-surface parameterization using branched coverings, Computer Graphics Forum 26 (2007) 375–384.

[20] D. Bommes, H. Zimmer, L. Kobbelt, Mixed-integer quadrangulation, ACM Transactions On Graphics (TOG) 28 (2009) 77.

[21] D. Bommes, M. Campen, H.-C. Ebke, P. Alliez, L. Kobbelt, Integer-grid maps for reliable quad meshing, ACM Transactions on Graphics (TOG) 32 (2013) 98.

[22] W. Celes, G. H. Paulino, R. Espinha, A compact adjacency-based topological data structure for finite element mesh representation, International Journal for Numerical Methods in Engineering 64 (2005) 1529–1556.

[23] W. Celes, G. H. Paulino, R. Espinha, Efficient handling of implicit entities in reduced mesh representations, Journal of Computing and Information Science in Engineering, Special Issue on Mesh-Based Geometric Data Processing 5 (2005) 348–359.

[24] M. Desbrun, M. Meyer, P. Schröder, A. H. Barr, Implicit fairing of irregular meshes using diffusion and curvature flow, in: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999, pp. 317–324. URL: `http://dx.doi.org/10.1145/311535.311576`. doi:10.1145/311535.311576.

[25] H. Erten, A. Üngör, C. Zhao, Mesh smoothing algorithms for complex geometric domains, in: B. Clark (Ed.), Proceedings of the 18th International Meshing Roundtable, Springer Berlin Heidelberg, 2009, pp. 175–193. URL: `http://dx.doi.org/10.1007/978-3-642-04319-2-11`. doi:10.1007/978-3-642-04319-2-11.

[26] J. R. Shewchuk, Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator, in: M. C. Lin, D. Manocha (Eds.), Applied Computational Geometry: Towards Geometric Engineering, volume 1148 of *Lecture Notes in Computer Science*, Springer-Verlag, 1996, pp. 203–222. From the First ACM Workshop on Applied Computational Geometry.

[27] C. S. Amaral, A. M. Costa, M. T. Carvalho, W. W. M. Lira, Descrição do sistema SIGMA: Sistema Integrado em Geotecnia para Múltiplas Análises, Agreement Tecgraf/PUC-Rio – CENPES/Petrobras, Rio de Janeiro, 1996.