An Algorithm for Two-Dimensional Mesh Generation for Arbitrary Regions with Cracks

ANTONIO C. O. MIRANDA[1] JOAQUIM B. CAVALCANTE NETO[2] LUIZ F. MARTHA[1]

[1]Department of Civil Engineering and
Computer Graphics Technology Group (TeCGraf),
Pontifical Catholic University of Rio de Janeiro (PUC-Rio) 22453-900,
Rua Marquês de São Vicente, 225,
Rio de Janeiro, RJ, Brazil.
amiranda,lfm@tecgraf.puc-rio.br

[2]Computation Department, Federal University of Ceara
Campus do Pici, bloco 910, 60455-760,
Fortaleza-CE, Brazil.
joaquimb@lia.ufc.br

Abstract. This paper describes an algorithm for generating unstructured triangulations for arbitrarily shaped two-dimensional regions. The algorithm works for regions without cracks, as well as for regions with one or multiple cracks. The algorithm incorporates aspects of well-known meshing procedures and includes some original steps. It includes an advancing front technique, which uses an quadtree procedure to develop local guidelines for the size of generated elements. The advancing front technique is based on a standard procedure found in the literature, to improve mesh quality (as far as element shape is concerned), an *a posteriori* local mesh improvement procedure is used.

1 Introduction

This paper describes an algorithm for generating triangular meshes for arbitrarily shaped two-dimensional regions. The algorithm was devised in the context of finite-element modeling for crack propagation simulation[1].

The algorithm works for regions without cracks, as well as for regions with one or multiple cracks. The cracks may be embedded or surface breaking. The algorithm is an adaptation of an algorithm for generating unstructured meshes for arbitrarily shaped three-dimensional regions [1].

The algorithm was designed to meet four specific requirements. First, the algorithm should produce well-shaped elements, avoiding elements with poor aspect ratio, whenever possible. While the algorithm does not guarantee bounds on element aspect ratios, care is taken at each step to generate the best shaped elements possible.

The second requirement is that the algorithm generates a mesh that conforms to an existing discretization on the boundary of the region. This is important in the context of crack growth simulation because it allows remeshing to occur locally in a region near a growing crack. That is, a relatively small number of elements near the crack can be deleted creating a void in the mesh. The crack is extended, and then this algorithm can be used to generate new elements that fill the void, and conform to the elements that were not removed. The remeshed zone is small and localized, leading to fast mesh generation and, for nonlinear problems, minimizes the amount of state information that needs to be mapped between an old and new mesh.

The third requirement of the algorithm is that it has the ability to transition well between regions with elements of very different sizes. In a crack analysis, it is not uncommon for the elements near the crack front to be two orders of magnitude smaller than other elements in the problem. Some other algorithms work best when all generated elements have similar characteristic size [2]. This is clearly unacceptable for the crack case, and the current algorithm has been designed to have good size transition capabilities.

The fourth requirement arises because cracks are usually idealized as having no volume. That is, the surfaces representing the two-sided of a crack edge are distinct, but geometrically coincident. This means that nodes on opposite sides of crack faces may have identical coordinates. The algorithm must be able to discriminate between the nodes and select the one on the proper crack edge.

The body of this paper is divided into two main sections. The following section describes the steps of the algorithm in some detail. In Section 3 comparisons are made between the proposed algorithm and other algorithms based on quadtree in relation to the quality of the generated

[1]Cracks may be considered as discontinuities in the domain of a body. They are usually induced by flaws in the manufacturing or construction process of a structure or equipment in conjunction with stress concentration.

meshes and processing time.

2 Description of the algorithm

The proposed algorithm incorporates aspects of well-known meshing procedures and includes some original steps. It includes an advancing front technique, but uses an quadtree procedure to develop local guidelines for the size of generated elements. The advancing front technique is based on a standard procedure found in the literature [2–4]. To improve mesh quality (as far as element shape is concerned), an *a posteriori* local mesh improvement procedure is used.

The input to the algorithm is described by a list of nodes defined by their coordinates and a list of edges defined by their node connectivity. This type of input has some aspects to be considered: it can represent geometries of any shape, including holes and cracks, and it can be easily incorporated into any finite element system.

The algorithm is organized in the following phases:

- Quadtree generation
  - Initialization based on discretization boundary
  - Refinement to force a maximum cell size
  - Refinement to provide minimum size disparity for adjacent cells
- Advancing front procedure
  - Geometry-based element generation
  - Topology-based element generation
- Local mesh improvement
  - Laplacian smoothing
  - Quality evaluation and local back-tracking with element deletion

2.1 Quadtree generation

As mentioned above, the primary purpose for the quadtree is to generate guidelines for the size of the element generated during the advancing front procedure. The element size distribution through the region is inferred by the size distribution in the input boundary.

The quadtree generation involves three steps. In the first step, the quadtree is initialized based on the input data. In the other two steps, the quadtree is further refined. Figures 1 to 4 are used to illustrate the process of generating the quadtree.

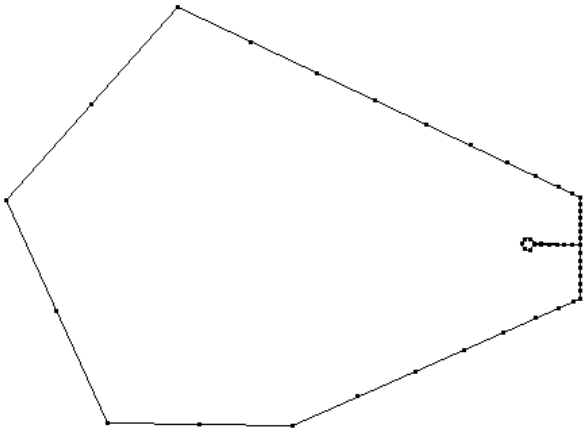

Figure 1: Hypothetical model and its boundary refinement.

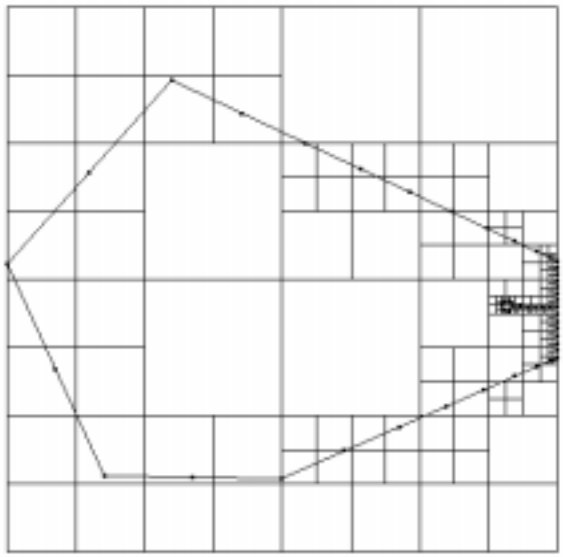

Figure 2: Initial quadtree of model.

2.1.1 Quadtree initialization based on discretization boundary

Initially, a bounding box is created based on the maximum range of any of the two cartesian coordinates of the nodes in the input data. This is the root cell of the quadtree. Figure 1 illustrates an hypothetical input data, representing a model and its boundary refinement. This model has an edge crack on its right hand side. At the crack tip, the boundary is contracted as if it had specially placed crack-tip elements. The boundary model presents an increasing degree of refinement from the left side to the right side.

In the first step of the quadtree generation, represented by the initialization of the quadtree, each segment of the input boundary data is used to determine the local depth

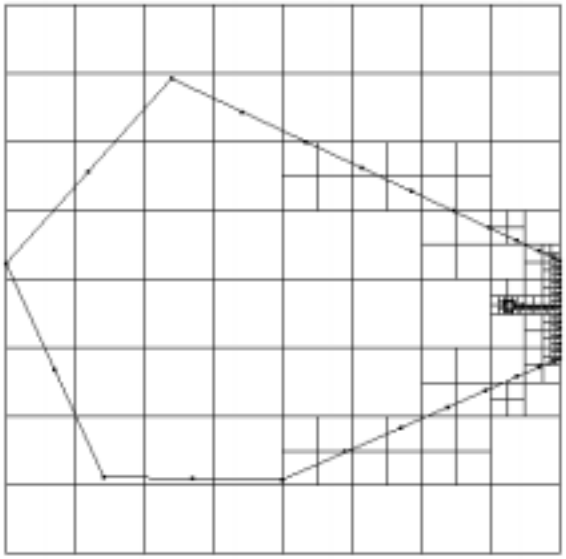

Figure 3: Quadtree of model after forcing largest cell size at boundary.

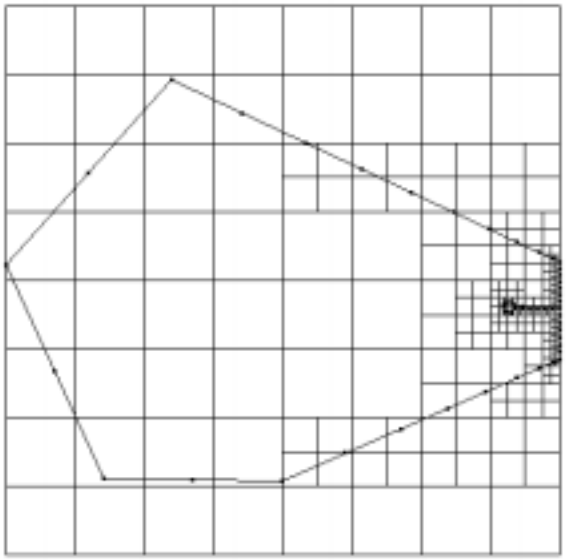

Figure 4: Quadtree of model after forcing maximum of one level of difference.

of subdivision. The quadtree cell containing the midpoint of each input segment is determined. If the length of this cell's edge is larger than the length of the boundary edge, then this cell is subdivided into four smaller cells. This process is repeated recursively and finishes when the length of the cell's edge is smaller than a factor of the length of the boundary segment. In this implementation a factor of 1.0 was used to avoid the excessive refinement in the quadtree generation [5,6]. Other algorithms use values different than 1.0 [1]. This process is repeated for all segments of the input data. The result is illustrated in Figure 2.

2.1.2 Quadtree refinement

The previous step can leave large quadtree cells in the interior of the region. In a second step, the quadtree is refined to guarantee that no cell in the interior is larger than the largest cell at the boundary. This will avoid excessive big elements in the domain interior. Figure 3 shows the quadtree generated after this operation.

2.1.3 Refinement to provide minimum size disparity for adjacent cells

The quadtree is further processed in a third step, to force only one level of refinement between neighboring cells. This enforces a natural transition between regions of different degrees of refinement. This operation is performed traversing the quadtree and examining the level of refinement between adjacent cells. If the difference is more than one level, then the appropriate cells are refined until the criterion is satisfied. Figure 4 shows the quadtree generated after this procedure.

2.2 Advancing front technique

The advancing front technique starts with a boundary that bounds a region to be filled with a triangulation. Triangular elements are "extracted" or "pared" from the region one at a time. As each element is extracted, the boundary is updated and the process is repeated. The procedure terminates when the entire region is meshed. Therefore, the boundary of the region to be meshed is formed by edges of triangles created in the contration process. These edges are referred to as *boundary edges*.

In this algorithm, the advancing front process is divided into two phases to ensure generation of valid triangulations. In the first phase, a geometry-based element generation is pursued to generate element of optimal shapes. After this ideal phase is exhausted, and no more optimal elements can be generated, a topology-based element generation takes place, creating valid, but not necessarily well shaped, elements in the remaining region.

2.2.1 Geometry-based element generation

Ideally, the entire mesh would be generated in the geometry-based phase. This depends on the geometry and topology of the given boundary model and, as observed, is strongly related to the segment size disparity of the given boundary refinement.

a) Boundary contraction lists

The process starts with the creation of the initial advanc-

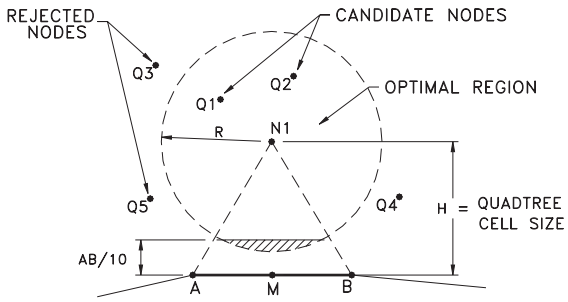

Figure 5: The determination of a triangle.

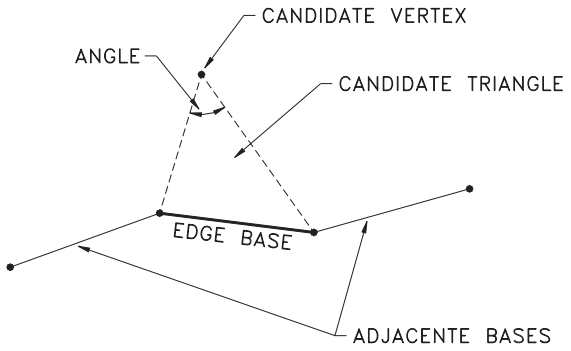

Figure 6: The definition of a angle for a vertex i.

ing front, which is formed by the given boundary segments. The current boundary edges are stored in two separate doubly-linked lists. The first is a list of active edges, which includes all boundary edges that have not been used in an attempt to generate valid triangles. The other is a list of rejected edges, that is, with the edges that failed in the generation of elements for the current phase. Initially, all segments of the given boundary refinement are stored in the first list, which is the list used in the geometry-based generation phase.

The initial list of active edges on boundary is sorted by the length of the edges. This has been recommended by other authors [7] to prevent large elements from penetrating into regions with small length edges.

It was also found convenient for some steps in the algorithm to have an additional data structure that holds a list of adjacent boundary edges for each node on the current advancing front. This data structure is initialized for all the nodes of the given boundary. The data structure is updated as the boundary contraction procedure progresses.

b) Generation of optimal elements

In the geometry-based element generation phase, the current boundary advances trying to form triangle based mainly on geometrical considerations. At each step, an edge, referred to as *base edge*, is chosen from the list of active edges.

The procedure for generating a triangle in this phase is explained by means of Figure 5. This procedure is divided into the following steps:

- The optimal location $N1$ for the vertex of the triangle to be formed is determined with the help of the quadtree. The quadtree cell containing the midpoint M of the base edge is determined. The optimal point $N1$ lies on a line perpendicular to the base edge passing through this midpoint. The distance from the optimal point to the base edge midpoint is equal to the quadtree cell size.

- The optimal point defines an optimal region where the vertex of triangle to be generated is located. This region is a sector of circle whose center is the optimal point and whose radius is proportional to the quadtree cell size. In the current implementation a factor of 0.85 was adopted. This circle defines an upper bound for the distance between the target vertex of the triangle and the centroid of the base edge. A lower bound is defined to ensure that the generated triangle will have area greater than the smallest acceptable area. In the current implementation, this lower bound is defined by a triangle with height equal to 1/10 of base edge. The optimal region is used for two reasons. First, to ensure shape quality of the elements to be generated and, second, to ensure that new internal nodes will be created only when it is strictly necessary and always in good positions.

- If no existing node is inside the optimal region, a new node is inserted at the optimal location $N1$ and an element is generated using this node. If only one node exists in the region, this node is used to generate the element. If more than one node is found in the region, they are ranked according to the angle that they will create with the base edge (Figure 6). A heap list is used to efficiently rank the nodes. The node that will create the largest angle is used to generate the element.

- Additional geometric checks are performed to insure that the edges of the new element do not intersect any existing edge of the advancing front. If this is the case, the element is rejected.

- For crack problems there may be two or more nodes with the same coordinates. Figure 7 illustrates this. The algorithm selects the proper node using a simple test, which is based on the lists of adjacent boundary edges of the nodes on the advancing front. When two candidate nodes at the crack surface are selected to form an element, the node which lies on the same side of the base edge with respect to the crack edge is cho-

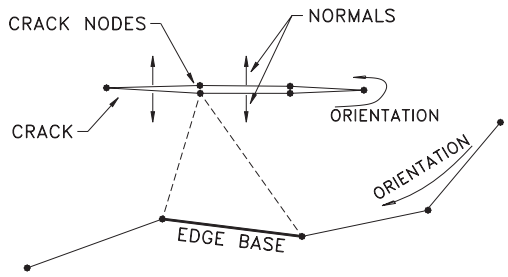

Figure 7: Selection of a candidate node at a crack edge.

sen. The normals to the crack curves adjacent to the select nodes are used to perform this test, as illustrated in Figure 7. This check assumes that all crack curves are smooth (with no abrupt change of direction and with no bifurcation, which it is not really a limitation in practical applications).

- Once a valid triangle is generated for the current base edge, the list of active edges is updated. This is done through the following steps. First, the base edge is removed from the list. Then, for the other edges of the element: each edge is deleted if it coincides with an edge already in the list, or the edge is inserted in the list as a new one.

- Due to geometric bounds imposed by the current advancing front, there are situations in which the algorithm fails in forming a valid triangle for the current boundary base edge. In these cases, the current base edge is removed from the list of active edges and is stored in the separate list of rejected edges. It might happen that an edge is subsequently removed from this latter list if it is used to form part of a valid triangle for an adjacent base edge.

- When there is no more edge in the list of active edges, the algorithm tries to generate elements using the edges that were rejected previously. It may be the case that base edges that failed previously may now work because the front has changed with the addition of elements. The geometry-based element generation phase ends when either there are no edges left in the boundary contraction lists (in which case an optimal mesh was generated) or when a rejected edge fails a second time.

2.2.2 Topology-based element generation

The objective of this phase of the algorithm is to force the generation of valid triangle, even if the new elements do not satisfy the bounds used in the previous phase for element shapes.

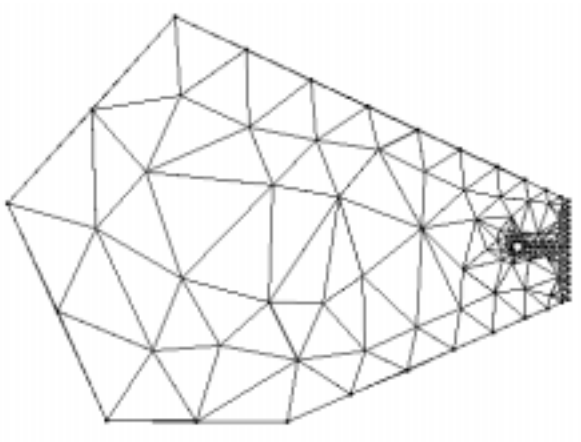

Figure 8: First mesh generated for the example.

The topology-based element generation phase starts when a boundary edge fails twice in trying to generate an optimal element. The list of rejected edges of the previous phase is transformed into an list of active edges and, similarly to the geometry-based phase, a list of rejected edges is created for edges that eventually fail in generating valid triangle.

In the topology-based element generation phase, any node close to the current base edge is selected and stored in the local heap list of candidate nodes. The node that has the best angle metric with respect to the base edge is chosen for the generation of the new triangle. If the edges of this triangle do not intercept any other edge of the current advancing front, the element is created and the boundary is accordingly contracted. The topology-based phase ends when the lists of active and rejected edges are empty. This phase always generates a valid mesh (although not optimal) because it is always possible to triangulate a region that is defined by its boundary edges [8]. Figure 8 shows the mesh of the example model after this phase.

2.3 Local mesh improvement

In the last phase of the advancing front technique described previously, triangular finite elements of non-optimal shapes might be generated. This section describes two *a posteriori* local mesh procedures that take place to improve mesh quality. The first is a conventional nodal relocation smoothing technique, which is based on node coordinates averaging. The second is a back-tracking procedure that deletes edges of "bad" shape elements to create a region where elements with better shape can be generated.

The local mesh improvement procedures imply that element shape quality measures are necessary. In this work a quality shape metric was adopted based on a comprehensive study on finite element shape quality [9–19]. In this

case, the adopted quality measure is a normalized ratio between the root mean square of the lenghts of the edges of a triangle ($S_{rms} = \sqrt{\frac{1}{3}\sum_{i=1}^{3} S_i^2}$, where S_i is the length of an edge) and the area A of the triangle:

$$\gamma = \frac{S_{rms}^2}{A}. \qquad (1)$$

This metric presents a good quality expressivity and is computationally efficient. The metric used in this work is the ratio $\gamma/\gamma*$, where $\gamma*$ is the metric for equilateral triangle. The range of valid values varies from one to infinity $[1, \infty)$.

2.3.1 Laplacian smoothing

A smoothing technique is used to improve mesh quality by relocating nodes within a patch. A general formulation for this technique is given through equation (2), which is a generic form of a weighted Laplacian function (Foley *et al.*, 1990):

$$\mathbf{X}_O^{n+1} = \mathbf{X}_O^n + \phi \frac{\sum_{i=1}^{m} \omega_{iO}(\mathbf{X}_i^n - \mathbf{X}_O^n)}{\sum_{i=1}^{m} \omega_{iO}} \qquad (2)$$

In equation (2), *m* is the number of nodes connected to node *O*, $\mathbf{X}_O^{n+1}$ is the position of node *O* at smoothing iteraction $n+1$, ω_{iO} is the weighted function between nodes *i* and *O*, and ϕ is a relaxation parameter which is normally set in the interval $(0, 1]$. In this work, a value of ϕ = 0.5 and $\omega_{iO} = 1$ were adopted, resulting in a simple average of nodes. The smoothing procedures is repeated 5 times.

2.3.2 Quality evaluation and local back-tracking with element deletion

The Laplacian smoothing is not sufficient to guarantee mesh quality. In this work, a back-tracking procedure is adopted to improve the quality of a generated mesh. This technique can also be used for any given mesh that is not related to the current meshing algorithm. This can be helpful in improving the quality of meshes obtained by other approaches.

The back-tracking procedure consists of deleting an element that is classified as a poorly shaped element and a group of elements in its vicinity. The classification of "bad" triangle is based on a specified measure, which in this work is the $\gamma/\gamma*$ metric described previously. For each element of the generated mesh, the quality measure $\gamma/\gamma*$ is evaluated. If the value of this metric is above a pre-defined limit value, the element is classified as a poorly shaped element. The limit value is defined empirically based on experiments and observations. In this work, the adopted value is 1.5. The back-tracking procedure, however, can be applied for any quality measure and for any limit value used to define a "bad" element.

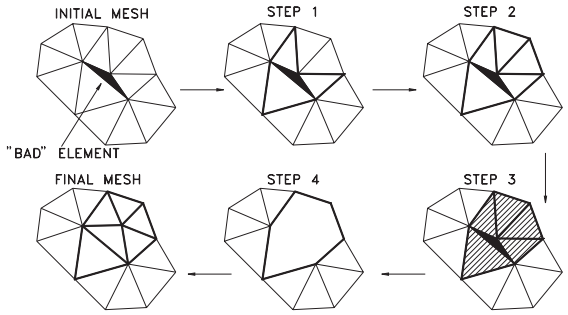

Figure 9: Back-tracking procedure to remesh around a "bad" element.

The objective of the back-tracking procedure is to delete element edges surrounding a "bad" element to create a local polygon that can be remeshed with better shape elements. A local polygon to be meshed is created by an algorithm that is defined through the following steps. This is illustrated by means of Figure 9:

- Step 1: creation of a initial list of adjacent nodes. For each element that is adjacent to the poorly shaped element (the gray element of Figure 9), store its nodes in the list. An adjacent element is an element that shares two nodes (an edge) with the "bad" element.

- Step 2: creation of an extended list of adjacent nodes. The additional nodes are the ones that share more than one element with nodes of the initial list of Step 1. That step was incorporated because it was seen that it that results in polygons with better shapes for reconstruction.

- Step 3: creation of a list of elements to be deleted. These elements are defined by having three nodes in the extended list of adjacent nodes.

- Step 4: creation of the local polygon to be remeshed. For each element to be deleted, each of its edges is inserted in the polygon boundary if it is not there yet, otherwise the edge is removed.

After the creation of the local polygon, elements are generated by applying of the advancing front algorithm procedures described previously.

The back-tracking procedure followed by element regeneration, similarly to the smoothing technique, is applied a number of times (5 times). In this work, the smoothing technique was used in conjunction with the back-tracking procedure. It was observed that better results are obtained if back-tracking/regeneration is applied after each step of smoothing. Figure 10 shows the final mesh generated for the example.

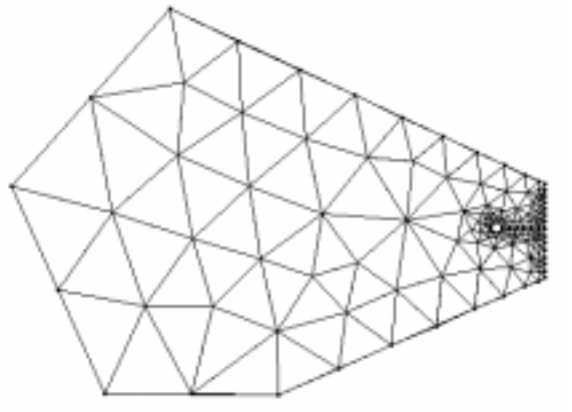

Figure 10: Final mesh generatated for the example.

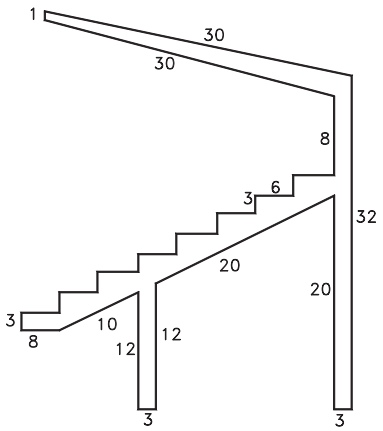

Figure 11: Example of a stand model.

3 Comparison with other quadtree algorithms

In this section, comparisons are made between the proposed algorithm and other algorithms based on quadtree in relation to the quality of the generated meshes and to processing time. The algorithms used for that comparison are: Vianna [20], based on Constrained Delaunay Triangulation [2, 15, 21], and Cavalcante Neto [6], based on triangulation by a quadtree technique with templates [22, 23].

The measured time does not include the time required to build the input data. The computer used in this operation was a 200 MHz Intel PC with 32 Megabytes of RAM, and running Windows 95.

The quality of generated meshes was measured with the normalized metric $\gamma/\gamma*$. This metric has a valid interval between 1.0 and infinity, and the value for the equilateral triangle is 1.0. It is desirable to have elements with values close to 1.0.

The following models were used in the comparisons: a square with 60 segments on each side; a circle with 120 segments; a stand illustrated in Figure 11 (the figure indicates the numbers of segments of each edge); and a frame illustrated in Figure 12. The number of generated elements for each model is presented in Table 1. Figures 13 to 16 show the generated meshes for the models.

Table 2 presents the processing time of each model. Table 3 presents the number of generated elements per second. One way verify that the proposed algorithm is faster than the other algorithms.

The quality of generated meshes are presented in the form of histograms in Figures 17 to 20. In the histograms, the horizontal axis represents the normalized metric in intervals of 0.1. The vertical axis represents the percentage of elements corresponding to the intervals of the metric. In all models used in the comparisons of mesh quality, the proposed algorithm had the best meshes.

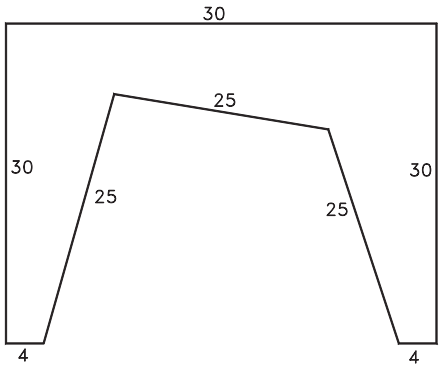

Figure 12: Example of a frame model.

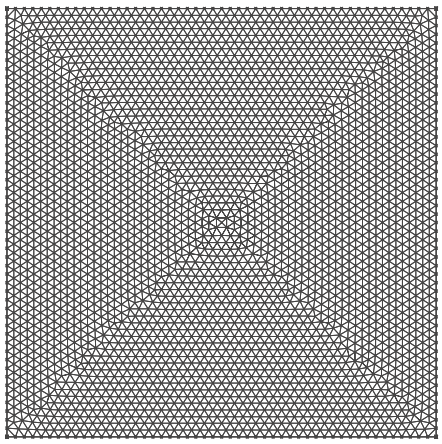

Figure 13: Mesh for the square model.

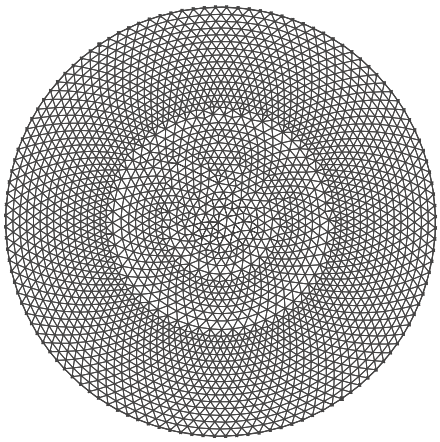

Figure 14: Mesh for the circle model.

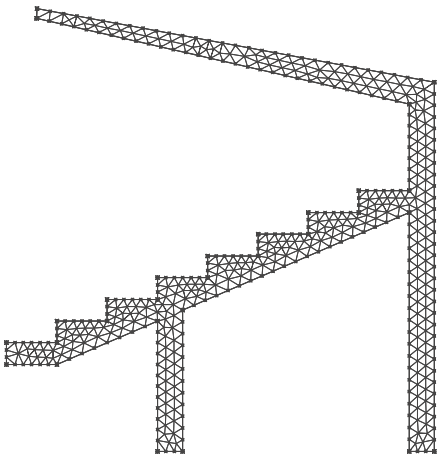

Figure 15: Mesh for the stand model.

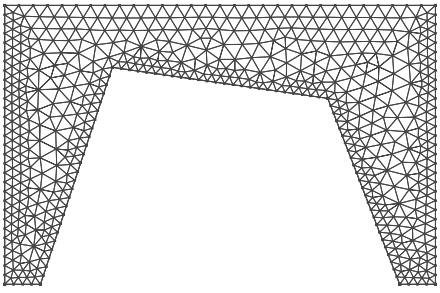

Figure 16: Mesh for the frame model.

EXAMPLE	ALGORITHM		
	VIANNA	CAVALCANTE NETO	PROPOSED
SQUARE	8070	1560	6330
CIRCLE	1670	600	4994
STAND	499	788	751
FRAME	979	1207	1061

Table 1: Comparison of number of generated elements.

EXAMPLE	ALGORITHM		
	VIANNA	CAVALCANTE NETO	PROPOSED
SQUARE	67.61	4.83	13.62
CIRCLE	4.34	2.64	8.29
STAND	2.75	13.73	1.70
FRAME	2.58	9.40	1.59

Table 2: Comparison of processing times (s).

EXAMPLE	ALGORITHM		
	VIANNA	CAVALCANTE NETO	PROPOSED
SQUARE	119.3	323.0	464.7
CIRCLE	384.8	227.3	602.4
STAND	181.5	57.4	441.8
FRAME	379.5	128.4	667.3

Table 3: Comparison of number of generated elements per second.

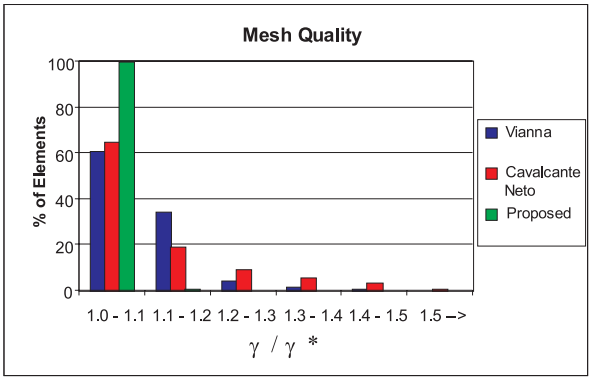

Figure 17: Histogram of element quality for the square model.

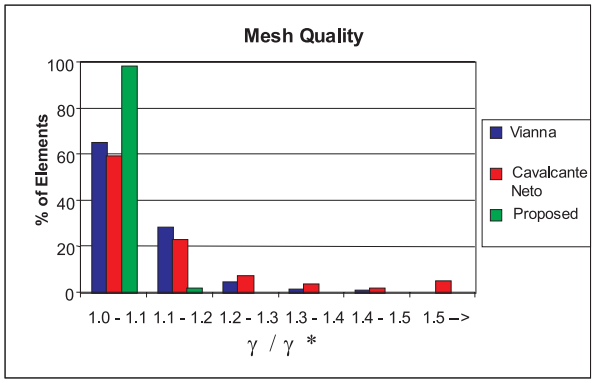

Figure 18: Histogram of element quality for the circle model.

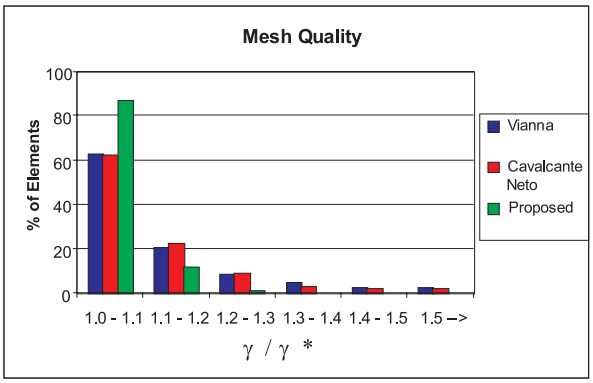

Figure 19: Histogram of element quality for the stand model.

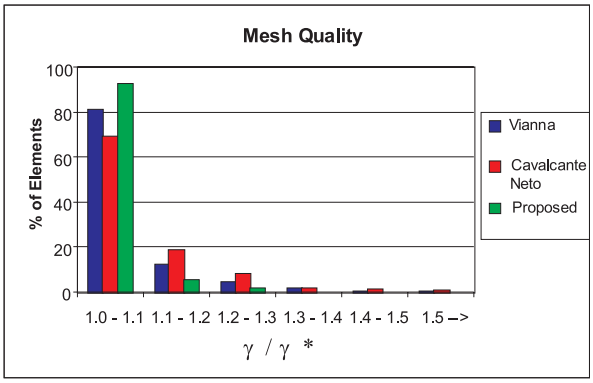

Figure 20: Histogram of element quality for the frame model.

4 Conclusions

An algorithm for generating unstructured triangulation for arbitrary shaped two dimensional regions was described. The algorithm incorporates aspects of well known meshing procedures and includes some original steps. The algorithm works for regions without cracks as well as for regions with one or multiple cracks. The cracks may be embedded or surface breaking. The algorithm was designed to meet four specific requirements.

- It should avoid producing elements with poor aspect ratios.
- It can generate meshes that conform to an existing discretization on the boundary of a domain.
- It generates meshes that exibit good transitions between regions of different element sizes.
- It works properly for cases where distinct boundary nodes are geometrically coincident (*e.g.* nodes on opposite edges of a crack).

The input to the algorithm is a list of nodes defined by their coordinates and a list of segments defined by their node connectivity, which describes the domain to be meshed. The steps in the algorithm are as follows:

- An quadtree is generated to control the distribution of node points generated in the interior.
- A two-pass advancing front procedure is used to generate elements. On the first pass, elements are generated based on geometrical criteria, which produce well shaped elements. On the second pass, elements are generated based only on the criterion that they have valid topology.

- The quality of the elements shapes is improved by the use of standard Laplacian smoothing and by locally deleting poorly shaped elements and those adjacent to them, and then restarting the boundary contration.

Four models were presented to compare the quality of generated meshes and processing time of the proposed algorithm with other quadtree-based algorithms. It was shown that the proposed algorithm generate meshes with good quality and efficient processing time.

References

[1] Cavalcante Neto, J.B. - "Geração de Malha e Estimativa de Erro para Modelos Tridimensionais de Elementos Finitos com Trincas," PhD Thesis, Pontifical Catholic of Rio de Janeiro (PUC-Rio), 1998.

[2] Lo, S. H. - "A New Mesh Generation Sheme for Arbitrary Planar Domains," *Int.J.Num.Meth.Engng.*, vol. 21, pp. 1403-1426, 1985.

[3] de Floriani, L. e Puppo, E. - "An On-line Algorithm for Constrained Delaunay Triangulation," *Graphical Models and Image Procesing*, vol. 54, pp. 290-300, 1992.

[4] Watson, D. F. - "Computing the n-Dimensional Delaunay Tesselation with Aplication to Voronoi Polytopes", *The Computer Jornal*, vol. 24, n. 2, pp 167-172 , 1981.

[5] Potyondy, D. O. - "A Software Framework for Simulating Curvilinear Crack Growth in Pressurized Thin Shells," PhD Thesis, School of Civil Engineering, Cornell University, 1993.

[6] Cavalcante Neto, J.B. - "Simulação Auto-Adaptativa Baseada em Enumeração Espacial Recursiva de Modelos Bidimensionais de Elementos Finitos," MSc Thesis, Pontifical Catholic of Rio de Janeiro (PUC-Rio), 1994.

[7] Peraire, J.; Peiro, J.; Formaggia, L.; Morgan, K. and Zienkiewicz, O.C. - "Finite Euler Computation in Three-Dimensions," *Int. J. Num. Meth. Engng.*, vol 26, pp. 2135-2159, 1988.

[8] O'Rourke, J. - Art Gallery Theorems and Algorithms, Oxford University Press, New York - NT, 1990.

[9] Bramble, J.H. and Zlámal, M. - "Triangular Elements in the Finite Element Method," *Math. Comp.*, vol. 24, pp. 809-810, 1970.

[10] Babuška, I. and Aziz, A.K. - "On the Angle Condition in the Finite Element Method," *SIAM J. Num. Anal.*, vol. 13, pp. 214-226, 1976.

[11] Cavendish, J.C.; Field, D.A. and Frey, W.H. - "An approach to Automatic Three-dimensional Finite Element Mesh Generation," *Int. J. Num. Meth. Engng.*, vol 21, pp. 329-347, 1995.

[12] Baker, T.J. - "Element Quality in Tetrahedral Meshes," *Proc. 7th Int. Conf. on Finite Element Meth. in Flow Problems*, Huntsville, AL, 1989.

[13] Cougny, H.L.; Shephard, M.S. and Georges, M.K. - "Explicit Node Smoothing Within Octree," Report 10-1990, SOREC, Rensselaer Polytechnic Institute, Troy - New York, 1990.

[14] Dannelongue, H.H. and Tanguy, P.A. - "Three-dimensional Adaptive Finite Element Computations and Applications to non-Newtonian Flows," *Int. J. Num. Meth. Engng.*, **13**, pp. 145-165, 1991.

[15] Joe, B. - "Delanay Triangular Meshes in Convex Polygons," *Int.J.Num.Meth.Engng.*, **31**, pp. 987-997, 1991.

[16] Parthasarathy, V.N.; Graichen, C.M. and Hathaway, A.F. - "A Comparison of Tetrahedron Quality Measures," Finite Elements Anal. Des., vol. 15, pp. 255-261, 1993.

[17] Weatherill, N.P. and Hassan, O. - "Efficient Three-dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints," *Int. J. Num. Meth. Engng.*, vol 37, pp. 2005-2039, 1994.

[18] Liu, A. and Joe, B. - "Relationship Between Tetrahedron Shape Measures," **BIT**, vol. 34, pp. 268-287, 1994.

[19] Lewis, R.W.; Zheng, Y. and Gethin, D.T. - "Three-dimensional Unstructured Mesh Generation: Part 3. Volume Meshes," *Comput. Meth. Appl. Mech.*, vol. 134, pp. 285-310, 1996.

[20] Viana, A. C. - "Modelagem Geométria Estendida para Modelos Bidimensionais," MSc Thesis, Pontifical Catholic of Rio de Janeiro (PUC-Rio), 1992.

[21] Chew, L. P. - "Constrained Delaunay Triangulation," *Algorithmica*, vol. 4 , pp. 97-108, 1989.

[22] Yerry, M.A. and Shephard, M.S. - "Automatic Three-Dimensional Mesh Generation by Modified-Octree Technique," *Int. J. Num. Meth. Engng.*, vol. 20, pp. 1965-1990, 1984.

[23] Baehmann, P.L. Wittchen, S.L., Shephard, M.S., *et al.* - "Robust Geometrically Based, Automatic Two-Dimensional Mesh Generation," *Int. J. Num. Meth. Engng.*, vol 24, pp. 1043-1087, 1987.