Proceedings of the XXVI Iberian Latin-American Congress on Computational Methods in Engineering CILAMCE 2005
Brazilian Assoc. for Comp. Mechanics & Latin American Assoc. of Comp. Methods in Engineering
Guarapari, Espírito Santo, Brazil, 19^{th}–21^{st} October 2005

Paper CIL14-0358

GENERATION OF UNSTRUCTURED TETRAHEDRAL MESHES USING AN ADVANCING FRONT METHOD

Vicente Helano F. Batista
Fernando Luiz B. Ribeiro
José Antonio F. Santiago
helano@coc.ufrj.br
fernando@coc.ufrj.br
santiago@coc.ufrj.br
Department of Civil Engineering, Federal University of Rio de Janeiro,
P.O. Box 68506, Rio de Janeiro, RJ, 21945-970, Brazil.

Abstract. *This paper presents an algorithm to generate uniform unstructured tetrahedral meshes based on advancing front method. The goal is to develop an efficient algorithm regardless the use of any external technique in order to improve convergence, such as backtracking or front controlling. Some examples show the ability of the algorithm to discretize arbitrary volumes, reaching reasonable distributions in terms of edge lengths, element volumes and other quality metrics.*

Keywords: *Mesh Generation, Unstructured Meshes, Advancing Front*

1. INTRODUCTION

Nowadays, various methods for generating unstructured tetrahedral meshes are available, but the most used are based on Delaunay triangulation (Delaunay, 1924), advancing front method (Lo, 1985), and recursive spatial decomposition (Meagher, 1982; Samet, 1989), in descending order of popularity. Advancing front based methods have the advantage that they produce boundary conforming meshes, and only one element is created per each node insertion, allowing total control when positioning the created nodes. However, the random way in which the front changes causes the generation of distorted polyhedra that usually break the discretization. In addition, it is necessary to use special data structures to make possible a robust code.

In this paper, an implementation of the advancing front method is proposed. This version is based on the algorithms conceived by Löhner and Parikh (1988) and Peraire and Peiró (1992). More algorithms such as Jin and Tanner (1993), Frykestig (1994), Dari (1994), Möller and Hansbo (1995), Rassineux (1998), Piteri (1998), Frey et al. (1998), and Cavalcante Neto et al. (2001) had also influenced it. Like in Golgolab (1989), Cavalcante Neto et al. (2001) and George and Seveno (1994), the domain surface mesh is assumed as an input data. This approach is interesting because it detaches the mesh generation process from the geometric modelling. Procedures to increase convergence, such as backtracking (Möller and Hansbo, 1995; Rassineux, 1998; Cavalcante Neto et al., 2001) or generation by layers (Piteri, 1998; Marcum, 2001; Ito et al., 2004), are not used. Only uniform meshes are generated, with element or edge sizes approximately constant throughout the domain.

2. PROPOSED METHOD

In the algorithm here proposed, the basic structure of a classic advancing front method is maintained. However, it was decomposed in two stages: generation of well shaped elements (Stage 1) and generation of elements to enforce convergence (Stage 2). During Stage 1, the maximum permitted number of elements with high quality are created, in order to avoid the propagation of distorted tetrahedra. Remaining void spaces after Stage 1 are discretized at Stage 2 with tetrahedra of inferior quality. Therefore, the larger is the number of the generated elements at Stage 2, the poorest is the quality of the mesh. In fact, it was developed one advancing front based algorithm for each particular stage.

Let $\mathcal{M}_\Omega$ be the tetrahedral mesh of a domain $\boldsymbol{\Omega}$, $\boldsymbol{\Omega} \subset \mathbb{R}^3$, with boundary denoted by $\boldsymbol{\partial\Omega}$, $\boldsymbol{\partial\Omega} \subset \mathbb{R}^3$, which has its discretization represented by $\mathcal{M}_{\partial\Omega}$. Then,

Definition 1. A *front* $\mathcal{F}_\Omega$ of a domain $\boldsymbol{\Omega}$, $\boldsymbol{\Omega} \subset \mathbb{R}^3$, is a surface mesh whose faces F_i, $F_i \in \mathcal{F}_\Omega$, $i = 1, 2, \ldots, n_F$, are used to create the tetrahedra that will compose $\mathcal{M}_\Omega$.

The main scheme of the proposed method is depicted in Alg. 1. The input parameters are an empty domain mesh $\mathcal{M}_\Omega$ and its corresponding boundary discretization $\mathcal{M}_{\partial\Omega}$. Initially, the front $\mathcal{F}_\Omega$ is equal to $\mathcal{M}_{\partial\Omega}$. After a number of elements is generated, $\mathcal{F}_\Omega$ is updated and no longer represents $\mathcal{M}_{\partial\Omega}$. Here, the subscript $\boldsymbol{\Omega}$ in $\mathcal{F}_\Omega$ specifies its original domain. Now, the faces can be classified in two types: *active* and *passive*. The following definition describes this notion.

Definition 2. A face F, $F \in \mathcal{M}_\Omega$, is called *active*, if $F \in \mathcal{F}_\Omega$. Otherwise, F is *passive*.

A tetrahedron E is created by selecting an active face F_i, $F_i \in \mathcal{F}_\Omega$, $i = 1, 2, \ldots, n_F$, and testing whether F_i generates a valid element. This particular selected active face is called *base face*. For this face, an "ideal" node is determined. This node is defined as follows.

Definition 3. A candidate node is said to be an *ideal node*, denoted by P_{ideal}, if its position is calculated in relation to a base face $F_i = P_1P_2P_3$, $F_i \in \mathcal{F}_\Omega$, $i = 1, 2, \ldots, n_F$, such that it maximizes the adopted shape quality metric $\mathcal{Q}_E$ of the element $E = P_1P_3P_2P_{ideal}$.

The ideal node P_{ideal} is calculated by iterations as in Frey et al. (1998) so that the resulting tetrahedron is almost equilateral. To check the quality of an element, it was used a tetrahedron shape quality metric ($\mathcal{Q}_E$) defined by

Algorithm 1 Main scheme

1: $\mathcal{F}_\Omega \leftarrow \mathcal{M}_{\partial\Omega}$.
2: Define element size h_E.
3: Stage 1 – Generation of well shaped elements.
4: Stage 2 – Generation of elements to enforce convergence.
5: If $\mathcal{F}_\Omega = \emptyset$ then
6: $\quad$ $\mathcal{M}_\Omega$ was successfully generated.
7: Else
8: $\quad$ Convergence deficiency.
9: End If

Definition 4 (Dompierre et al. (1998))**.** A *shape quality metric* of a tetrahedron E, or simply a *shape metric* of E, is a continuous scalar function $\mathcal{Q}_E : \mathbb{R}^3 \rightarrow \mathbb{R}$, satisfying the following conditions:

S1. $\mathcal{Q}_E$ is invariant under translation, rotation, reflection and uniform scaling of E;

S2. $0 \leq \mathcal{Q}_E \leq 1, \forall E$;

S3. $\mathcal{Q}_E = 1 \Leftrightarrow E$ is a regular tetrahedron;

S4. $\mathcal{Q}_E = 0 \Leftrightarrow E$ is degenerated.

In this algorithm, $\mathcal{Q}_E$ is equal to the minimum solid angle of a tetrahedron. Considering an element $E = P_1P_2P_3P_4$, the solid angle (θ_i) in each vertex P_i, $i = 1, 2, 3, 4$, is equal to the surface area of a spherical triangle delimited by the intersection between E and an unit sphere centering in P_i. For details about the minimum solid angle and other shape metrics see Parthasarathy et al. (1993), Liu and Joe (1994) and Naylor (1999).

Whenever an element E is created, $\mathcal{F}_\Omega$ advances into $\mathbf{\Omega}$. The domain $\mathbf{\Omega}$ will be totally discretized and the mesh $\mathcal{M}_\Omega$ completely defined, when $\mathcal{F}_\Omega = \emptyset$. At this moment, Alg. 1 returns $\mathcal{M}_\Omega$ and the discretization process finalizes. If any triangular face is left in $\mathcal{F}_\Omega$ after Stage 2, the discretization process will fail. This is an indicative of deficient convergence of advancing front based algorithms. Convergence problems are solved by restarting the entire process with a new $\mathcal{M}_{\partial\Omega}$ or applying a perturbation δ over the element size (h_E), i.e., $h_E = (1 + \delta)\, h_E$. Caution is necessary in the application of δ, because it affects the distribution of h_E throughout the domain. In this work, $|\delta| \leq 0.15$ was used.

2.1 Stage 1 – Well Shaped Elements

The algorithm of this stage is similar to that of Löhner and Parikh (1988), and Peraire and Peiró (1992), but with the addition of validity tests analogous to those of Jin and Tanner (1993). However, the algorithm here proposed has four levels of quality, similarly to the multifrontal scheme presented by Rassineux (1998).

The steps of Stage 1 are shown in Alg. 2. $\mathcal{F}_{Rej}$ is a list created to store the faces from $\mathcal{F}_\Omega$ that did not generate tetrahedra satisfying the imposed quality restrictions, i.e., it stores rejected faces. The advancing front algorithm of Stage 1 is executed for each of the four quality levels. After each execution, $\mathcal{F}_\Omega$ must be empty. Then, rejected faces are set as actives, and the process restarts only if some element was created. When it is not possible to generated elements satisfying the imposed restrictions, a new quality level is created by relaxing the restrictions, and the discretization process is restarted.

Algorithm 2 Stage 1 – Generation of well shaped elements

```
1: Define minimum quality values.
2: For each quality level, do:
3:     Repeat
4:         Execute the advancing front algorithm of Stage 1;
5:         F_Ω ← F_Rej;
6:     While the elements have been generated.
7: End For
```

The advancing front algorithm applied at Stage 1 is shown in Alg. 3. While there are faces in $\mathcal{F}_\Omega$, the steps from 2 to 8 will be executed. For each base face $F = P_iP_jP_k$, $F \in \mathcal{F}_\Omega$, an ideal node P_{ideal} is determined. Initially, P_{ideal} is the center of a spherical region $\mathbf{S}$, with radius $r_\mathbf{S} = \max\left(\ell_{P_iP_{ideal}}, \ell_{P_jP_{ideal}}, \ell_{P_kP_{ideal}}\right)$, where $\ell_{P_iP_j}$ is the length of the edge $L = P_iP_j$, used to search preexisting nodes near F. Like in Löhner and Parikh (1988), an *octree* data structure was used to speed up this operation. Every node P, $P \in \mathbf{S}$, is said to be in a list $\mathbf{L}_P$ of preexisting candidate nodes to form a valid tetrahedron with base F. The candidate node P, $P \in \mathbf{L}_P$, that results in a tetrahedron $E = P_iP_kP_jP$ with greatest shape quality metric $\mathcal{Q}_E$, satisfying the validity and quality tests, is selected. Thus, a new element is created, and $\mathcal{F}_\Omega$ and $\mathcal{M}_\Omega$ must be updated. If none of the nodes P, $P \in \mathbf{L}_P$, results in a valid element, then the ideal node P_{ideal} is verified. In case the tetrahedron $E = P_iP_kP_jP_{ideal}$ satisfies the validity and quality tests, a new element and a new node (P_{ideal}) are created. Then, $\mathcal{F}_\Omega$ and $\mathcal{M}_\Omega$ are updated. If it is not possible to generate an element with this base face, F is moved from $\mathcal{F}_\Omega$ to $\mathcal{F}_{Rej}$.

2.2 Stage 2 – Enforcing Convergence

It is said that the convergence is enforced at this stage because the quality restrictions are relaxed. Therefore, it is interesting to control the quantity of generated elements at this stage so that it does not decrease significantly the global mesh quality. Ideally, the entire domain is discretized at Stage 1 resulting in an mesh with maximum quality, considering the maximum value obtained by the current algorithm. However, the random evolution of $\mathcal{F}_\Omega$ originates well distorted polyhedra of hard decomposition, even allowing the insertion of Steiner nodes (see Schönhardt (1928); Senechal (1981); Chazelle (1984); Ruppert and Seidel (1992)). Thus, it is clear that Stage 2 is fundamental to ensure the convergence of this method.

The algorithm for Stage 2 are similar to Alg. 2. However, the initial quality conditions are less restrictive than in Stage 1. As in Stage 1, while there are faces in $\mathcal{F}_\Omega$ the advancing front algorithm (see Alg. 4) is traversed. A face $F = P_iP_jP_k$, $F \in \mathcal{F}_\Omega$, is selected and the position of P_{ideal} is calculated. The list $\mathbf{L}_P$ of preexisting nodes inside a spherical region $\mathbf{S}$, with center P_{ideal} and radius $r_\mathbf{S}$, is constructed. If a node P, $P \in \mathbf{L}_P$, forms an element

Algorithm 3 Advancing front algorithm of Stage 1

1: While $\mathcal{F}_\Omega \neq \emptyset$, do:
2: Select a face $F = P_iP_jP_k$, $F \in \mathcal{F}_\Omega$.
3: Calculate P_{ideal} with relation to F.
4: Mount a list $\mathbf{L}_P$ of preexisting candidate nodes.
5: Select the node P, $P \in \mathbf{L}_P$, that forms an element $E = P_iP_kP_jP$ satisfying validity and quality tests, and with the greatest value of $\mathcal{Q}_E$ among all $P \in \mathbf{L}_P$. If such a node exists, continue from step 8.
6: If P_{ideal} forms a tetrahedron $E = P_iP_kP_jP_{ideal}$ satisfying validity and quality tests for ideal node, continue from step 8.
7: Move F from $\mathcal{F}_\Omega$ to $\mathcal{F}_{Rej}$ and continue from step 9.
8: Update $\mathcal{F}_\Omega$ and $\mathcal{M}_\Omega$;
9: End While

Algorithm 4 Advancing front algorithm of Stage 2

While $\mathcal{F}_\Omega \neq \emptyset$, do:
    Select a face $F = P_iP_jP_k$, $F \in \mathcal{F}_\Omega$.
    Calculate P_{ideal} with relation to F.
    Mount a list $\mathbf{L}_P$ of preexisting candidate nodes.
    Select the node P, $P \in \mathbf{L}_P$, that forms an element $E = P_iP_kP_jP$ satisfying validity and quality tests, and with the greatest value $\mathcal{Q}_E$ among all $P \in \mathbf{L}_P$. If such a node exists, continue from step 8.
    If P_{ideal} forms a tetrahedron $E = P_iP_kP_jP_{ideal}$ satisfying validity and quality tests for the ideal node, continue from step 8. Otherwise, decrease the height of P_{ideal} with relation to F and restart step 6.
    Remove F from $\mathcal{F}_\Omega$, store it in $\mathcal{F}_{Rej}$ and continue from step 9.
    Update $\mathcal{F}_\Omega$ and $\mathcal{M}_\Omega$;
End While

$E = P_iP_kP_jP$ satisfying validity and quality tests, resulting in the largest value of $\mathcal{Q}_E$ among all candidate nodes in $\mathbf{L}_P$, this node is selected. Consequently, a new element is created, and $\mathcal{F}_\Omega$ and $\mathcal{M}_\Omega$ must be updated. Otherwise, P_{ideal} is used as a candidate node to generate a tetrahedron $E = P_iP_kP_jP_{ideal}$. If E satisfies validity and quality tests, an element and a node (P_{ideal}) are created. After this, $\mathcal{F}_\Omega$ and $\mathcal{M}_\Omega$ are updated.

If $E = P_iP_kP_jP_{ideal}$ do not pass the tests, the height of P_{ideal} in relation to face F is decreased by a factor of 20% and step 6 of Alg. 4 is restarted. If the minimum allowed height of P_{ideal} is reached (e.g. 20% of the original height) and the element do not pass the tests, face F is removed from $\mathcal{F}_\Omega$ and stored in $\mathcal{F}_{Rej}$. Some authors (Ruud and Wille, 1990; Peraire and Peiró, 1992; Frykestig, 1994) treat these nodes as members of a separated list called *escape nodes*.

2.3 Validity and Quality Tests

An element can be accepted as valid only if it satisfies validity and quality tests. These tests are just verifications of validity and quality criteria to ensure mesh consistence and to avoid the propagation of distorted elements into the domain. The validity criterion applied is the following.

Definition 5 (Validity Criterion)**.** Let be $\mathcal{F}_\Omega$ a front of a domain $\mathbf{\Omega}$, $\mathbf{\Omega} \subset \mathbb{R}^3$, with tetrahedral mesh $\mathcal{M}_\Omega$. An element $E = P_iP_jP_kP_l$ is said to be *valid* if it satisfies the following conditions:

V1. $V_E > 0$, and;

V2. If $E \cap F \neq \emptyset$, $\forall F \in \mathcal{F}_\Omega$, then $E \cap F \prec E$ and $E \cap F \prec F$.

where V_E denotes the volume of E.

Condition V1 of Def. 5 prevents the creation of degenerated tetrahedra, and condition V2 requires that the intersection between a tetrahedron E and every face F, $F \in \mathcal{F}_\Omega$,

must be empty, a vertex or an edge of either, or only equal to F. In this work, the intersection between two triangular faces is checked based on orientation predicates similar to the Guigue and Devillers (2003).

The quality criterion tries to guarantee local (i.e. element) and global (i.e. with relation to active faces) quality. It is used to check if an element is suitable to be inserted in $\mathcal{M}_\Omega$. Its definition is as follows.

Definition 6 (Quality Criterion)**.** Let be $\mathcal{F}_\Omega$ a front of a domain $\mathbf{\Omega}$, $\mathbf{\Omega} \subset \mathbb{R}^3$, with tetrahedral mesh $\mathcal{M}_\Omega$. An element $E = P_iP_jP_kP_l$ is said to be *suitable* if it satisfies the following conditions:

Q1. $\mathcal{Q}_E \geq min_{\mathcal{Q}_E}$.

Q2. The minimum distance between new faces of E and neighbor nodes in $\mathcal{F}_\Omega$, d_{fp}, is greater than or equal to $min_{d_{fp}}$.

Q3. The minimum distance between new edges of E and neighbor nodes in $\mathcal{F}_\Omega$, d_{lp}, is greater than or equal to $min_{d_{lp}}$.

Q4. The minimum distance between new edges of E and neighbor edges in $\mathcal{F}_\Omega$, d_{ll}, is greater than or equal to $min_{d_{ll}}$.

Q5. If P_l is a new node, the minimum distance between it and neighbor faces in $\mathcal{F}_\Omega$, d_{pf}, is greater than or equal to $min_{d_{pf}}$.

Q6. The angle between new faces of E and incident edges, θ_{fl}, is greater than or equal to $min_{\theta_{fl}}$.

Q7. The angle between new faces of E and adjacent faces, θ_{ff}, is greater than or equal to $min_{\theta_{ff}}$.

Condition Q1 from Def. 6 ensures the minimum element quality levels. On the current implementation, Stage 1 initializes with $min_{\mathcal{Q}_E} = 0.60$, which is maintained until the second loop over $\mathcal{F}_\Omega$. During the last two loops at Stage 1, $min_{\mathcal{Q}_E}$ is equal to 0.40. From the first to the third loop of Stage 2, $min_{\mathcal{Q}_E}$ is equal to 0.40, 0.30 and 0.20, respectively. And if there are faces in $\mathcal{F}_\Omega$, the fourth loop is traversed with $min_{\mathcal{Q}_E} \approx 0.0$ to enforce convergence. Conditions Q2 – Q7 from Def. 6 are posed to maintain a minimum distance between active faces and new nodes, edges or faces. This is necessary to preserve an empty region between opposed front faces that can be discretized.

The minimum values of distances and angles were obtained based on a unit regular tetrahedron E. For such an element, it is observed the following properties:

- The minimum distance between any vertex P, $P \in E$, and its corresponding opposed face is equal to $0.81h_E$.
- The minimum distance between any vertex P, $P \in E$, and a non-incident edge L, $L \in E$, $P \notin L$, is equal to $0.86h_E$.

- The minimum distance between any edge L, $L \in E$, and its corresponding opposed edge is equal to $0.70h_E$.
- The angle between any face F, $F \in E$, and an incident edge to F is equal to $54.7°$.
- The angle between any face F, $F \in E$, and an adjacent face to F is equal to $70.5°$.

From these properties, initial minimum values were established: $min_{d_{fp}} = 0.67h_E$, $min_{d_{lp}} = 0.67h_E$, $min_{d_{ll}} = 0.60h_E$, $min_{\theta_{fl}} = 45°$, and $min_{\theta_{ff}} = 60°$. After each loop over $\mathcal{F}_\Omega$, they are decreased about a factor of 15%. At the end, they become close to zero. In this case, an element is judged valid only checking the validity criterion of Def. 5.

New nodes that results in valid elements must satisfy an additional quality criterion as described in condition Q5 from Def. 6. This condition avoids the insertion of nodes that are close to the active faces. During Stage 1, $min_{d_{pf}}$ is made equal to $0.67h_E$ to guarantee that new nodes are located in an optimum distance from $\mathcal{F}_\Omega$. At the begin of Stage 2, $min_{d_{pf}}$ is maintained equal to $0.67h_E$. However, $min_{d_{pf}}$ is decreased about a factor of 50% after each loop over $\mathcal{F}_\Omega$ until $min_{d_{pf}}$ is approximately equal to zero in the last loop.

2.4 Front Management

Active faces are ordered using a heap $\mathcal{H}_\Omega$ (Löhner, 1988). The area of each face $F \in \mathcal{F}_\Omega$ was used as the key ordering. Thus, a face with smallest area among all in $\mathcal{F}_\Omega$ will be always at the root of $\mathcal{H}_\Omega$. When an active face is selected from $\mathcal{F}_\Omega$, the number of the face in the root of $\mathcal{H}_\Omega$ is returned. Although h_E is constant throughout $\mathbf{\Omega}$, this strategy is used to prevent large elements going into zones of small elements. The rejected faces list $\mathcal{F}_{Rej}$ is ordered in a similar way, using another heap $\mathcal{H}_{Rej}$.

New front faces can arise after each created element. In most of algorithms, these faces are stored in $\mathcal{F}_\Omega$ and are inserted in $\mathcal{H}_\Omega$. Nevertheless, in the actual implementation, they are stored in $\mathcal{F}_{Rej}$ and are inserted in its corresponding heap $\mathcal{H}_{Rej}$, in which the respective areas are the value of key. Also, when a base face F is rejected from $\mathcal{F}_\Omega$, the face F is moved to $\mathcal{F}_{Rej}$ and inserted in $\mathcal{F}_{Rej}$, but with a null value of key instead of its area. This scheme keeps oldest faces located before newest faces in $\mathcal{F}_{Rej}$. Then, when the faces in $\mathcal{F}_{Rej}$ become actives, $\mathcal{F}_\Omega$ will advance from outer to the inner part of the subdomain delimited by the current front. This is used to decrease the effects of the random way the front develops without significant changes of the advancing front algorithm.

3. RESULTS AND DISCUSSIONS

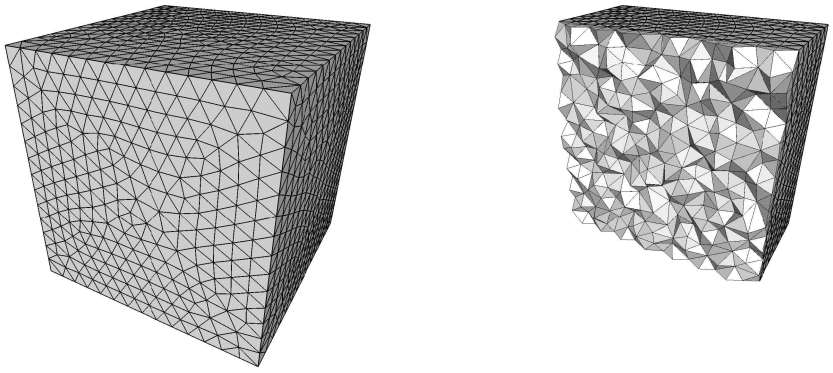
In this section, a classical benchmark, the unit cube, and two other examples of arbitrary geometries are presented. The quality parameters of the generated meshes and the corresponding computational cost are shown, allowing a full analysis of the proposed algorithm.

All results were obtained using C programming language and Microsoft® Visual C++® 6.0 on a AMD Athlon™ XP1800+ processor (512MB of RAM, 256KB of cache, and Windows® XP Professional SP2 as operational system).

3.1 Unit Cube

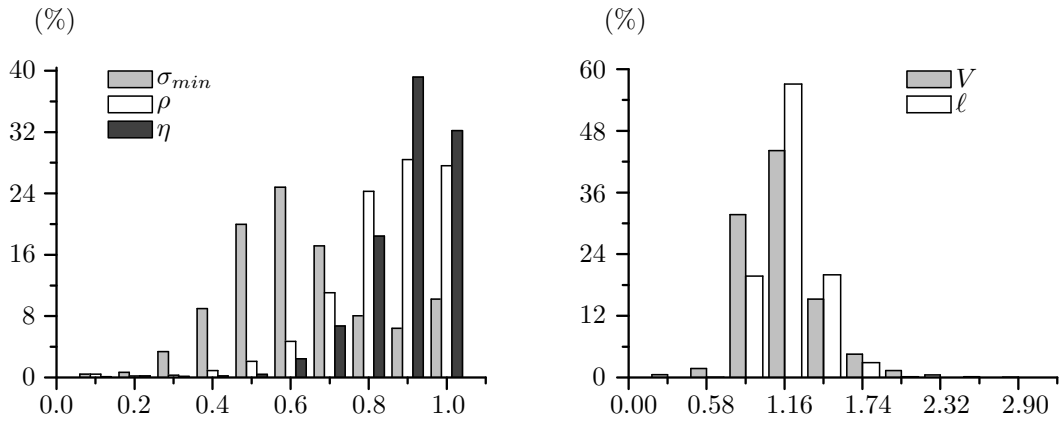
The input data for this example is a surface mesh with 1341 vertices and 2678 faces (see Fig. 1(a)). The final tetrahedral mesh, whose cross section is shown in Fig. 1(b), consists of 3723 vertices, 23222 edges, 37661 faces and 18161 elements. Approximately 72% of the elements was generated in Stage 1 and 28% in Stage 2. The total CPU time was 35.75 seconds ($31.47s$ in Stage 1 and $4.28s$ in Stage 2).

The relation between maximum and minimum edge lengths (ℓ) is around 6 (six). The minimum volume (V) is equal to 4.25×10^{-8} (see Tab. 1) indicating the presence of flat elements mostly generated at Stage 2. In spite of this, it is noted in Fig. 1(b) an uniform distribution of element sizes h_E. Figure 2(a) shows a quasi-Gaussian distribution of the minimum solid angle (σ_{min}), characteristic of σ_{min} distributions as observed by Dompierre et al. (1998). About 66%, 96% and 99% of the elements has σ_{min}, radius ratios (ρ), and mean ratios (η), respectively, greater than 0.50. The distributions of Fig. 2(b) show the ability of the algorithm to reach the required element size. In addition, the mean value of h_E was equal to 0.0833 (14% greater than h_E in $\mathcal{M}_{\partial\Omega}$).



(a) Surface mesh (b) Cross section

Figure 1: Unit cube.



(a) Tetrahedron shape metrics (b) Normalized V and ℓ

Figure 2: Distributions of tetrahedron shape metrics (σ_{min}, ρ, and η) and normalized volumes (V) and edge lengths (ℓ) corresponding to the tetrahedral mesh of the unit cube.

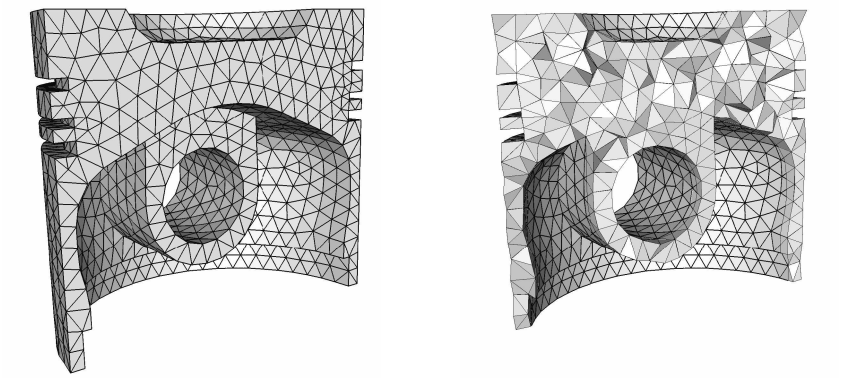
Table 1: Statistical data from the tetrahedral mesh of the unit cube.

	Non-Normalized				Normalized		
	min	μ	max	σ	min	max	σ
ℓ	2.50×10^{-2}	8.33×10^{-2}	1.56×10^{-1}	1.67×10^{-2}	3.00×10^{-1}	1.881	2.00×10^{-1}
V	4.25×10^{-8}	5.50×10^{-5}	1.56×10^{-4}	1.53×10^{-5}	7.72×10^{-4}	2.834	2.79×10^{-1}
σ_{min}	8.49×10^{-4}	5.92×10^{-1}	1.000	1.92×10^{-1}	1.43×10^{-3}	1.688	3.25×10^{-1}
ρ	4.57×10^{-6}	8.03×10^{-1}	1.000	1.52×10^{-1}	5.69×10^{-6}	1.244	1.89×10^{-1}
η	9.70×10^{-3}	8.45×10^{-1}	1.000	1.20×10^{-1}	1.14×10^{-2}	1.183	1.42×10^{-1}

3.2 Piston

Figure 3(a) illustrates the surface mesh of a piston constituted of 1497 vertices and 2994 faces. The CPU time to generate the entire tetrahedral mesh was 13.36 seconds, where 11.09 seconds were spent at Stage 1 and 2.27 seconds at Stage 2. The final mesh consists of 1986 vertices, 10648 edges, 15827 faces and 7165 elements. Stages 1 and 2 have generated 65% and 35% of the mesh elements, respectivelly. Figure 3(b) illustrates a cross section of the tetrahedral mesh. In this figure, it may be observed an uniform distribution of h_E.

The ratio between maximum and minimum values of ℓ was about 5 (five), cf. Tab. 2. As in the first example, flat elements were generated to ensure the convergence of the algorithm (see minimum volume in Tab. 2). Approximately 58%, 95% and 98% of the elements presented σ_{min}, ρ, and η, respectively, greater than 0.50. Nearly 70% of the edges has normalized edge lengths ℓ ranging from 0.56 to 1.12 (see Fig. 4(b)).

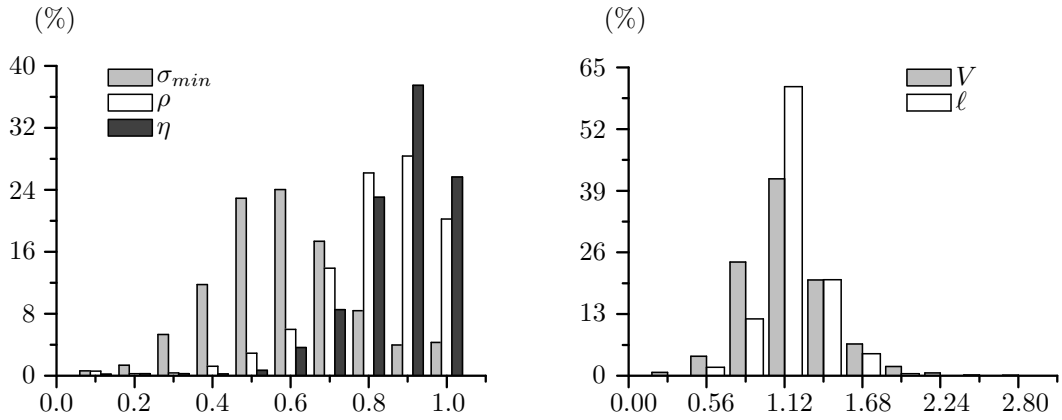


(a) Surface mesh (b) Cross section

Figure 3: Piston (courtesy of Drexel Repository).

Table 2: Statistical data from the tetrahedral mesh of the piston.

	Non-Normalized				Normalized		
	min	μ	max	σ	min	max	σ
ℓ	2.29×10^{-3}	6.27×10^{-3}	1.19×10^{-2}	1.32×10^{-3}	3.66×10^{-1}	1.896	2.11×10^{-1}
V	1.91×10^{-13}	2.38×10^{-8}	6.56×10^{-8}	7.37×10^{-9}	8.03×10^{-6}	2.757	3.09×10^{-1}
σ_{min}	1.47×10^{-5}	5.41×10^{-1}	1.000	1.75×10^{-1}	2.73×10^{-5}	1.846	3.24×10^{-1}
ρ	2.51×10^{-9}	7.74×10^{-1}	1.000	1.55×10^{-1}	3.24×10^{-9}	1.291	2.00×10^{-1}
η	5.84×10^{-4}	8.18×10^{-1}	1.000	1.25×10^{-1}	7.13×10^{-4}	1.222	1.53×10^{-1}



(a) Tetrahedron shape metrics (b) Normalized V and ℓ

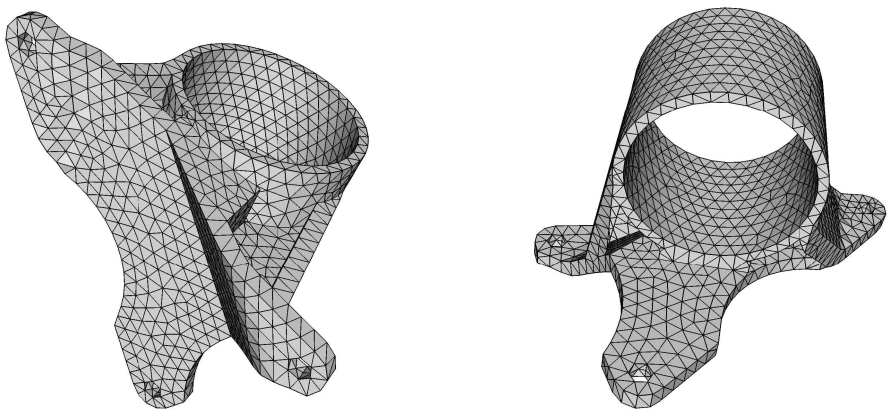
Figure 4: Distributions of tetrahedron shape metrics (σ_{min}, ρ, and η) and normalized volumes (V) and edge lengths (ℓ) corresponding to the tetrahedral mesh of the piston.

3.3 LH Mount

The surface mesh of a LH mount, shown in Fig. 5, has 1781 vertices and 3578 faces. The total CPU time to generate the entire mesh was 8.32 seconds, where 4.26 seconds were spent at Stage 1 and 4.06 seconds at Stage 2. The final tetrahedral mesh has 1939 vertices, 9848 edges, 14021 faces and 6116 elements, from which 46% were created at Stage 1 and 54% at Stage 2. Obviously, the increase in the number of generated elements at Stage 2, in relation to the other examples, has reduced the mesh quality.

As it can be inferred from Tab. 3, the ratio between the maximum and minimum values ℓ was approximately 14 (fourteen). Mean values of σ_{min}, ρ and η were 0.50, 0.74 and 0.79, respectively (see Tab. 3). These values are less than most of the values previously shown.

About 50%, 92% and 96% of the tetrahedra has values of σ_{min}, ρ and η, respectively, greater than 0.5 (cf. Fig. 6(a)). Nearly 63% of the elements and 80% of the edges have normalized volumes and edge lengths between 0.59 e 1.18 (see Fig. 6(b)).

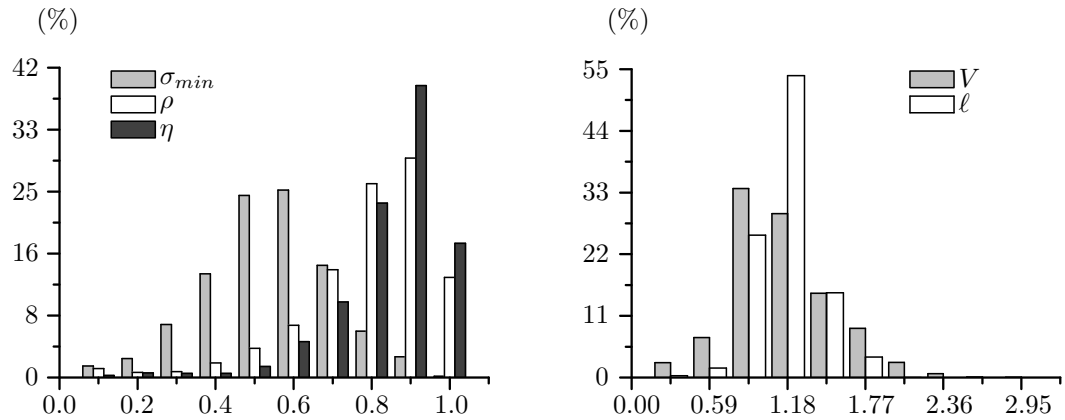


(a) (b)

Figure 5: LH mount (courtesy of Eric Saltel).

Table 3: Statistical data from the tetrahedral mesh of the LH mount.

	Non-Normalized				Normalized		
	min	μ	max	σ	min	max	σ
ℓ	5.30×10^{-3}	4.09×10^{-2}	7.41×10^{-2}	9.10×10^{-3}	1.29×10^{-1}	1.810	2.22×10^{-1}
V	1.44×10^{-9}	6.40×10^{-6}	1.87×10^{-5}	2.40×10^{-6}	2.25×10^{-4}	2.930	3.75×10^{-1}
σ_{min}	1.16×10^{-4}	4.94×10^{-1}	9.29×10^{-1}	1.60×10^{-1}	2.36×10^{-4}	1.881	3.25×10^{-1}
ρ	1.54×10^{-7}	7.42×10^{-1}	9.95×10^{-1}	1.72×10^{-1}	2.07×10^{-7}	1.342	2.32×10^{-1}
η	3.91×10^{-3}	7.90×10^{-1}	9.96×10^{-1}	1.39×10^{-1}	4.95×10^{-3}	1.261	1.76×10^{-1}



(a) Tetrahedron shape metrics (b) Normalized V and ℓ

Figure 6: Distributions of tetrahedron shape metrics (σ_{min}, ρ, and η) and normalized volumes (V) and edge lengths (ℓ) corresponding to the tetrahedral mesh of the LH mount.

3.4 Computational Cost

Early researches on advancing front based algorithms for generation of unstructured tetrahedral meshes (Löhner, 1988; Peraire and Peiró, 1992) report a computational cost $\mathcal{O}\left(N\log\left(N\right)\right)$, N being the number of generated elements, independently of the domain geometry. Jin and Tanner (1993) developed an algorithm whose computational cost changes according to different domain discretizations. They observed $\mathcal{O}\left(N^{1.15}\log\left(N\right)\right)$ for an unit cube, and $\mathcal{O}\left(N\log\left(N\right)\right)$ for an aircraft discretization. Later, Möller and Hansbo (1995) concluded that $\mathcal{O}\left(N^{p}\log\left(N\right)\right)$, with p slightly greater larger than 1.0, was a more realistic measure of the computational cost for this type of algorithm.

In fact, algorithms that employ special data structures to speed up search operations, such as *heap* (Löhner, 1988), *octree* (Löhner, 1988), and *alternating digital tree* (Bonet and Peraire, 1991), are strongly dependent on the way the generation front advances (Möller and Hansbo, 1995). In addition, intersection tests between triangular faces also contribute to the increase of the total cost (Löhner and Parikh, 1988; Löhner, 2001).

The obtained computational cost of the proposed algorithm was evaluated considering the CPU time spent in ten different discretization levels of a unit cube ($h_E = 1/5, 1/6, \ldots, 1/14$). Figure 7 shows the CPU times for generation of each discretization level. The computational cost of the proposed algorithm for the unit cube was $\mathcal{O}\left(N^{1.18}\log\left(N\right)\right)$, which is more close to that predicted by Möller and Hansbo (1995).

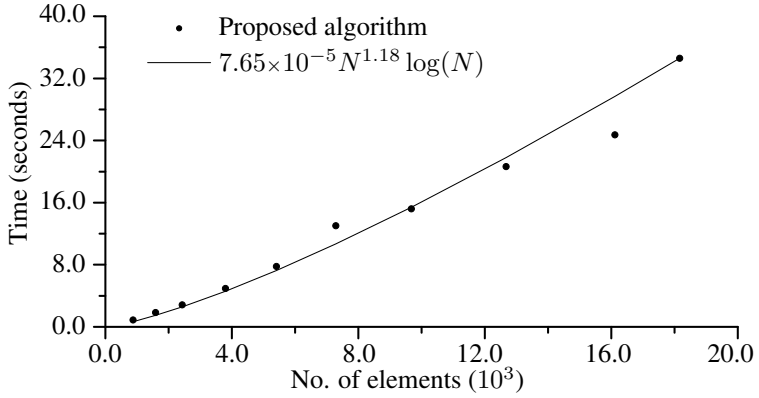


Figure 7: CPU times corresponding to ten levels of discretization of a unit cube ($h_E = 1/5, 1/6, \ldots, 1/14$).

4. CONCLUSION

An advancing front based algorithm to generate unstructured tetrahedral meshes was described. The goal of this algorithm is to generate as much well shaped elements as possible, enforcing convergence by filling up distorted void spaces with poorly shaped elements. The presented examples show that the algorithm reached a reasonable degree of stability. Good quality meshes were obtained, even without the use of any external technique to improve convergence. Moreover, the ability of the present algorithm in dealing with arbitrary geometries was clearly observed. The computational cost was also satisfactory, in accordance with previously observed results. Therefore, it may be concluded that this algorithm can be used as a core of more complex mesh generators based on the advancing front method.

Acknowledgements

The authors would like to thank the financial support from the National Council of Scientific and Technological Development (CNPq). The authors are also indebted to professors Joaquim B. Cavalcante Neto from the Federal University of Ceará (UFC) and Marco Antônio Piteri from the São Paulo State University (Unesp).

REFERENCES

Bonet, J. & Peraire, J., 1991. An Alternating Digital Tree (ADT) Algorithm for 3D Geometric Searching and Intersection Problems. *International Journal for Numerical Methods in Engineering*, vol. 31, n. 1, pp. 1–17.

Cavalcante Neto, J. B., Wawrzynek, P. A., Carvalho, M. T. M., Martha, L. F., & Ingraffea, A. R., 2001. An Algorithm for Three–Dimensional Mesh Generation for Arbitrary Regions with Cracks. *Engineering with Computers*, vol. 17, n. 1, pp. 75–91.

Chazelle, B., 1984. Convex Partitions of Polyhedra: A Lower Bound and Worst-Case Optimal Algorithm. *SIAM Journal on Computing*, vol. 13, n. 3, pp. 488–507.

Dari, E. A., 1994. *Contribuciones a la Triangulación Automática de Dominios Tridimensionales*. Tesis de doctorado, Instituto Balseiro, Río Negro, ARG.

Delaunay, B. N., 1924. Sur la Sphère Vide. In *Proceedings of the International Mathematical Congress*, volume I, pp. 695–700, Toronto, CAN. The University of Toronto Press.

Dompierre, J., Labbé, P., Guibault, F., & Camarero, R., 1998. Proposal of Benchmarks for 3D Unstructured Tetrahedral Mesh Optimization. In *Proceedings of the 7th International Meshing Roundtable*, pp. 459–478, Williamsburg, VI, USA. Sandia National Laboratories.

Frey, P. J., Borouchaki, H., & George, P.-L., 1998. 3D Delaunay Mesh Generation Coupled with an Advancing–Front Approach . *Computer Methods in Applied Mechanics and Engineering*, vol. 157, n. 1–2, pp. 115–131.

Frykestig, J., 1994. *Advancing Front Mesh Generation Techniques with Application to the Finite Element Method*. Ph.d. thesis, Chalmers University of Technology, Göteborg, SWE.

George, P.-L. & Seveno, M. E., 1994. The Advancing–Front Mesh Generation Method Revisited. *International Journal for Numerical Methods in Engineering*, vol. 37, pp. 3605–3619.

Golgolab, A., 1989. Mailleur 3D Automatique pour des Geometries Complexes. Rapports de Recherche 1004, INRIA, Rocquencourt, FRA.

Guigue, P. & Devillers, O., 2003. Fast and Robust Triangle–Triangle Overlap Test Using Orientation Predicates. *Journal of Graphics Tools*, vol. 8, n. 1, pp. 25–32.

Ito, Y., Shih, A. M., & Soni, B. K., 2004. Reliable Isotropic Tetrahedral Mesh Generation Based on an Advancing Front Method. In *Proceedings of the 13th International Meshing Roundtable*, pp. 95–106, Williamsburg, VA, USA. Sandia National Laboratories.

Jin, H. & Tanner, R. I., 1993. Generation of Unstructured Tetrahedral Meshes by Advancing Front Technique. *International Journal for Numerical Methods in Engineering*, vol. 36, pp. 1805–1823.

Liu, A. & Joe, B., 1994. Relationship between Tetrahedron Shape Mesaures. *BIT*, vol. 34, pp. 268–287.

Lo, S. H., 1985. A New Mesh Generation Scheme for Arbitrary Planar Domains. *International Journal for Numerical Methods in Engineering*, vol. 21, n. 8, pp. 1403–1426.

Löhner, R., 1988. Some Useful Data Structures for the Generation of Unstructured Grids. *Communications in Applied Numerical Methods*, vol. 4, n. 1, pp. 123–135.

Löhner, R., 2001. *Applied Computational Fluid Dynamics Techniques: An Introduction Based on Finite Element Methods*. John Wiley & Sons, New York, USA, 1 edition.

Löhner, R. & Parikh, P., 1988. Generation of Three–Dimensional Unstructured Grids by the Advancing–Front Method. *International Journal for Numerical Methods in Fluids*, vol. 8, pp. 1135–1149.

Marcum, D. L., 2001. Efficient Generation of High–Quality Unstructured Surface and Volume Grids. *Engineering with Computers*, vol. 17, n. 3, pp. 211–233.

Meagher, D., 1982. Geometric Modeling Using Octree Encoding. *Computer Graphics and Image Processing*, vol. 19, n. 2, pp. 129–147.

Möller, P. & Hansbo, P., 1995. On Advancing Front Mesh Generation in Three Dimensions. *International Journal for Numerical Methods in Engineering*, vol. 38, pp. 3551–3569.

Naylor, D. J., 1999. Filling Space with Tetrahedra. *International Journal for Numerical Methods in Engineering*, vol. 44, n. 10, pp. 1383–1395.

Parthasarathy, V. N., Graichen, C. M., & Hathaway, A. F., 1993. A Comparison of Tetrahedral Quality Mesures. *Finite Elements in Analysis and Design*, vol. 15, pp. 255–261.

Peraire, J. & Peiró, J., 1992. Adaptive Remeshing for Three–Dimensional Compressible Flow Computations. *Journal of Computational Physics*, vol. 103, pp. 269–285.

Piteri, M. A., 1998. *Geração Automática de Malhas Hieráquico–Adaptativas em Domínios Bidimensionais e Tridimensionais*. Tese de doutorado, Universidade Técnica de Lisboa, Instituto Superior Técnico, Lisboa, PT.

Rassineux, A., 1998. Generation and Optimization of Tetrahedral Meshes by Advancing Front Technique. *International Journal for Numerical Methods in Engineering*, vol. 41, n. 4, pp. 651–674.

Ruppert, J. & Seidel, R., 1992. On the Difficulty of Triangulating Three–Dimensional Non–Convex Polyhedra. *Discrete & Computational Geometry*, vol. 7, n. 3, pp. 227–253.

Ruud, H. K. & Wille, S. Ø., 1990. An Advancing Front Algorithm for Three Dimensional Mesh Generation. In *NUMETA 90: Proceedings of the 3rd International Conference on Advances in Numerical Methods in Engineering: Theory and Application*, volume 2, pp. 1141–1148, London, UK. Elsevier Applied Science.

Samet, H., 1989. *The Design and Analysis of Spatial Data Structures*. Addison–Wesley, Reading, MA, USA.

Schönhardt, E., 1928. Über die Zerlegung von Dreieckspolyedern in Tetraeder. *Mathematische Annalen*, vol. 98, pp. 309–312.

Senechal, M., 1981. Which Tetrahedra Fill Space? *Mathematics Magazine*, vol. 54, n. 5, pp. 227–243.