1 Introduction

In numerical simulations using the finite element method (FEM), two important aspects to be considered are the automatic generation of the model’s finite element mesh and the definition of the level of refinement associated with this mesh. For the first aspect, there are a variety of algorithms with different techniques to generate planar, surface, and volumetric meshes. The second aspect, i.e., the level of refinement, is usually defined manually by a specialist based on his/her own experience. However, this refinement should consider the fact that the density of the generated elements varies according to the gradient of the obtained solution, which is initially unknown. In this context, this work presents a methodology for adaptive generation of 3D finite element meshes using geometric modeling with multiregions and parametric surfaces, considering a geometric model described by curves, surfaces, and volumes. This methodology is applied in the simulation of stress analysis of solid structures using a displacement-based finite element method and may be extended to other types of 3D finite element simulations. The adaptive strategy is based on an independent and hierarchical refinement of curves, surfaces, and volumes. From an initial model, new sizes of elements obtained from a discretization error analysis and from geometric restrictions are stored in a global background structure, a recursive spatial composition represented by an octree. Based on this background structure, the model’s curves are initially refined using a binary partition algorithm. Curve discretization is then used as input for the refinement of adjacent surfaces. Surface discretization also employs the background octree-based refinement, which is coupled to an advancing front technique in the surface’s parametric space to generate an unstructured triangulated mesh. Surface meshes are finally used as input for the refinement of adjacent volumetric domains, which also uses an advancing front technique but in 3D space. In all stages of the adaptive strategy, the refinement of curves, surface meshes, and solid meshes is based on estimated discretization errors associated with the mesh of the previous step in the adaptive process. In addition, curve and surface refinement takes curvature information into account. Numerical examples of simulation of engineering problems are presented in order to validate the methodology proposed in this work. [DOI: 10.1115/1.4024106]

Keywords: 3D adaptive analysis, finite element method, mesh generation
order and density of 3D finite element meshes. Lee and Xu [12] generate a surface mesh for the mid-surface of the thin-walled structure, controlling element size, and convert the surface mesh to a 3D solid mesh by extrusion. More recently, Qian and Zhang [13] and Zhang et al. [14] have generated tetrahedral and hexahedral meshing in multimaterial domains using a grid-based method that employs an octree structure, refining meshes also locally.

In global refinement, on the other hand, at each refinement step, the entire mesh is deleted and another is generated based on new sizes of elements obtained from a discretization error estimation analysis. This process is used by Kettl et al. [15] only in regions with complicated parts. Hughes et al. [16] refine the structured meshes with NURBS surfaces. Our work presents a methodology that employs the global h-refinement approach.

A previous paper [17] proposed a two-dimensional self-adaptive strategy that was able to perform simulations involving automatic generation of meshes and adaptive methods. Other works have considered the same problem through different approaches, such as the study by Shephard’s team [18], from the Rensselaer Polytechnic Institute. Cavalcante-Neto [19] proposed a technique for the generation of volumetric meshes of tetrahedral elements for arbitrary region domains. Combining this technique with the implementation of 3D error estimators, the authors defined a prototype of an environment for adaptive generation in three dimensions. However, in this previous work, the complete process was not performed automatically, i.e., the mesh was generated independently from the error estimation and had to be manually combined. Moreover, despite treating multiregions, it was not very efficient, and parametric surfaces, which are used in several types of simulations, were not considered. Other strategies for volume representation is described by Xu et al. [20].

This paper aims to present a methodology for adaptive generation of three-dimensional finite element meshes, using geometric modeling with multiregions and parametric surfaces. Basically, the whole mesh adaptive process involves three steps: (1) analysis of a finite element (FE) model with discretization error estimation; (2) construction of a background structure to store new FE sizes that take into account the estimated discretization error and curve and surface curvature; and (3) hierarchical refinement of an FE model that is represented geometrically by curves, surfaces, and volume regions. This process may be repeated until a desired maximum allowed error metric is achieved. The methodology described herein covers in detail only the last two steps since discretization error estimation can be computed through different processes [1]. However, for the sake of completeness, a summary of the adopted discretization error estimation technique is presented.

The paper is organized as follows. In Sec. 2, some background on the adopted discretization error estimation technique is summarized. Section 3 explains in detail the proposed adaptive refinement strategy. Section 4 describes all the steps required to generate the background data structure that is used to define FE sizes in the adaptive process, Section 5 presents the hierarchical refinement of curves, surfaces, and volume regions. Five examples of adaptive refinement are presented in Sec. 6 to evaluate result convergence and performance and the quality of the generated elements. Finally, in Sec. 7, the authors present comments and conclusions of this work.

2 Theoretical Background: Discretization Error Estimation and Adaptive Refinement

This theoretical background is based on a standard technique used in the literature [21]. There are different approaches for this technique, as presented by Macke rle [1], and one of them is described here. This section is a summary of an existing technique, and the authors do not aim to discuss it in detail.

To compute the local error correctly, mathematic norms are introduced to measure the discretization error. The discretization error in a displacement-based finite element solution is often quantified on the basis of a norm for the strain energy error $\|e\|$, which can be expressed in terms of stresses as

$$\|e\|^2 = \int_\Omega (\sigma - \hat{\sigma})^T D^{-1} (\sigma - \hat{\sigma}) d\Omega$$  

(1)

where $\sigma$ and $\hat{\sigma}$ are the exact and the finite element stress fields, $D$ is the constitutive matrix, and $\Omega$ is the problem definition domain.

The basic purpose of error estimators is to substitute the field $\hat{\sigma}$ obtained by means of recovery procedures (e.g., ZZ, SPR, or REP) [21–23] for the field $\sigma$, which is generally unknown. Therefore, the expression for computing the approximate (estimated) relative error distribution $\|e\|_{\text{rel}}$ can be expressed as

$$\|e\|_{\text{rel}}^2 = \int_\Omega (\sigma - \hat{\sigma})^T D^{-1} (\sigma - \hat{\sigma}) d\Omega$$  

(2)

Taking the finite element discretization into account and considering a specific finite element $i$, Eq. (2) can be rewritten as

$$\langle \|e\|_{\text{rel}}^2 \rangle = \int_{\Omega_i} (\sigma - \hat{\sigma})^T D^{-1} (\sigma - \hat{\sigma}) d\Omega_{\text{cell}}$$

(3)

where standard isoparametric elements are assumed, $\Omega_i$ is the determinant of the Jacobian transformation matrix, and $\Omega_{\text{cell}}$ is the element domain.

The energy norm for the error can be evaluated over the whole domain or part of it. The contribution of all the elements $m$ in the mesh is given by

$$\|e\|^2 = \sum_{i=1}^{m} (\|e\|)^2$$

(4)

The relative percentage error in the energy norm $\eta_i$, for the whole domain or part of it, can be obtained as $\eta_i = \|e\|/\|u\|$, where $\|u\|$ is the square root of twice the strain energy and is given by

$$\|u\| = \left( \int_{\Omega_i} \sigma^T D^{-1} \sigma d\Omega_{\text{cell}} \right)^{0.5}$$

(5)

A simple criterion to achieve a solution error with an acceptable level for the whole domain can be stated as $\eta_{\text{max}} \leq \eta_{\text{max}}$, where $\eta_{\text{max}}$ is the maximum allowable error defined by the analyst and the estimated error $\eta_{\text{est}}$ is given by

$$\eta_{\text{est}} = \frac{\langle \|e\|_{\text{rel}} \rangle}{\|u\| + \langle \|e\|_{\text{rel}} \rangle}^{1/2}$$

(6)

where $\|u\|$ is the energy norm obtained from the finite element solution.

A sound criterion for an “optimal mesh” consists of requiring that the energy norm error be equidistributed among the elements because it leads to meshes with high convergence rates [21]. Thus, for each element $i$

$$\|e\|_{\text{rel}} < \eta_{\text{max}} \left( \frac{\|u\|^2 + \langle \|e\|_{\text{rel}} \rangle^2}{m} \right)^{1/2}$$

(7)

By defining the ratio

$$\xi_i = \frac{\|e\|_{\text{rel}}}{\|u\|}$$

(8)

then refinement is needed if $\xi_i > 1$. By assuming a certain rate of convergence, the value of the discretization error ratio $\xi_i$ can be used to decide the new size of the element. Thus
where \( h_i \) is the initial size of the element, \( ch_i \) is the desired characteristic size of the element, and \( p \) is the polynomial order of the finite element approximation.

3 Adaptive Refinement Strategy

The three-dimensional geometric model has a topological description of the vertices, curves, surfaces, and regions, as well as an associated geometric description, which consists of the coordinates of the vertices and the mathematical representation of the curves and surfaces. The curves used here are lines, polylines, arcs, and splines. To represent surfaces, ruled, translational, bilinear, trilinear, and spline surfaces were implemented [24]. The geometric model can contain many regions. In this environment, the attributes of the simulation, such as the properties of the materials, loads, and restrictions, are associated with the geometric entities. In this framework, the entities of finite element mesh (nodes and elements) automatically receive the attributes of the geometric entities that are related to them. Using this approach, it is possible to create new meshes without losing the attributes.

Figure 1 illustrates the automatic adaptive strategy of the proposed refinement process. The input data are the initial volumetric mesh of the problem in question and the geometric entities (curves and surfaces) as well as their associated attributes. Initially, this mesh is numerically analyzed, the information required to initiate the adaptive procedure. Such information basically consists of numerical discretization errors associated with each volumetric element of the mesh. From these errors, the need for adaptive refinement is verified. If the results converge, the adaptive process is concluded with a final discretization.

If convergence is not reached, the sizes of the new elements are computed based on the estimated discretization error. All the resized data are stored in an auxiliary background structure. Although many background structures are published in the literature, as reviewed by Quadros et al. [25], the present work uses a background octree structure, which has the advantage of not only allowing fast search procedures down to internal leaves but also of representing the desired size of the elements defined by the size of the internal leaves. For these reasons, an octree is used to support the discretization of curves, surface meshes, and the volumetric mesh.

In addition to discretization error estimation, curve and surface discretization is also required, especially when the curves and surfaces present high curvatures. In such locations, the meshes should be locally refined. Therefore, new element size data, based on the geometric information of the curves and surfaces, are computed and stored in the background octree. After this procedure, the background octree is internally finalized to provide a better transition between regions with elements of highly varying sizes.

Using the size information from the background octree, the next step consists of a three level hierarchical approach to create a new volumetric mesh. First, the curves are refined based on the size of the elements stored in the octree structure. This refinement subdivides the curves into segments with sizes consistent with those of the discretization error analysis and geometric criteria. After refining the curves, the meshes associated with each of the model’s surfaces are discretized using an advancing front scheme in parametric space. This meshing scheme starts by subdividing curves on the boundary of each surface. Geometric curvature information is considered in surface refinement because the background octree takes this information into account. The last stage of the adaptive refinement process is related to the discretization of the domains of the model’s regions. Such discretization uses a 3D advancing front technique that starts from the triangulated meshes associated with the boundary surfaces of each 3D region, also considering the sizes of the elements provided by the background octree. As can be seen, this adaptive meshing methodology supports multiregions in a consistent manner, considering curve and surface curvature information in addition to the estimated discretization error.

Finally, a new discretization error analysis is performed to assess the quality of the results. If convergence is not obtained, the whole adaptive process is repeated as described above.

The next sections detail the proposed refinement strategy.
4 The Background Octree

An octree is a tree data structure based on a cell with eight children. Each cell of an octree represents a cube in the physical space. Each child represents one octant of its parent. On the leaves of the tree are the computational cells of the grid. In this work, the background octree has two main objectives. The first is to develop local guidelines used to define the discretization of curves and surfaces. The second is to define the sizes of tetrahedral elements to be generated during the advancing front procedure. The octree generation includes four steps. In the first step, the octree is initialized based on the input mesh data, which are the new element sizes obtained in the discretization error analysis. The second step refines the octree based on the geometric curvatures of curves. In the third step, the octree is refined based on the geometric curvatures of surfaces. Finally, the octree is refined in order to obtain a better transition between the sizes of the elements generated in the advancing front surface refinement.

4.1 Background Octree Refinement Based on Error Analysis. Initially, a bounding cube is created based on the maximum range of the three Cartesian coordinates of the input model. This cube is the octree’s root cell. In the first step of the octree refinement, each discretization error result of an element is used to determine the local depth of the subdivision. The characteristic size $\chi_i$ of each element is calculated and the octree cell containing the element’s central point is determined. If the size of the cell edge is larger than the calculated characteristic size, then this cell is subdivided into eight smaller cells. This process is repeated recursively and finishes when the size of the cell is smaller than the given size. This process is repeated for every element of the current FE mesh.

The characteristic size $\chi_i$, which is used in the refinement of the background octree, is calculated as follows. First, the volume of the existing element $Vol_i$ is obtained. Then, the length of edge $L_i$, considering an equilateral tetrahedron with the same volume, is estimated as

$$L_i = \sqrt[3]{\frac{12Vol_i}{\sqrt{2}}}$$

(10)

The $L_i$ value is used to obtain the characteristic size $\chi_i$, Eq. (9), as proposed by Zienkiewicz and Taylor [21].

The background octree works as a density function to guide the adaptive process. It could be replaced by other functions if desired, although it has the advantage of also allowing fast search procedures down to internal leaves. It could also have a different orientation to better adapt to models that are not parallel to Cartesian coordinates. However, the bounding cube parallel to the Cartesian coordinates is easier and faster to implement and usually gives very good results.

4.2 Octree Refinement Based on Curve Curvature. In some cases, when only the discretization error is considered in the adaptive process the new generated mesh (in the following step of the process) does not respect the actual geometry of the model’s curves. This behavior occurs when parts of a curve, for example, present high curvatures in a region where the discretization error is low. In these situations, it is necessary to refine the background octree based on the curvatures of the curves to preserve the original geometric characteristics of the model.

The methodology used to refine the curves of the model based on their curvatures is a one-dimensional version of the procedure applied to discretize the background octree. The refinement of each curve employs a recursive spatial numbering technique similar to a binary tree data structure [26], as illustrated in Fig. 2.

The main purpose is to generate a discretization on a curve according to its curvatures. The curvatures are calculated for specific curve segments. At first, the whole length of the curve is considered as the segment to be tested. If the curvature of the segment is lower than the maximum allowed curvature $\theta_{max}$, the process is interrupted. Otherwise, the segment is recursively subdivided in two segments, and each one is tested in the same
Fig. 3 Approximating the curvature by circular arcs

way, until the maximum curvature criterion is satisfied. At the end of this process, all the curve segment sizes and their middle points are transported to the background octree, using the same procedure explained in Sec. 4.1. Figure 2 demonstrates this curve refinement process. In this figure, a binary tree is used in each level of the recursive refinement.

In fact, the curvatures of the curves are measured indirectly considering local approximation by circular arcs. Let $T_i$ and $T_j$ be the tangent vectors at vertices $p_i$ and $p_j$ (see Fig. 3). Consider the circle arc $s_{ij}$ going from $p_i$ to $p_j$, $l_{ij}$ is the actual curve length from $p_i$ to $p_j$, and $d_{ij}$ is the distance between the two vertices. Then the angle $\theta_{ij}$ of $s_{ij}$ can be estimated as

$$\theta_{ij} \approx \arccos \left( \frac{T_i \cdot T_j}{|T_i| |T_j|} \right)$$

(11)

However, this curvature approximation is not sufficient to detect high curve curvatures. For this reason, Eq. (11) is used together with another equation

$$\frac{l_{ij}}{d_{ij}} \leq \frac{\theta_{\text{max}}}{2 \cdot \sin \left( \frac{\theta_{\text{max}}}{2} \right)}$$

(12)

To understand the need for Eq. (12), see for example the case of level 0 in Fig. 2 with Eq. (11). In this case, the tangent vectors from only two vertices can be lower than the maximum allowed curvature. Considering also Eq. (12), the segment is subdivided into two segments, as shown in level 1. In this example, the process is repeated until level 5 to satisfy both equations.

4.3 Octree Refinement Based on Surface Curvature. After the background octree is refined considering curve curvatures, the octree refinement is increased based on surface curvatures. This step captures high curvatures of surfaces, computes the required element sizes, and their locations, and passes this information to the background octree. The reason to perform this step is the same as the previous stage: to represent the original geometry of the model.

As in Sec. 4.2, the methodology applied to refine the surfaces of the model based on their curvatures is a two-dimensional version of the procedure used to discretize the background octree: a background quadtree, which is created similarly to the one presented by Miranda and Martha [27]. This way of computing the curvatures has shown to be efficient and robust, and this is the main reason for its adoption. The background quadtree generation follows some steps:

- quadtree initialization based on given boundary edges
- refinement to force maximum cell size
- refinement to provide minimum size disparity for adjacent cells
- refinement to force minimum curvature difference between adjacent cells (this stage is explained ahead)

As described in detail by Miranda and Martha [27], the first step has some modifications in relation to the original 3D algorithm [28]. The second and third steps have not changed. The fourth step was added to take high surface curvatures into account.

The fourth step of the quadtree generation refines this auxiliary structure to force a minimum curvature difference between adjacent cells. Initially, the algorithm stores in each cell gradient vectors of the quadtree evaluated at the center of the cell. Then, it computes a vector normal to the surface of each cell. Finally, the algorithm obtains the cosine of the angle between the normal vectors, $N_A$ and $N_B$, of the two adjacent cells

$$\cos(\theta) = \frac{N_A \cdot N_B}{|| N_A || || N_B ||}$$

(13)

and compares it to a minimum value $\cos(\theta_{\text{min}})$. This kind of comparison is similar to comparing the angle between the normal vectors and the maximum angle. If $\cos(\theta)$ is smaller than $\cos(\theta_{\text{min}})$, then a new cell size $H_{\text{new}}$ is obtained from the current size $H_{\text{old}}$ as

$$H_{\text{new}} = \frac{H_{\text{old}}}{\cos(\theta_{\text{min}}) \cdot \cos(\theta)}$$

(14)

This new size is used to locally refine the adjacent cells of the quadtree. This process is repeated recursively for every cell. The new element sizes stored in the auxiliary surface quadtree are transferred to the global background octree. At the end of this step, the background octree is refined considering the geometric curvatures of all of the model’s surfaces.

4.4 Octree Final Refinement. The previous step can leave large octree cells in the interior of a 3D region. In the first step of this final stage, the octree is refined to guarantee that no cell in the interior is bigger than the largest cell on the boundary. This will avoid excessively large elements in the domain interior. The octree is, subsequently, processed to force a single difference level between neighboring cells. This leads to a natural transition between regions with different degrees of refinement. This refinement is performed by traversing the octree and examining the difference in tree depth between adjacent cells. If the difference is larger than one level, the adequate cells are refined until the criterion is satisfied.

Figures 4–8 depict the external appearance of the background octree of a hypothetical model following the steps described previously.
5 Hierarchical Refinement

After the construction of the background octree, considering the new element sizes based on the discretization error and on curvatures of the geometric model, the final step is to regenerate the mesh of the whole model. As mentioned previously, it is assumed that the three-dimensional geometric model has a topological description of the vertices, curves, surfaces, and regions, as well as an associated geometric description, which consists of the coordinates of the vertices and the mathematical representation of the curves and surfaces. The geometric model can contain many regions. Mesh regeneration employs a hierarchical refinement of (1) curves, (2) surfaces, and (3) regions. The process is described in the following subsections.

5.1 Curve Refinement. The methodology used to refine the model’s curves is similar to the one presented in Sec. 4.2. The curve refinement process is explained by means of an example shown in Fig. 9, which also depicts a binary tree data structure for each refinement level.

At the beginning of the process, a curve is defined by its mathematical geometric description and by two nodes (initial and final points), as shown in Fig. 9, level 0. Then, the curve length and middle node are obtained. From the middle node, one can determine the cell in the background octree where this node is located. A comparison is then made to verify whether the segment size is smaller than that of the corresponding cell. If the criterion is satisfied, the curve refinement process ends considering the
nodes generated so far. Otherwise, the new node is inserted (level 1) on the curve, this curve is subdivided in two partitions (level 2), and each one is tested in the same way, until the criterion is met (levels 2–4).

5.2 Surface Refinement. Surface mesh generation is based on the algorithm presented by Miranda and Martha [27]. This algorithm is applied to the generation of triangular meshes on each surface with arbitrary geometry, using its parametric description. The parametric description is used because it is common and efficient, since the surface mesh is generated using two-dimensional triangulation techniques. However, additional length and angle corrections are needed to consider metric distortions between parametric and 3D Cartesian spaces. With this procedure, generated triangles present good shape in 3D space.

Figure 10 shows an edge AB that will create a triangle with point NC in parametric space. mAB is the midpoint of points A and B. h2D is the height of the triangle, and N2D is a unit vector from points mAB to NC. This vector is normal to edge AB in 3D space. Given a desired size h2D in 3D space obtained from the background structure, the distance h2D in parametric space is given by

\[ h_{2D} = \frac{h_{3D}}{\sqrt{F_{mAB}U_{N2D}^2 + 2G_{mAB}U_{N2D}V_{N2D} + K_{mAB}V_{N2D}^2}} \]  

where

\[ F = \lVert \overrightarrow{\sigma}_U(U, V) \rVert \cdot \lVert \overrightarrow{\sigma}_V(U, V) \rVert \]
\[ G = \lVert \overrightarrow{\sigma}_U(U, V) \rVert \cdot \lVert \overrightarrow{\sigma}_V(U, V) \rVert \]
\[ K = \lVert \overrightarrow{\sigma}_U(U, V) \rVert \cdot \lVert \overrightarrow{\sigma}_V(U, V) \rVert \]

in which \( \overrightarrow{\sigma}_U(U, V) \) and \( \overrightarrow{\sigma}_V(U, V) \) are the gradient vectors at a point on the surface in parametric directions U and V. From Eq. (15), \( E_{mAB}, F_{mAB}, \) and \( G_{mAB} \) are obtained at point mAB of Fig. 10. Vector \( N_{2D} \), normal to AB (in 3D space), is computed using the metrics from Eq. (16)

\[ N_{2D} = \left\{ \frac{U_{N2D}}{V_{N2D}} \right\} = \left[ \begin{array}{c} G_{mAB} \\ -F_{mAB} \\ -K_{mAB} \end{array} \right] \cdot \overrightarrow{V}_{AB} \]  

in which \( \overrightarrow{V}_{AB} \) is a unit vector from point A to point B.

The input data for the present surface mesh generation algorithm is a parametric description of the surface and a polygonal description of the boundary of the surface patch to be meshed. The surface meshing algorithm has the following steps [27]: (a) the background quadtree, generated in Sec. 4.3, is used to store local surface distortion metrics; and (b) a two-pass advancing front procedure is used to generate elements. In the first pass, the elements are generated based on geometric criteria, which produce well-shaped elements. In this phase, the background octree is also used to develop local guidelines for the new element sizes. Therefore, element sizes will take discretization error information into account. In the second pass, the elements are generated to guarantee a valid topological mesh but with no good-shape requirements. For well-conditioned surface geometries, the second pass is usually not necessary.

A smoothing technique is used to improve the quality of the mesh by relocating nodes within a patch. A general formulation for this technique is based on a generic form of a weighted Laplacian function [29].

The smoothing procedure is performed to satisfy the principle of equidistribution of nodes through the migration of nodes within a patch, which means that the nodes are relocated until they find an optimal position considering their neighborhood. Thus, relocating a particular node has an impact on its neighbors, and then the process should be repeated a number of times for all nodes until the optimal position for them is found. However, it is necessary to balance the quality improvement and the computational cost of the relocation. In this work, the smoothing procedure is repeated five times for all internal nodes in order to achieve good quality results without impacting the performance. In addition, smoothing is done in the parametric space but takes into account the metrics of size distortion the between the parametric and 3D space.

5.3 Volume Refinement. Three-dimensional mesh generation in each closed region of the model is based on a technique presented by Cavalcante-Neto et al. [28] and is used to obtain tetrahedral elements in arbitrary domains. Similarly to the procedure applied to generate surface meshes, this one is based on an advancing front technique coupled to a recursive spatial decomposition technique (octree). Originally, the algorithm employed an independent background octree in each 3D region to control the distribution of the node points generated in the interior. In the adaptive methodology proposed here, the global background octree is used for this purpose.

The algorithm was designed to meet four specific requirements: to avoid producing elements with poor aspect ratios, to generate meshes conforming to existing triangular meshes at the boundary of a domain, to generate meshes exhibiting good transitions between regions of different element sizes, and to work properly for cases in which distinct boundary nodes are geometrically coincident (e.g., nodes on opposite faces of a crack).

The input to the algorithm is a triangular surface mesh, which describes the domain to be meshed. This mesh is obtained from the surface meshes on the boundary of a 3D region to be meshed. The algorithm steps are as follows:

- A two-pass advancing front procedure is applied to generate elements. In the first pass, elements are generated based on geometric criteria, producing well-shaped elements. The background octree presented in Sec. 4 is used to control the
sizes of the elements and the position of the interior nodes. The octree determines an ideal position for an optimal node to form a new element. This ideal position defines a search region where an optimal node for the new element may be located. This region is a sector of a sphere whose center is the ideal position and whose radius is proportional to the octree cell size. If one or more existing nodes are inside this region, they are ranked based on a solid angle criterion, in order to get the best node for the new element. However, if no existing node is found, a new node is inserted at the ideal position and an element is generated using this node. In the second pass, elements are generated based only on the criterion that they have valid topology. Here, any existing node that forms a valid new element can be used, regardless of whether it is close to the ideal position or not. However, the same quality criterion is used, and the node that forms the best solid angle is chosen for the generation of the new element.

- If the advancing front procedure cannot progress, a backtracking strategy [30] is employed to delete some elements, and the procedure is restarted. It consists basically of backtracking a few steps in the mesh generation and deleting faces that hinder the front from converging. This creates better regions where valid elements can be then generated.

It is possible that the process of finding better regions may fail, for instance, if faces to be removed are part of the original boundary. When this occurs, other elements are deleted instead and the procedure is restarted. If a mesh still cannot be generated for this region, the algorithm fails and terminates. In principle, it is possible to create a boundary input mesh that forces the failure of the volume mesh generation. Such failure, however, has not yet been observed in “noncontrived” input, i.e., in any realistic input boundary meshes, such as the ones showed in the examples and many more tested so far.

- Once a valid mesh is created, the quality of the element shapes is improved by using the standard Laplacian smoothing technique and locally deleting poorly shaped elements and those adjacent to them. The boundary contraction is then restarted.

After the generation of volumetric elements in all regions of the model as exposed above, a new error analysis is performed to assess the quality of the results. If convergence is not obtained, the whole adaptive process is repeated as described in the previous sections. Section 6 provides some examples of the proposed adaptive refinement process.

6 Examples

This section presents examples of adaptive 3D finite element meshes that were generated using the proposed adaptive methodology. The main objectives of this section are to (1) analyze the visual aspect of the meshes generated, comparing the results with and without adaptivity; (2) compare the results of the convergence rates with results presented in the literature; and (3) comment on the obtained results. It is important to emphasize that this paper does not aim to compute the performance of mesh generators (surface and volume) or assess the quality of the elements generated because these tasks were covered in previous works [27,28].

The adaptive strategy proposed in this paper results from the application of unstructured mesh generation techniques in surfaces and regions, combined with numerical errors associated with discretization. Numerical error estimators are implemented based on procedures developed for two-dimensional models [31] extended to three dimensions. These error estimators are supported by error estimation techniques widely adopted in the literature called superconvergent path recovery technique (SPR) [22,23] and recovery by equilibrium in patches (REP) [32]. The numerical error estimators were implemented in a finite element numerical analysis program [33]. It is worth stressing that these error estimation techniques can be easily and directly replaced by any other technique that is more recent or efficient, as this is supported by the object-oriented organization of the analysis program. In the following examples, SPR is employed for error estimation.

Fig. 11 Example 1: Short cantilever under end shear

Fig. 12 Example 2: L-shaped domain under horizontal uniform force (face y = 0)

Fig. 13 Example 3: Biaxial bending of a column footing
Figures 11–15 show five model examples to be tested employing the proposed adaptive strategy. The first three examples were obtained from the literature [34], and the adaptive process was applied by inserting additional nodes at the midpoint of the longest or quasi-longest line segment of the mesh, which bisects the original edges and generates new elements. In these examples, the adaptive refinement was carried out until the points were sufficient to allow comparison with reference results. The last two examples were chosen from real engineering problems, and the adaptive refinement was carried out until the relative error (3%) was reached. The model examples are described below:

1. Short cantilever under end shear. A short cantilever beam is loaded in one extremity under uniform shear forces, while the other extremity is free. Due to symmetry, only half of the model is created, and the displacement in the $x$ direction along the face OHGA is set to zero.

2. L-shaped domain under tension. An L-shaped domain is loaded under tension at the left extremity and free at the right extremity. Because of the distribution of internal stress, this model presents a singularity when it creates the L-form. Again, only half of the model is created due to symmetry, and the displacement in the face $x = 0$ is set to zero in the $x$ direction.

3. Biaxial bending of a column footing. This example is a column footing loaded at the top under uniform shear, face $z = 3$, and near the base under uniform pressure, face $x = 1.5$. All displacements of the footing face $z = 0$ are set to zero.

4. Three-dimensional frame. This example is composed of three regions with different materials, named regions 1, 2, and 3. In all of them, the cross section is $1 \times 1$. In region 3, the model is loaded under uniform tension in the $x$ direction in the face $x = 6$. The displacements are fixed at one extremity of region 1 in the face $y = 0$. This model covers a combination of external loading of tension (all regions), bending (regions 1 and 2), and torsion (region 1).

5. Bike suspension rocker [35]. This example is composed of four cylinders that are connected by a central body. Uniformly distributed forces, of unitary intensity in the $y$ direction are applied to the internal faces of the two top cylinders. Displacement restrictions ($u$, $v$, $w$) are applied to the internal face of the lower cylinder.

Figures 16–20 show the meshes generated by the adaptive refinement, considering quadratic tetrahedral elements. Similarly to the reference models Figs. 16–18, the meshes refine regions with singularities and regions with a high stress gradient. In example 1 (Fig. 16), the mesh is refined along lines AB, BC, and CO, with singularities. In example 2 (Fig. 17), the mesh is refined along the re-entrant corner line (AB), where the singularities are located, and coarsened on the face $y = 1$, where the constant tension stress is dominant. In example 3 (Fig. 18), the mesh is refined mostly at the junction between the column and the footing (lines CD, DF, EF, and CE), where singularities are located. Other refinement locations are the top of the column under uniform shear and near the base under uniform pressure, creating a singularity along line AB. In example 4 (Fig. 19), the adaptive refinement presents an important characteristic of the proposed methodology: to coarsen locations of low gradient stress. In region 3, the mesh is refined in the first step and coarsened in the second (final) step because the constant tension stress is dominant. In this example, only one background octree was used to hold the new sizes of the elements, another feature of the proposed methodology. The final mesh is coherent with the FE analysis because the mesh is more refined in region 1, where complex stress fields are present. In example 5 (Fig. 20), the mesh is refined in the intermediate cylinder and only part of the lower cylinder, where there are concentrations of stress. This example also demonstrates the importance of considering the curvatures of the supporting surface in the adaptive refinement, which is another characteristic of the present methodology. Figure 21 shows a detail of the mesh.
refinement, where part (on the bottom) of the lower cylinder is refined based on curvature.

Table 1 shows mesh and convergence information for all examples. “Ex.-Step” means example and current step of adaptivity, $\eta_{rel}$ is the estimated relative percentage error, and $\xi_{avg}$ and $\xi_{std}$ are the estimated and standard deviation error ratio of the finite element model, respectively. Note that all convergence information decreases to a minimum error.

Figure 22 shows the results of convergence rates compared to those of Ref. [9]. All results obtained herein converge faster to a small relative error. For instance, in example 1, the reference paper required three steps to achieve a relative error close to 1%, while the present work needed only two steps with adaptive refinement. The difference between these results is due to the fact that the present work employs a new global generation of tetrahedral elements in all regions, allowing the new elements to have greater shape and size flexibility, whereas using a local refining approach, as the reference work did, the shape and size of the new elements depend on the old mesh. All examples presented herein achieved a relative error close to $2\sim3\%$ with only two adaptive refinements.
To estimate the performance of the proposed methodology, Fig. 23 shows the elapsed processing times of a 3D mesh generator for the examples. Here, only the volumetric refinement step is included because this is the step where it takes longer to generate the final finite element model. These processing times were measured on a Pentium Core i7-1.73 GHz PC with 6 GB of RAM, under Windows 7 operating system. Note that the graph is in log–log scale. Using this graph, it is possible to estimate the time required for $1 \times 10^6$ elements, for example, which would

Table 1  Mesh and convergence information for examples

<table>
<thead>
<tr>
<th>Ex.–Step</th>
<th># of nodes</th>
<th># of elements</th>
<th>$\eta_{rel}$</th>
<th>$z_{avg}$</th>
<th>$z_{dev}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–0</td>
<td>463</td>
<td>268</td>
<td>6.55</td>
<td>0.946</td>
<td>0.908</td>
</tr>
<tr>
<td>1–1</td>
<td>4781</td>
<td>2785</td>
<td>2.09</td>
<td>0.197</td>
<td>0.369</td>
</tr>
<tr>
<td>1–2</td>
<td>10,479</td>
<td>6,379</td>
<td>1.29</td>
<td>0.140</td>
<td>0.216</td>
</tr>
<tr>
<td>1–3</td>
<td>19,994</td>
<td>12,376</td>
<td>0.85</td>
<td>0.105</td>
<td>0.134</td>
</tr>
<tr>
<td>1–4</td>
<td>23,835</td>
<td>14,678</td>
<td>0.74</td>
<td>0.099</td>
<td>0.110</td>
</tr>
<tr>
<td>2–0</td>
<td>268</td>
<td>137</td>
<td>7.97</td>
<td>1.170</td>
<td>1.387</td>
</tr>
<tr>
<td>2–1</td>
<td>5760</td>
<td>3721</td>
<td>2.86</td>
<td>0.217</td>
<td>0.599</td>
</tr>
<tr>
<td>2–2</td>
<td>12,832</td>
<td>8525</td>
<td>1.45</td>
<td>0.155</td>
<td>0.292</td>
</tr>
<tr>
<td>2–3</td>
<td>29,251</td>
<td>19,815</td>
<td>0.86</td>
<td>0.115</td>
<td>0.208</td>
</tr>
<tr>
<td>3–0</td>
<td>1302</td>
<td>721</td>
<td>19.56</td>
<td>2.018</td>
<td>3.355</td>
</tr>
<tr>
<td>3–1</td>
<td>16,672</td>
<td>10,682</td>
<td>6.82</td>
<td>0.710</td>
<td>1.164</td>
</tr>
<tr>
<td>3–2</td>
<td>96,537</td>
<td>65,988</td>
<td>3.21</td>
<td>0.320</td>
<td>0.557</td>
</tr>
<tr>
<td>4–0</td>
<td>229</td>
<td>112</td>
<td>13.71</td>
<td>2.102</td>
<td>1.768</td>
</tr>
<tr>
<td>4–1</td>
<td>5419</td>
<td>3004</td>
<td>7.40</td>
<td>0.855</td>
<td>1.208</td>
</tr>
<tr>
<td>4–2</td>
<td>57,300</td>
<td>36,996</td>
<td>1.35</td>
<td>0.177</td>
<td>0.849</td>
</tr>
<tr>
<td>5–0</td>
<td>10,303</td>
<td>6223</td>
<td>17.14</td>
<td>1.607</td>
<td>3.027</td>
</tr>
<tr>
<td>5–1</td>
<td>273,633</td>
<td>186,238</td>
<td>3.97</td>
<td>0.307</td>
<td>0.733</td>
</tr>
</tbody>
</table>

Fig. 21  Detail of mesh refinement for example 5

Fig. 22 Comparison of convergence rate

(a)  

(b)  

(c)  

Fig. 23  Comparison of convergence rate
A computational complexity of $O(N \log^p N)$ has been reported in the literature for advancing front techniques, $N$ being the number of generated elements [36–38]. It has also been reported that a performance of $O(N^p \log N)$, with $p$ slightly larger than one, is a more realistic measure [39,40]. In the present work, the complexity of the algorithm is reported as $O(CN^p \log N)$, where $C \approx 0.000269 - 0.0000629$ and $p \approx 1.2$ [28].

Other well-known refinement and mesh improvement algorithms from the literature achieved running times on the same range of the proposed algorithm or worse (for some of them). Although some of them are based on different approaches, such as Delaunay, and they were tested in different models (many times more simple than the ones presented in this work), they were included here to give an idea of running times for models of similar sizes. Gosselin and Ollivier-Gooch [41] developed a fast algorithm that consisted basically of creating a mesh from a point cloud, then refining and improving the mesh. The refinement phase for a mesh of the ridged torus using $R = 2$ and $G = 10$ yielded around 330,000 tetrahedra and required 16.3 s, therefore, inserting slightly less than 4000 vertices every second. The same insertion rates were observed for the other meshes presented in their work. When considering the mesh improvement phase, however, the insertion rates dropped considerably, but even then they were able to mesh a model made of roughly 25,000 tetrahedra connecting random vertices placed inside a cube in 9.1 s, using their mesh improvement algorithm. In the proposed methodology, the generation of the same number of tetrahedra, for more complex models, took around the same time (Fig. 23). The work of Klingner and Shewchuk [42], on the other hand, attains its best possible results at a very high computational cost [41]. For the same model, Klingner and Shewchuk’s implementation took between 1430 and 4658 s [41]. Another important work by Alliez et al. [43] produced meshes with remarkably high average quality and reasonable number of elements (275,000 elements for one of their models) in 4 min [41]. Extrapolation from these timings indicates that they can generate a mesh containing about $1 \times 10^6$ tetrahedra in around 15 min, which is the same range of the proposed algorithm (Fig. 23).

7 Conclusion

This paper described a methodology for adaptive generation of three-dimensional finite element meshes, using geometric modeling with multiregions and parametric surfaces. The mesh adaptive process involves three steps: (1) FE analysis with error estimation, (2) construction of a structure to store the new sizes of the FE, and (3) refinement of the FE model. The approach adopted is the global refinement of the whole model in each adaptive refinement, using a background octree structure. After the construction of the octree, the new model is geometrically rediscretized employing a hierarchical curve, surface, and volume refinement.

Some examples have demonstrated important characteristics of the proposed methodology, namely:

- the ability to refine and coarsen in regions of high and low response gradients
- the use of only one background octree for all regions of the model, allowing a smooth transition between regions and elements
- the hierarchical refinement of curves, surfaces, and volumes
- the consideration of curvatures of the supporting curves and surfaces in the adaptive refinement

In addition, comparing the results obtained by the present work with the ones in the referred literature, the former converges faster to a lower relative error because the 3D mesh generator used in this work has more freedom to create new elements based on desired element sizes. This characteristic generates a desired mesh with the application of only a few steps of the adaptive refinement. In contrast, using a local refinement strategy, (local) element manipulations restrict the shape quality of new elements.

Obviously, the current 3D mesh generation takes more time to create new elements because the whole FE model must be created at each step. In this work, most of the meshes were generated in less than 1 min of clock time. However, in models that require a large number of elements ($1 \times 10^6$ or more), time consumption can increases exponentially, as shown in Fig. 23. While in many problems, this is not an issue, it can be a limitation of the current approach that can be solved in two manners: (1) decomposing the domains into subdomains and applying the mesh generator to each subdomain, or (2) using a parallel 3D mesh generator. The latter option is our future work and is currently under development.

Acknowledgment

The authors would like to thank the National Council for Scientific and Technological Development (CNPq), the CAPES Foundation of the Ministry of Education of Brazil, the Computer Graphics Technology Group (Tecgraf), and the Pontifical Catholic University of Rio de Janeiro (PUC-Rio) for the financial support and for providing the necessary space and resources used during the development of this work. We thank ANP, Petrobras, and the Laboratory of Scientific Computing and Visualization (LCCV) of the Federal University of Alagoas for granting access to the GradoBR/UFAL cluster computing resources of Rede Galileu.

Nomenclature

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\xi$</td>
<td>discretization error ratio</td>
</tr>
<tr>
<td>$\xi_{avg}$</td>
<td>average error ratio of finite element model</td>
</tr>
<tr>
<td>$\xi_{dev}$</td>
<td>standard deviation error ratio of finite element model</td>
</tr>
<tr>
<td>$\nu$</td>
<td>Poisson’s coefficient of material</td>
</tr>
<tr>
<td>$\theta_{ij}$</td>
<td>estimated angle between curve tangents between vertices $i$ and $j$</td>
</tr>
<tr>
<td>$\theta_{max}$</td>
<td>maximum allowed curvature angle of curves</td>
</tr>
<tr>
<td>$\eta$</td>
<td>relative percentage error in the energy norm</td>
</tr>
<tr>
<td>$\eta_{rel}$</td>
<td>estimated relative percentage error</td>
</tr>
<tr>
<td>$\eta_{max}$</td>
<td>maximum allowable relative percentage error</td>
</tr>
<tr>
<td>$\delta_{U}(U,V)$</td>
<td>gradient vector in direction $U$ at a parametric point $(U,V)$ on a surface</td>
</tr>
<tr>
<td>$\delta_{V}(U,V)$</td>
<td>gradient vector in direction $V$ at a parametric point $(U,V)$ on a surface</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>exact stress fields</td>
</tr>
<tr>
<td>$\hat{\sigma}$</td>
<td>finite element stress fields</td>
</tr>
<tr>
<td>$\tilde{\sigma}$</td>
<td>recovery stress fields</td>
</tr>
<tr>
<td>$\delta_{U}(V,U)$</td>
<td>gradient vectors, direction $V$, at a parametric point $(U,V)$ on the surface</td>
</tr>
</tbody>
</table>
References


