Robust h-adaptive meshing strategy considering exact arbitrary CAD geometries in a Cartesian grid framework

Onofre Marco[a,], Juan José Ródenas[a], José Manuel Navarro-Jiménez[a], Manuel Tur[a]

[a]*Centro de Investigación en Ingeniería Mecánica (CIIM), Universitat Politècnica de València, Valencia 46022, Spain*

Abstract

Geometry plays a key role in contact and shape optimization problems in which the accurate representation of the exact geometry and the use of adaptive analysis techniques are crucial to obtaining accurate computationally-efficient Finite Element (FE) simulations. We propose a novel algorithm to generate 3D h-adaptive meshes for an Immersed Boundary Method (IBM) based on Cartesian grids and the so-called NEFEM (NURBS-Enhanced FE Method) integration techniques. To increase the accuracy of the results at the minimum computational cost we seek to keep the efficient Cartesian structure of the mesh during the whole analysis process while considering the exact boundary representation of domains given by NURBS or T-Splines.

Within the framework of Cartesian grids, the two significant contributions of this paper are: a) the methodology used for the mesh-geometry intersection, which represents a considerable challenge due to their independence; and b) the robust procedure used to generate the integration subdomains that exactly represent the CAD model. The numerical examples given show the proper convergence of the method, its capacity to mesh complex 3D geometries and that Cartesian grid-based IBM can be considered a robust and reliable tool in terms of accuracy and computational cost.

Keywords: Cartesian grids, h-refinement, NURBS, NEFEM

Email addresses: `onmaral@upvnet.upv.es` (Onofre Marco), `jjrodena@mcm.upv.es` (Juan José Ródenas), `jonaji@upvnet.upv.es` (José Manuel Navarro-Jiménez), `manuel.tur@mcm.upv.es` (Manuel Tur)

1. Introduction

It has recently become clear that a major drawback to the rapid structural analysis of geometrically elaborated 3D domains using Finite Element Analysis (FEA) is the time allotted to creating an appropriate finite element mesh. Even after the development of sophisticated mesh generators, a significant amount of skilled human resources is required to create good quality finite element meshes for the geometrically complex models required for the solution of common industrial problems.

The aim of adaptive mesh generation and automatic error control in FEA is to eliminate the need for manual re-meshing and re-running design simulations to check the numerical accuracy. In ideal circumstances, the user should only input the component model and a coarse finite element mesh. The software should then autonomously and adaptively reduce the element size where required, reducing the error in the solution fields to a predetermined value.

Adaptive methods of finite element simulations were first proposed in the late 70's[1, 2]. The most common criterion in general engineering use is that of prescribing a limit for a global magnitude, such as the error computed in the energy norm, though it is possible to define magnitudes of interest to evaluate the goodness of the evaluated meshes[3, 4, 5].

The procedures for the refinement of finite element meshes fall mostly into two categories:

1. The h-refinement, in which the element type is maintained but the elements are changed in size. In some locations of the mesh the element sizes are made smaller, or larger (not very common), where needed to provide maximum computational economy in reaching the desired solution.
2. The p-refinement, in which the element size is kept constant and the order of the polynomial, used in its definition is increased where necessary, generally by using hierarchical shape functions[6, 7].

There exists a third category, the hp-refinement [8, 9], which consists of simultaneously adapting the size of the elements and their approximation degree.

In this contribution we will present an h-adaptive refinement strategy based only on the size of the element keeping the polynomial order of the interpolation constant.

Decades after the development of functional meshing techniques[10, 11, 12], mesh generation still has to evolve in order to minimize the design cycle time because real industrial applications are, in general, geometrically complex and traditionally require a skilled workforce to generate an analysis-suitable finite element mesh.

One alternative to reduce the meshing burden, related to the proposals in this paper, is to use mesh generators based on simple discretizations such as *octrees* [13, 14, 15]. In octree-based mesh generators [12, 16, 17] an embedding cube-shaped domain is created and meshed following a Cartesian hierarchy through the mesh generation process for efficiency. After adapting the octree mesh to the geometry and splitting the cut cells into tetrahedrons to capture the boundary of the model, the octree is broken up into a valid body-fitted mesh and then smoothing techniques are used achieve good quality finite elements.

It is apparent that mesh generation could be greatly simplified by using implicit meshing approaches in which (as in octree techniques) the geometrically complex domain is embedded into a geometrically simpler domain whose meshing is simple if not trivial. As opposed to octree techniques, in the approach described here the non-conforming FE mesh is not modified to fit the boundary. Instead, the matching between geometry and mesh is done during the evaluation of element integrals, which are defined only by the part of the elements cut by the boundary that lies within the domain. Many methods are described in the literature in which the geometrically complex domain is embedded into a geometrically simpler domain. Among many other names used to describe these FE techniques in which the mesh does not match the domain's geometry, there is the Immersed Boundary Method (IBM) [18], the Immersed Finite Element Method (IFEM) [19] or the Finite Cell Method (FCM)[20, 21, 22]. Immersed boundary methods, often referred to as embedded methods, have been studied by a number of authors for very different problems such as, for example, shape optimization [23, 24] or bio-mechanics [25, 26, 27]. Most of these techniques rely on an integration submesh in the elements cut by the boundary to perform the body-fitted numerical integration appearing in the weak formulation.

Implementing the Finite Element Method in combination with the embedded-domain concept offers a powerful alternative due to the potential benefits: virtual automatic domain discretization, suitable for creating hierarchical data structures for simple data transfer and re-use of calculations, ability to easily create adapted domain discretizations, a natural platform for efficient structural shape optimization processes, multigrid and multiscale analyses, etc. However, there are also tradeoffs with this approach related to the fact that the boundary of the domain does not necessarily coincide with the element faces. For example, there are difficulties in accurately integrating the weak form of the governing equations over the elements intersected by the boundary. There are also difficulties in imposing essential boundary conditions as the nodes do not necessarily lie on the Dirichlet boundary and the direct enforcement of the essential boundary conditions is in general not possible.

As an efficient solution for these drawbacks, we used a methodology based on the use of Cartesian grids independent of the geometry. This methodology, known as cgFEM[28, 29], was implemented in a computer code for the structural analysis of 3D components considering uniform meshes. The first 3D version of this methodology, known as FEAVox[30] was described in a previous paper. The aspect that distinguishes FEAVox from other immersed boundary approaches is that it is able to consider the exact CAD representation of the boundary of the domain, given by NURBS[31, 32] or T-Splines[33], in the evaluation of volume integrals. To perform the numerical integration, instead of simplifying the embedded geometry, for instance using triangular facets for its definition, FEAVox includes novel techniques to perform the exact integration (up to the accuracy of the quadrature rule) over the true computational domain. In particular, these integration techniques are the techniques considered by the NURBS-Enhanced Finite Element Method (NEFEM) [34, 35].

The accurate evaluation of integrals in elements cut by the boundary, it is necessary to maintain the optimal convergence rate of the error of the FE solution. This is, therefore, an active area of research. In fact, several methodologies have recently emerged to perform high-order integration in embedded methods, such as the so-called 'smart octrees' tailored to Finite Cell approaches[36] or techniques in which the geometry is defined implicitly by level sets[37]. We used the NURBS-Enhanced integration techniques because their consistency considering the exact geometric description[38] is of major importance when dealing with CAD models in applications such as shape optimization or contact between bodies.

This contribution will show how the capabilities of the cgFEM methodology have been improved by developing h-adaptive analysis techniques. These techniques have been successfully implemented in FEAVox in order to handle complicated CAD models without renouncing the traditional properties of embedded methods as well as developing a robust enhanced procedure for geometry-mesh intersection.

The paper is organized as follows: a brief review of the basic features of the cgFEM methodology is given in Section 2. Section 3 explains how the mesh-geometry intersection problem is solved. Section 4 describes an extended scheme to efficiently integrate elements intersected by the boundary. Section 5 gives details of the refinement strategies. Section 6 contains numerical results showing the behavior of the proposed technique. Finally, the conclusions are given in Section 7. The derivation of the h-adaptive refinement criterion for 3D meshes is given in Appendix A.

2. Cartesian grids with exact representation of the geometry: FEAVox

The present work is the logical continuation of [30], which introduced the new cgFEM methodology implemented in an FE code, called FEAVox, for the analysis of structural 3D components. Its main novelty was its ability to perform accurate numerical integration in non-conforming meshes independent of the geometry. A brief review of cgFEM and its features is given here as a background to the present paper.

The foundations of mesh generation in cgFEM consists of defining an embedding domain Ω such that a bounded domain $\Omega_{\texttt{Phys}}$ fulfills $\Omega_{\texttt{Phys}} \subset \Omega$. Let us assume that the embedding domain is a cube, although rectangular cuboids could also be considered. This means Ω is much easier to mesh than the domain of interest $\Omega_{\texttt{Phys}}$. Figure 1 gives an example of the different domains defined. Figure 1b only gives the elements of the embedding domain interacting with $\Omega_{\texttt{Phys}}$ denoted by $\Omega_{\texttt{Approx}}$ and Figure 1c shows a representation of the submesh used only for integration purposes.

The original version of FEAVox considered a sequence of uniformly refined Cartesian meshes to mesh the Ω, where the different levels of the Cartesian meshes were connected by predefined hierarchical relationships. The term Cartesian grid set, denoted by $\{\mathcal{Q}_h^i\}_{i=1,\dots,m}$, is used to define the sequence of m meshes utilized to discretize the embedding 3D domain Ω. For each level i of refinement, the embedding domain Ω is partitioned into $\texttt{n}_{\texttt{el}}^i$ disjoint cubes of uniform size, where $\texttt{n}_{\texttt{el}}^{i+1} = 8\texttt{n}_{\texttt{el}}^i$. While in a uniform refinement process this operation was carried out for every element in the mesh, in our h-adaptive approach, the subdivision step will be guided either by local geometrical parameters or a discretization error-based criterion, so we could end up with elements of different sizes in the same mesh.

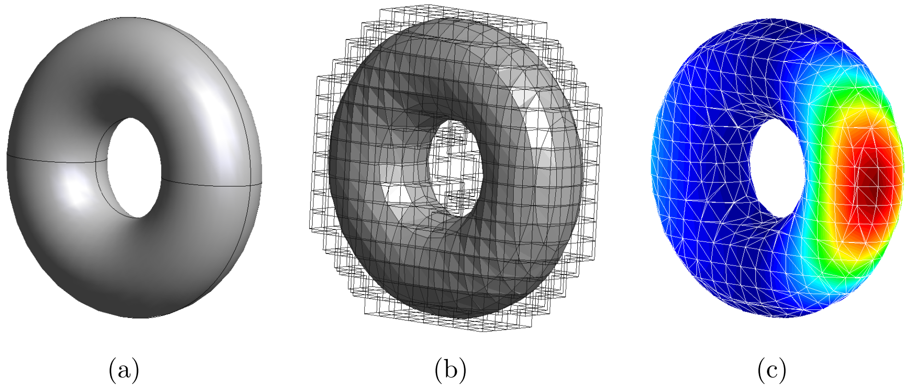


Figure 1: Immersed Boundary Method environment. (a) Torus geometry, $\Omega_{\texttt{Phys}}$. (b) Approximation mesh, $\Omega_{\texttt{Approx}}$. (c) Integration mesh, $\Omega^h_{\texttt{Phys}}$.

It is worth noting that the hierarchical relationships considered in the data structure provide automatisms for mesh refinement, thus positively affecting the efficiency of the FE implementation. Nodal coordinates, mesh topology, hierarchical relationships, neighborhood patterns, and other geometric or topological information can be obtained algorithmically. This information is therefore not stored in the memory, making the proposed algorithm more efficient, not only in terms of computational cost but also in terms of memory requirements.

During the creation of the FE analysis mesh used to solve the boundary value problem we can classify the elements of the Cartesian grid into three groups: boundary, internal and external elements. Let Γ be the boundary of $\Omega_{\texttt{Phys}}$ and Ω^e the domain of every element conforming the embedding domain Ω.

We define $\Omega_{\texttt{I}}$ as the set of elements fully contained in the model domain, $\Omega^e \subset \Omega_{\texttt{Phys}}$ (green elements in Figure 2). We also define $\Omega_{\texttt{B}}$ as the set of elements such that $\Omega^e \cap \Gamma \neq \emptyset$. Within these elements we will use a submesh to take into account that only a portion of these elements needs to be integrated, i.e. the portion of these elements that lies inside the physical domain, namely $\Omega_{\texttt{B}}^{\texttt{Phys}} = \Omega_{\texttt{B}} \cap \Omega_{\texttt{Phys}}$ (blue triangles in Figure 2).

The internal elements are standard FE elements and the affinity with respect to the reference element is exploited in order to avoid the computational cost of creating their element matrices. For the elements cut by the

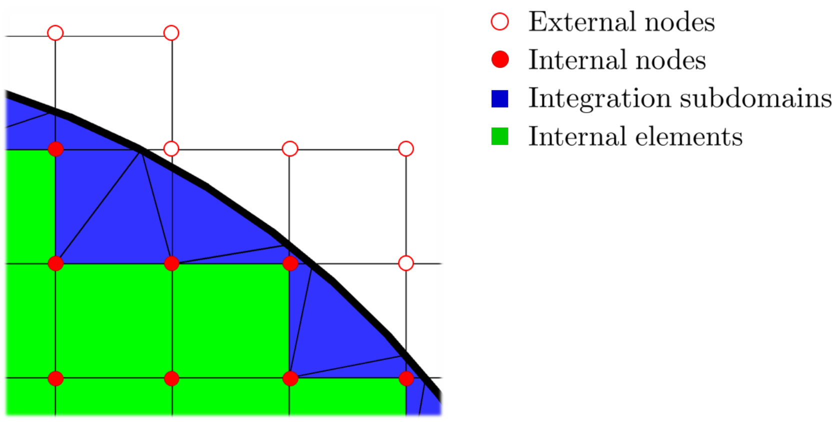


Figure 2: Section of a 3D Cartesian grid showing the different types of elements and nodes.

boundary, due to the independence between the embedded geometry and the mesh, as it is necessary to determine the relative position of the elements with respect to the physical boundary, specific strategies are proposed to find the intersections with the boundary and to perform the numerical integration. Since the intersection process is a key step in our algorithm, Section 3 will be devoted to giving a detailed explanation of its main aspects.

Regarding the integration of intersected elements in cgFEM, we proposed a strategy to perform the integration over $\Omega_{\mathtt{B}}^{\mathtt{Phys}}$ employing a tetrahedralization of this region in each boundary element that incorporates the exact boundary representation of $\Omega_{\mathtt{Phys}}$. Numerical integration over the region $\Omega_{\mathtt{B}}^{\mathtt{phys}}$ is then accomplished by integrating over each subdomain of the tetrahedralization. The strategy proposed in the NURBS-Enhanced Finite Element Method (NEFEM)[34, 35] is adopted to perform the integration over the subdomains. As has been demonstrated in the previous work, this approach can be successfully used in a Cartesian grid environment, giving it the ability to integrate the curved subdomains of domains parametrized by NURBS, T-Spline or other parametric representations.

In cgFEM, Dirichlet boundary conditions are imposed using stabilized Lagrange multipliers, or more precisely, the procedure chosen to impose the constraints follows the technique proposed in [39]. This method is suitable for h-refinement in the context of hierarchical Cartesian grids, where the problem is stabilized by a functional added to the initial formulation.

3. Geometry-mesh intersection

As a logical consequence of the independence between the analysis mesh and the geometrical model, the problem arises of discriminating the parts of the mesh inside and outside the model. The simplest approach is to find the intersections of the physical boundary with the edges of the Cartesian grid elements. This is usually a simple problem when using a model described by a tessellated boundary or a linear interpolation from implicit boundary representations, as for example level-set functions. However, when dealing with exact explicit representations, as in the present work, or high-order implicit boundary representations[37], more sophisticated boundary-tracking procedures are needed.

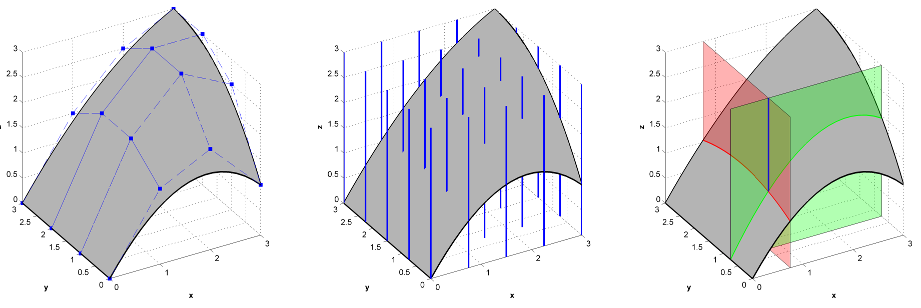
There are several methods available in the literature to evaluate the intersection between parametric surfaces and arbitrary rays. These are known as ray-tracing strategies and are widely used by the computer graphics community and the animation and videogames industries, whose need for better representation technologies fostered the rapid proliferation of these techniques.

The ray-surface intersection is generally calculated in one of two ways, according to the nature of the surface. If the surface is defined by a tessellation, the ray-tracing is performed on the resulting set of triangular surfaces, which is algebraically trivial. When using parametric surfaces, the curve-surface intersection is solved directly, usually by a numerical method. There are plenty of algorithms for ray-tracing parametric surfaces available in the literature[40, 41, 42, 43, 44].

Here we propose a robust algorithm for finding the intersections of a Cartesian grid and parametric surfaces. The algorithm includes the multivariate Newton method and incorporates criteria to minimize the disadvantage of requiring an initial guess, which must generally be close to the root itself. In this section, since we only need a basic version of the well known Newton's Method, we will focus on the details of how to make an accurate initial guess for the intersections.

The process will be illustrated with an example. Figure 3a contains an arbitrary parametric surface and its control points. Since we are using Cartesian grids, we need to intersect this surface with straight lines following directions X, Y or Z only. In Figure 3b a set of axes defined along Z are plotted that intersect the parametric surface.

In addition, we know that each one of the Cartesian axes will be defined by two Cartesian planes, as shown in Figure 3c, where it can be seen that



(a) (b) (c)

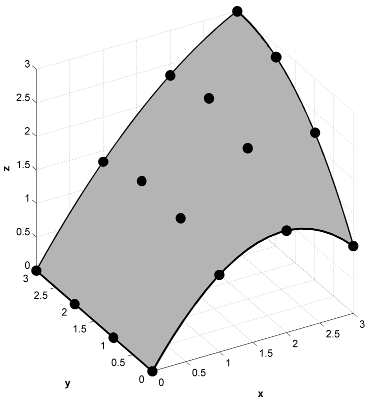
Figure 3: Example of surface and Cartesian axes. (a) Parametric surface. (b) Axes in direction Z. (c) Cartesian planes of an axis.

the intersection between a YZ-plane, defined by the coordinate x, and a XZ-plane, defined only by y, yields a Z-axis.

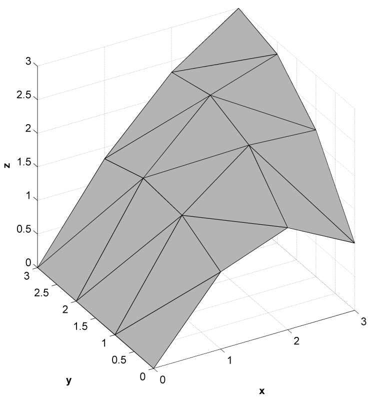
In order to make a good initial guess for every axis in the Cartesian grid we have to choose points that would be close enough to the actual intersections. To do this in an efficient way we will generate a triangulation of the surface by evaluating a properly defined set of points, Figures 4b and 4a respectively. The points and the subsequent triangulation are defined in the parametric space and then projected onto the physical space in which the intersecting planes are defined.

It is worth noting that if the triangulation is too coarse a Cartesian axis could intersect the same triangle several times (illustrated in [30]). If the triangle is defined in an area of the surface with curvature changes, considering the same initial guess for different roots will prevent the convergence of the Newton-Raphson algorithm to all the different roots. To avoid this situation, we recommend that the distance between the set of points evaluated on the surface be related to the distance between the Cartesian planes of the mesh, and thus related to the element size.

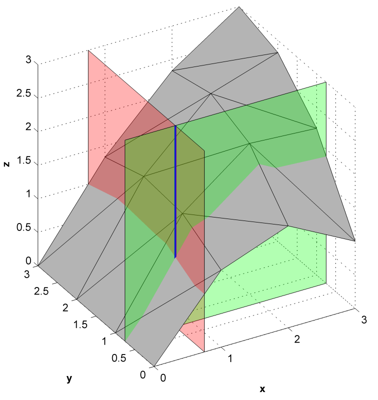
In order to do this, we first evaluate the physical bounding box of each surface. With this information, the bounding box of the embedding domain and the maximum refinement level allowed by the user, we estimate how many axes will be intersecting the surfaces. We set the number of points that define the uniform grid, with the vertices of the auxiliary triangulation, such as $N_P = 3 \cdot max\left\{N_A^x, N_A^y, N_A^z\right\}$, where N_A is the number of axis that intersect the bounding box of the surface in each Cartesian direction. This criterion has been successfully applied to different examples. In any case, the user, to be able to capture any kind of curvature changes, could tune the factor that multiplies N_A.



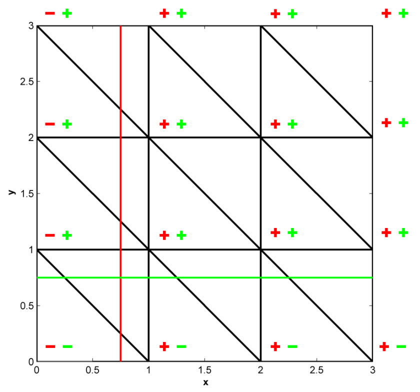
(a)



(b)



(c)



(d)

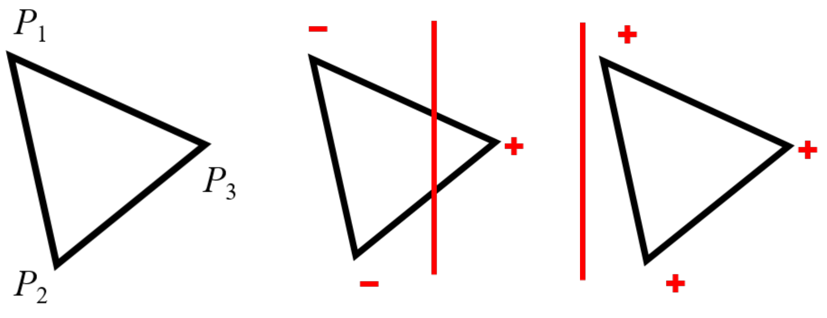
Figure 4: Procedure for the initial Newton-Raphson guess. (a) Arbitrary discretization of the surface. (b) Triangulation generated. (c) Triangulation and intersection planes. (d) 2D view of the level sets.

After obtaining the auxiliary triangulation we evaluate the level sets of the intersection planes (Figure 4c) with respect to the points on the parametric surface. In this way we will identify the position of the points with respect to the two planes by evaluating the sign of the distances in the physical space. This represents a trivial operation as it only requires comparing the global coordinate of each point of the triangulation with the coordinates that define the Cartesian planes. Figure 4d shows a view of the level sets calculated in the parametric space. The signs of the distances to the planes XZ and YZ are in red and green, respectively. Our strategy consists of finding triangles that are cut at the same time by the two planes defining the intersecting axis.

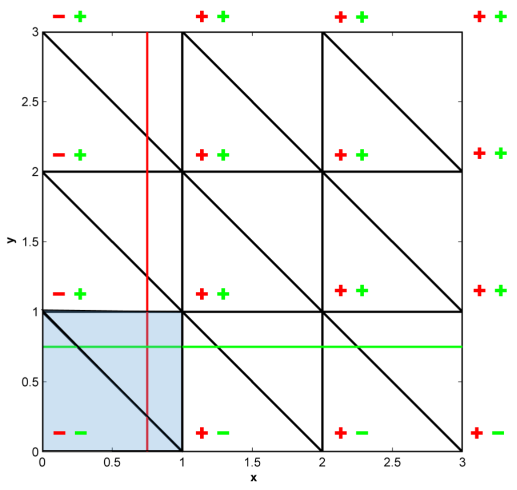
Now, if every triangle T_i of the triangulation is defined by the vertices $\{P_1, P_2, P_3\}$ where the coordinates of P_i are given by $\{P_i^x, P_i^y, P_i^z\}$ (Figure 5a), then we say that the triangle is cut by a plane when we can find vertices on both sides of the plane at the same time, i.e., there is a change in the sign of the vertices (Figure 5b). Otherwise, the triangle is not intersected when the signs of all vertices are the same (Figure 5c). As we have said, every axis in the mesh is defined by two planes, so if a triangle is intersected by both planes at the same time we will use it to make the initial guess of the axis in question. Using these criteria, the area where we have to make the initial guess for the axis for the case in Figure 4 can easily be identified (see Figure 5d).

It should be noted that the fact of not meeting the criteria does not necessarily mean the intersection does not exist. Indeed, if we use a linear triangulation to discretize an arbitrary parametric surface, it could happen that a plane intersects the triangles that had not been detected by the previous criteria. An example of this can be seen in Figure 6a, where the sign of the vertices indicate that the triangle is not intersected, but if we had considered the real definition of one edge of the triangle the intersection would have existed.

To identify the intersection we introduce a new criterion based on the distances of the vertices to the plane of intersection. Let C_i be the maximum length of the bounding volume defined by the vertices of the triangle T_i such that $C_i = max\left\{C_i^x, C_i^y, C_i^z\right\}$ where $C_i^x = max\left\{P_i^x\right\} - min\left\{P_i^x\right\}$ and C_i^y and C_i^z are defined in the same way. Then, d_1, d_2 and d_3 being the distances from the vertices to a plane (Figure 6b), we say that if $\left\{|d_1|, |d_2|, |d_3|\right\} < C_i$, the triangle is ambiguous because we cannot ensure the existence or non-existence of the intersection. To eliminate this ambiguity we subdivide the triangle and recalculate the criteria. This subdivision is recursively applied until the ambiguity is eliminated. An extreme case will be the existence of



(a) (b) (c)



(d)

Figure 5: Intersecting the Cartesian planes and the triangulation. (a) Element of the triangulation. (b) Cutting plane. (c) Non-cutting plane. (d) Target triangles for intersection.

tangent points as shown in Figure 7a. In this scenario, the subdivision will continue and the procedure will stop when the triangle size is small enough to assume the axis is tangent to the surface, Figure 7b. When a surface itself is coplanar to Cartesian axes, see Figure 7c, the intersection of those axes have to be circumvented to avoid the excessive subdivision due to the theoretical presence of infinite intersections. In order to do this, it is possible to check if a surface is defined along any Cartesian coordinate and if so, detecting any

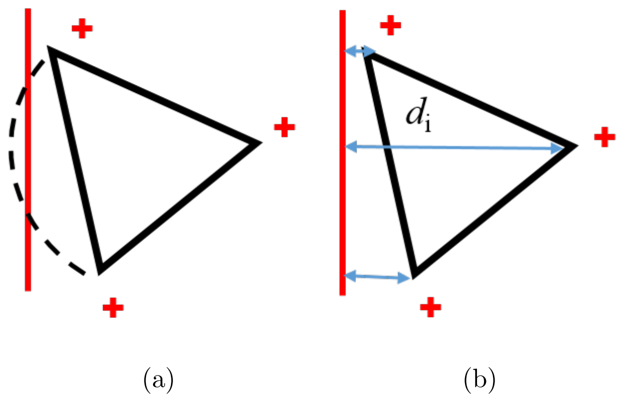


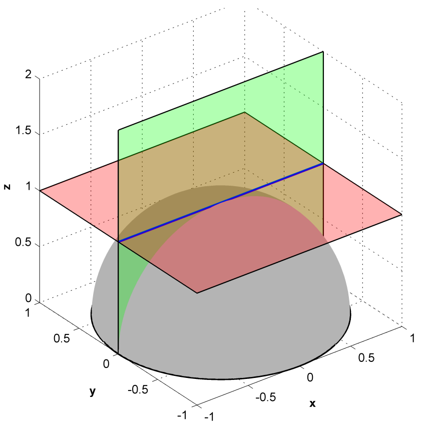
Figure 6: Ambiguous intersection between triangles and planes. (a) Case of intersection not detected. (b) Distances of points to plane.

axis contained in it is simple. The intersections evaluated on these surfaces will come from the axes normal to the tangent plane, as pictured in Figure 7d.

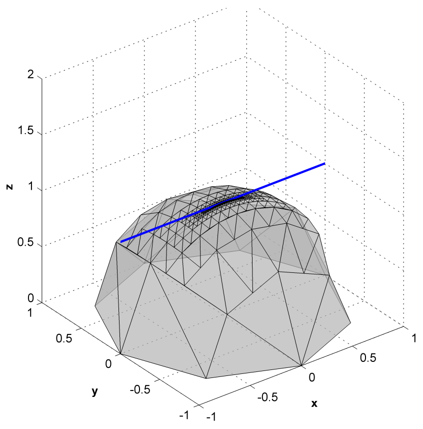
Once we have identified all the candidate triangles to be intersected by every axis in the mesh, we will choose their geometrical center in the parametric coordinates as the initial guess to evaluate the intersection. After obtaining these initial points we will compute the intersections using the Newton-Raphson method, as mentioned above. Regarding this iterative process, when dealing with badly parametrized surfaces, the derivatives could present rapid changes. The parametric space of a surface is defined, if normalized, as a quadrilateral with dimensions $[0,1] \times [0,1]$. Rapid changes in the derivatives during the Newton-Raphson procedure could yield in iteration parameters, $\{\xi_i, \eta_i\}$, outside the definition of the parametric space. To avoid this situation we force the parameters to be $0 \leq \{\xi_i, \eta_i\} \leq 1$, allowing to keep points that would be discarded in an intermediate stage of the process.

With the intersections evaluated, it is trivial to classify the nodes as internal or external by simply marching along the edges of the Cartesian grid and the classification of elements as internal, boundary or external, is automatically achieved by counting the number of internal and external nodes in each element.

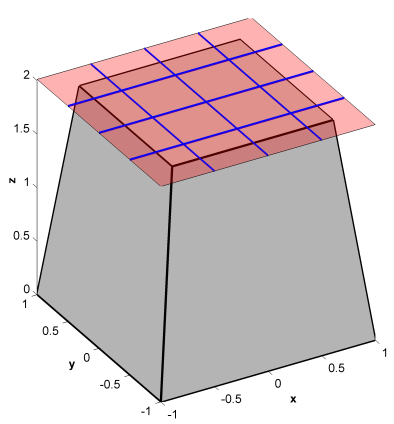
It is important to understand that this intersection step is the keystone to achieve overall robustness of the methodology. Both the generation of



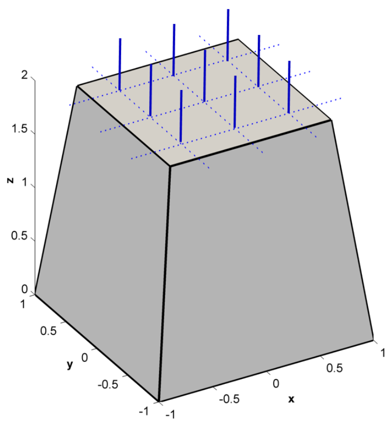
(a)



(b)



(c)



(d)

Figure 7: Treatment of tangent situations. (a) Axis tangent to a surface. (b) Resulting subdivided triangulation. (c) Plane tangent to a surface. (d) Intersection of normal axes only.

subdomains (see Section 4) and the geometrical refinement (see Section 5.1), rely in the assumption of the quality of the information obtained during this intersection step.

4. Integration patterns

The FEM requires the computation of integrals over the domain, $\Omega_{\texttt{Phys}}$, and over its surface, Γ. The numerical integration in the IBM requires special attention as the mesh is usually independent of the geometry of the physical domain.

As explained in [30], internal elements, $\Omega_{\texttt{I}}$, are standard finite elements in which the integration is performed using a tensor product of one-dimensional Gauss quadratures with the specified number of points in each direction. Nevertheless, the contribution of the boundary elements, $\Omega_{\texttt{B}}$, requires special attention as the integrals in these elements must be computed only over the portion of the element that lies inside the physical domain, namely $\Omega_{\texttt{B}}^{\texttt{Phys}} = \Omega_{\texttt{B}} \cap \Omega^{\texttt{Phys}}$ (see Figure 1c). In fact, the independent generation of the Cartesian grid with respect to the embedded geometry implies that the region of each element intersected by the mesh lying inside the computation domain, $\Omega_{\texttt{B}}^{\texttt{Phys}}$ can be extremely complex.

The approach proposed in [30] to perform the integration over $\Omega_{\texttt{B}}^{\texttt{Phys}}$ consists of employing a tetrahedralization of this region that incorporates the exact boundary representation of $\Omega_{\texttt{Phys}}$. This strategy was inspired on the Marching Cubes (MC) algorithm [45], which uses a set of templates for the intersection between surfaces and the edges of cubes to define a surface triangulation. Since a cube has 8 corners and each corner can have two states, there are $2^8 = 256$ possible types of intersection. By symmetry considerations, this can be reduced to 15 basic cases (14 if we remove the case with no intersections).

The idea is very simple; we start with the reference hexahedral element in Figure 8, in which we have identified the 8 nodes and the 12 edges where the intersections can appear. The algorithm will only allow one intersection point with an edge, so the edge numbers in Figure 8 can also be considered as possible intersection points. We extended the idea behind the MC algorithm to create templates that define tetrahedralizations of the domain contained in each boundary element. To represent these templates we will consider that the CAD surface intersects the element edges at their midside point, as shown in 8. Flat surface tetrahedrons will be considered to define the templates.

Figure 9a shows an example of an integration pattern with an internal node (red dot), 3 intersections (green squares) and 7 external nodes (blue dots). With this set of nodes and intersections we can generate a tetrahe-

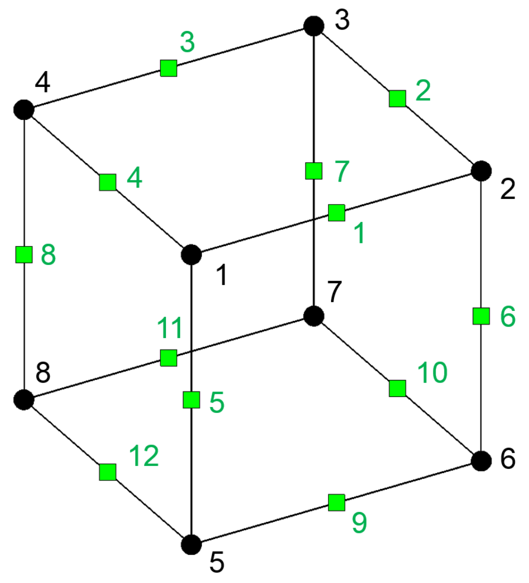


Figure 8: Numbering of nodes and edges in a reference element.

dralization, for example using the Delaunay tetrahedralization, to discretize the element into subdomains (Figure 9b). We can deduce that all the elements that present the same configuration could share the same pattern of tetrahedrons, so we only need to keep in the computers memory one set of tetrahedrons for every configuration and use it multiple times.

In the previous study [30], it was assumed that the edges of the elements are intersected, at most, once by the boundary of the physical domain. From this premise, we need only 7 out of 14 templates of the original MC algorithm (1, 2, 5, 8, 9, 11 and 14, see [45]). The seven patterns considered are depicted in Figure 10. In the figures we can see the set of tetrahedra used for each pattern and the corresponding nodal topologies. Colors identify internal and external subdomains (or different materials in the case of multi-material interfaces). The actual location of the intersections on the edges and curved-face tetrahedrons exactly defining the CAD surface will be considered during the integration process once the integration pattern has been determined. Since we have found the intersections between the geometrical model and the mesh, and thus the nodes that are internal or external to the body, we can identify the intersection pattern for every boundary element in our reference element.

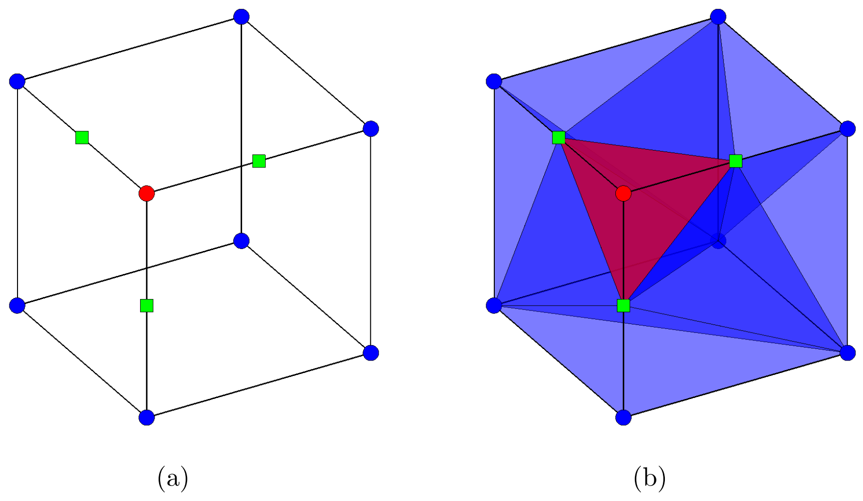
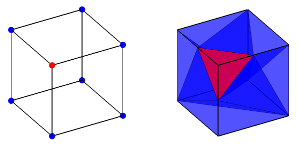


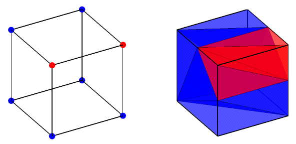
Figure 9: Integration pattern example. (a) Nodal and edge intersection topology. (b) Tetrahedralization.

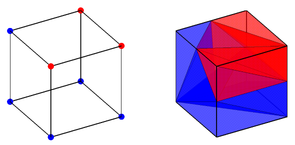
These patterns are valid when the elements are intersected by only one surface, but in problems defined by arbitrary geometries there will be elements intersected by several surfaces at the same time. A common situation is the existence of sharp features inside an element generated by the interfaces of connecting surfaces, see Figure 11a. The proposed method is to evaluate these elements individually, generating specific sets of tetrahedra, using for instance a Delaunay procedure, as in Figure 11b.

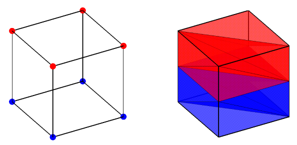
Following [34], integration subdomains with several faces on different surfaces are split into tetrahedrons with only one face on a parametric boundary. It is worth noting that subdivisions are only applied to design a numerical quadrature. Two examples are presented to illustrate the proposed strategy. The first example considers a tetrahedral element Ω_T with two faces on different surfaces ($P_1 - P_2 - P_4$ on Γ_A and $P_2 - P_3 - P_4$ on Γ_B) (see Figure 12). In this example, we will use the only edge not lying on the boundary, edge $P_1 - P_3$, to define its geometrical center P_E and generate two new subdomains with only one face on the boundary.

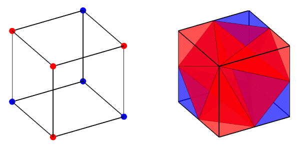
The second example considers an element Ω_T with three faces on different surfaces, as represented in Figure 13. In this case, the subdomain is split into three tetrahedrons using the geometric center P_F of the only face not lying on any boundary (face $P_1 - P_2 - P_4$). New subdomains are then defined as a

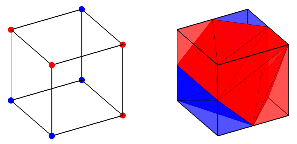

(a) Configuration 1.


(b) Configuration 2.


(c) Configuration 3.


(d) Configuration 4.


(e) Configuration 5.


(f) Configuration 6.

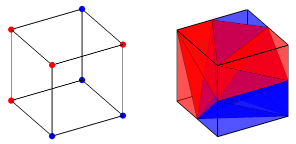
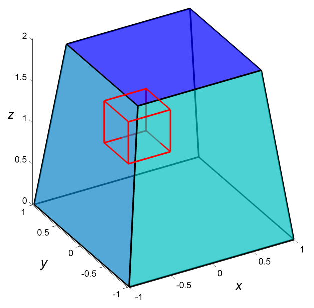
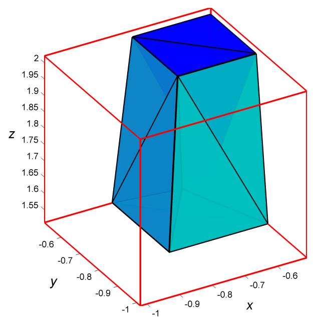

(g) Configuration 7.

Figure 10: Intersection patterns inspired on the MC algorithm. Nodal topology (left) and tetrahedralization (right).



(a)



(b)

Figure 11: Exception to the intersection patterns. (a) Element intersected by several patches of a trapezoidal prism. (b) Detail of the resulting tetrahedralization.

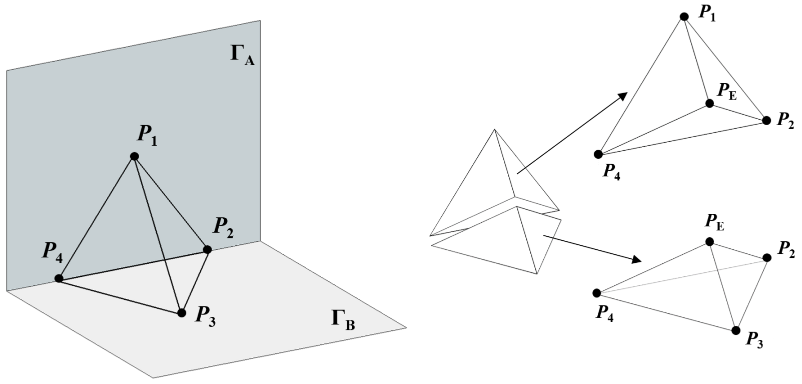


Figure 12: Subdivision of tetrahedrons with two faces on different parametric boundaries.

linear convex combination of P_F and original boundary faces of Ω_T, having at most one face on a parametric boundary.

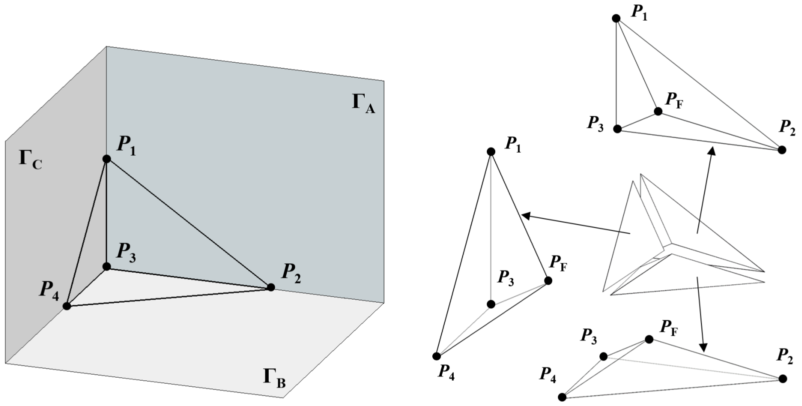


Figure 13: Subdivision of tetrahedrons with three faces on different parametric boundaries.

The evaluation of these elements has a higher computational cost than that of elements with standard patterns, however, the ratio of the amount of elements with configurations not represented by the standard patterns to the number of elements in the mesh is very low, in general. In addition, the strategy presented allows both the consideration of sharp features and a proper discretization, to integrate parametric boundaries through NURBS-Enhanced rationale in an immersed boundary environment.

Remark 1. *It is worth mentioning, that for other piecewise boundary definitions (surface triangulations for instance) FEAVox uses the linear version of the subdomains generated to approximate the boundary, as depicted in Figure 14a. In these cases, a proper h-adaptive strategy of the boundary would be necessary to improve geometrical accuracy. In addition, when it is important to properly capture the piecewise boundary representation, then it is possible to apply the procedure proposed in this contribution,treat all the different components of these surfaces as individual parametric surfaces, and generate a submesh, as shown in Figure 14b. Lastly, if the model is defined with high-order piecewise polynomials we could always use the NEFEM integration (Figure 14c). The last two options would yield excessive mesh refinements and a higher computational cost due to the large number of unions between surfaces, which implies the exclusive tetrahedralizations of more elements.*

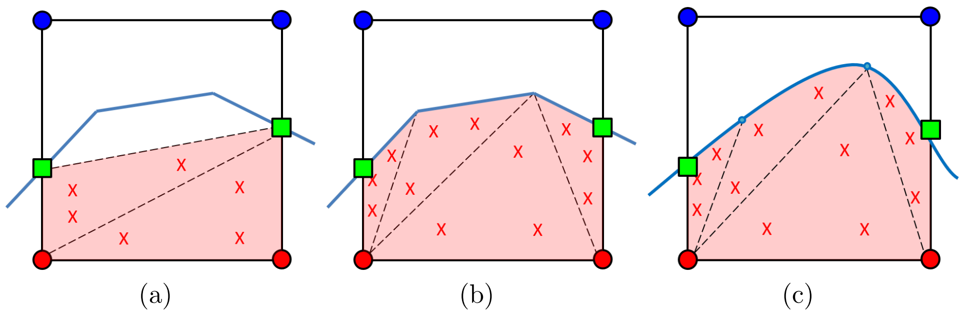


Figure 14: Treatment of a linear piecewise boundary within an element. (a) Subdomains template. (b) Exclusive subdomain creation. (c) Exclusive subdomain creation and NURBS-Enhanced integration.

In order to improve the robustness of the method we extend the previously described concepts by assuming the existence of intersections on the nodes of the element, or in other words, boundary nodes. These possibilities were not considered in the first development and are very likely to exist when dealing with complicated CAD geometries and h-adaptive mesh generation. Although the exact intersection on the node is not, in general, very likely, we can have many of these intersections due to the use of a geometrical tolerance, so intersections of the surface with element edges within the geometrical tolerance of the element nodes will be considered as intersections on the nodes.

For example, if we take Configuration 1 in Figure 10a, we can quickly see that we have as many options as intersections we can move to the nodes (see Figure 15) the boundary nodes being the magenta dots.

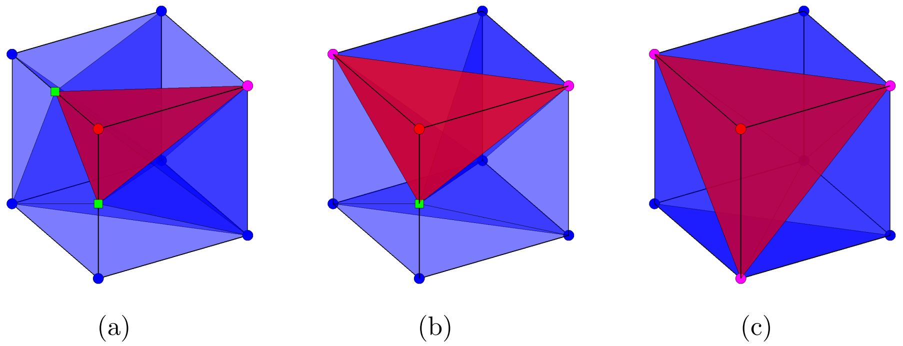


Figure 15: Configuration 1. Different options when considering surface intersections on boundary nodes. (a) One boundary node. (b) Two boundary nodes. (c) Three boundary nodes.

In this contribution we propose a strategy to handle all these new possibilities using exactly the same number of stored patterns, i.e. the original 7 patterns. This is possible because of the relationship between boundary nodes and intersections, and the latter with the integration patterns.

To explain the procedure we will use the Configuration 1 shown in Figure 9, which yields the cases seen in Figure 15. The numbering used is that of the one defined in Figure 8.

Starting with the case shown in Figure 15a, the strategy follows the next steps:

1. Position of nodes. We have to analyze the in/out topologies of the element nodes. Assuming the red nodes are the internal ones, in Figure 15a we can observe that only node 1 is internal. With the position of the nodes defined we can then identify the intersection pattern, in this case Configuration 1 in Figure 10a.
2. Intersection topology. The intersection topology comes automatically with the intersection pattern. In our example, Configuration 1 needs intersection 1, 4 and 5 to be able to generate the tetrahedralization.

3. Boundary node connectivity. Each node is related to 3 intersections on the edges shown in Table 1. In the example, the node on the boundary is n^o 2, which is related to edges 1, 2 and 6.

Node	Related intersections on edges
1	$1-4-5$
2	$1-2-6$
3	$2-3-7$
4	$3-4-8$
5	$5-9-12$
6	$6-9-10$
7	$7-10-11$
8	$8-11-12$

Table 1: Relations between nodes and intersections.

4. Boundary node to intersection. Figure 15a shows that we only count intersections 4 and 5, but Configuration 1 needs intersections 1, 4 and 5. Then, knowing that node 2 is related to intersections 1, 2 and 6, in which 2 and 6 are not used in this pattern, we can assume that intersection 1 is equivalent to node 2 when generating the pattern.
5. Tetrahedra generation. If we generate this reference tetrahedralization with the coordinates of the boundary node 2 as intersection 1 included, then the result will be similar to Figure 15a, but obviously if we use the original set of tetrahedrons some of them will be collapsed (volume zero) as the location of intersection 1 will coincide with node 2. These zero-volume tetrahedrons will be removed because they will not add anything to the integration step.

The cases represented in Figures 15b and 15c, in which each boundary node will be related to a determined intersection, are further cases in which the procedure could be used. This strategy will also work for all configurations and not only for these simple cases.

Remark 2. *It is important to note that some patterns can yield the same degenerated tetrahedralization. For example, in Figure 16a we have the original intersection pattern of Configuration 2. If intersections 2 and 6 were on node 2, the latter would become a boundary node, as in Figure 16b. The final tetrahedralization will be equivalent to the one shown in Figure 15a, which* *is topologically identical to the one obtained from Configuration 1 (Figure 15a), therefore leading to ambiguity. Although the tetrahedrons of both configurations do not coincide, the result after the integration will be the same. However, this case will not appear during the execution of the algorithm because if we compare Figures 16a and 16b we can observe how node 2 is transformed from an internal (red dot) to a boundary node (magenta dot). This violates the requirement of conserving the original in/out node topology, so even though the result were correct, we do not allow it to avoid ambiguity during the process.*

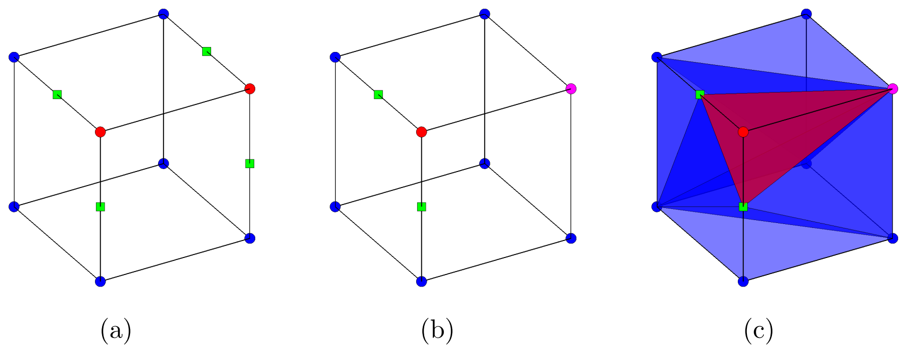


Figure 16: Configuration 2. Degeneration equivalent to Configuration 1. (a) Intersection pattern. (b) One boundary node. (c) Resulting tetrahedralization.

To summarize; the strategy proposed will yield 3 different options, depending on the case under study:

1. If the topology of the nodes and the intersected edges coincide with any of the precomputed patterns and there are no boundary nodes, then we directly use the standard pattern assigned.
2. If there is a match between the nodal in/out topology and the stored patterns, and there is at least one boundary node present, we will proceed as explained above to obtain the proper tetrahedralization from an original pattern without adding computational cost.
3. When the intersection pattern of an element is not stored or there are several surfaces intersecting the element, we will proceed with a Delaunay tetrahedralization of the element.

5. Mesh refinement

As mentioned in Section 1, there are several strategies to tackle the mesh refinement. In this contribution we propose a procedure based on the subdivision of the integration region into successively smaller nested sub-regions, thus modifying the density of elements to yield a more precise solution, keeping the element polynomial order constant.

The three main components of the h-adaptive finite element analysis we propose are:

1. Calculation of the parameters used to drive the subdivision process. They could be geometrical parameters or we can use parameters obtained from the finite element solution, for example, the estimated error in energy norm or any other quantity of interest.
2. Mesh generation. Since we are using Cartesian grids independent of the geometry we do not need to generate a new mesh from the beginning, instead we stick to the first mesh and subdivide the elements flagged by the parameters calculated. Note that we will consider mesh conformity, i.e. the maximum refinement difference between two elements adjacent by face or edge is limited to one level, and Multipoint constraints (MPCs)[46, 47] are used to enforce C^0 continuity between adjacent elements of different levels.
3. Projection of variables from the old mesh to the new mesh. In this case, our hierarchical data structure allows the automatic transference of properties from old elements to new ones.

Then, the input to this h-refinement procedure is a uniform coarse mesh and a prescribed limit to the refinement level. Both the initial level of the mesh and the maximum level of refinement will be chosen by the user. We propose a two-step adaptive meshing strategy:

1. Geometrical refinement to obtain the first mesh for the FE analysis. The elements of the initial grid mesh will be refined following geometrical considerations until a mesh properly adapted to the geometry is obtained. This mesh will be the first mesh used for the FE analysis.
2. Solution based refinement. After the FE analysis of the first mesh, the mesh refinement is guided by estimations of the error in the energy norm or in magnitudes of interest evaluated from the FE solution.

5.1. Geometrical refinement

The refinement based on the features of the geometrical model is widely used in FEA, since the user can easily identify where the mesh should be finer to properly capture the boundaries of the models. For any kind of geometrical representation, from tesselations to advanced parametric surfaces, it is possible to define parameters to evaluate changes of curvature, small features and any other characteristic that could influence the Finite Element solution if the discretization is not properly defined on the boundaries of the models. However, as in any other mesh generation task, choosing the proper element size for different areas and obtaining a good quality mesh of a complicated model would require a considerable amount of time.

Our idea is to take advantage of the information already obtained during the boundary-mesh intersection step to evaluate the goodness of the mesh even before the resolution and to ensure that the requirements imposed by the integration procedure are fulfilled (such as the need to ensure that each edge of a boundary element cannot contain more than one intersection with the boundary). We have to clarify that during the intersection process the geometry is intersected with several levels of refinement of the Cartesian grid. For instance, if the user sets the initial and the maximum levels allowed to levels 2 and 9, the geometry will be intersected in a preprocess stage with a level 5 mesh, which includes the edges of coarser meshes. This is done using the Newton-Raphson algorithm (see Section 3) and gives useful extra information about the boundary. The remaining levels of refinement will be intersected locally as they appear in the discretization.

We will illustrate the criteria implemented using 2D examples for clarity.

The first criterion is the simplest and the most frequently used. Figure 17 shows intersections with the elements of the current mesh as large green squares, internal nodes of the actual element as red dots and external nodes of the current element as blue dots. It also shows a virtual subdivision of the element, up to the maximum level the user would allow, as well as the corresponding intersections of the boundary with the refined mesh, represented by small green squares.

Figure 17a shows the unit normal vectors $\hat{u}$ calculated during the intersections process. We know that in a 3D Cartesian system the components of any unit normal vector $(\hat{x}, \hat{y}, \hat{z})$ are bounded in the interval $[-1, 1]$, so the maximum span of variation within an element and for any of the Cartesian directions would be 2. Since we are interested in a relatively smooth representation inside every element cut by the boundary, we can measure

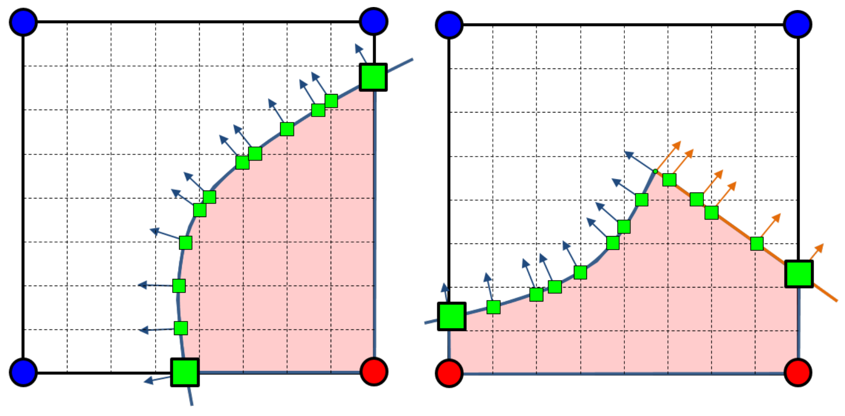

(a) (b)

Figure 17: Geometrical features within elements. Variation of unit vectors with an (a) smooth boundary or a (b) sharp edge.

the variability of the unit normal vectors limiting the span, of this variation within an element, to a threshold value. From our experience, if the variation of the components of the unit normal vectors exceeds the value of 1, then the element cannot be considered valid and will be refined. Figure 17b gives another example, but in this case there is a sharp edge due to the change of curve (surface in 3D). In this case we only need to measure the change of the unit normal vector of the union point to evaluate the abruptness of the change of definition. When integrating the boundary with NURBS-Enhanced techniques, this criterion is not very important due to the ability of the proposed methodology to properly capture the volume, but when dealing with other piecewise approximations this criterion is key to obtaining good discretization of the mesh along the boundary, as mentioned in Remark 1.

The next criterion completes the previous one and is related to the nature of the problem. FEAVox was first implemented to solve linear elasticity problems in which some internal corners could originate singularities and produce large gradients when evaluating displacements or stresses. These cases require an adequate discretization around the singularities to obtain a better representation of the singular solution. In Figure 18a we see an example of a corner, where P_0 is the intersecting point between the two curves in the element, P_I is the centroid of the intersections and $P_\epsilon = P_0 + \epsilon$ being ϵ

a differential of $\hat{u_1} + \hat{u_2}$ ($\hat{u_1}$ and $\hat{u_2}$ are unit normal vectors calculated in both curves intersecting in P_0). Then the corner is re-entrant to the geometry, thus a potential singularity, if $|P_I - P_\epsilon| < |P_I - P_0|$. With this condition we can assume that the corner is concave with respect to the material and in this case a refinement around that point will be automatically generated, see Figure 18b. In 3D this evaluation will occur at several points along the intersections curve between surfaces.

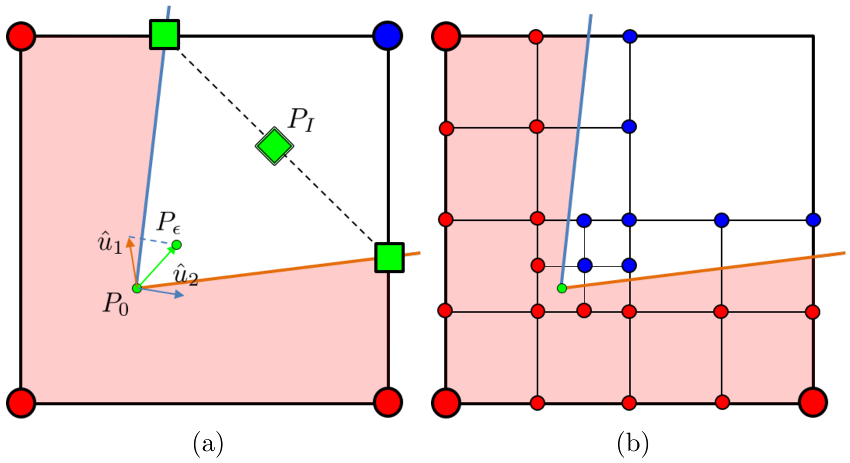


Figure 18: Identification of singularities. (a) Measure of distances. (b) Example of refinement.

The ability to represent all the small features of a geometrical model is very important for all mesh generators, especially if we aim to develop a mesh generator in which the mesh is independent of the geometry. Figure 19a gives a clear example in which a small feature, a small ellipse in this case, will not be considered during the integration due to the element size. Our solution to this problem is to locally refine the elements of the mesh until the intersections related to all the geometrical entities of the model are present in the mesh, as shown in Figure 19b. To detect these small entities we have to take into account that it is very easy to know if we are using intersections of all the geometrical entities in the actual mesh and the ease of locating points in the Cartesian elements using our hierarchical data structure.

The last refinement criterion comes naturally with the mesh generation strategy implemented. As we explained in Section 4, only 7 out of the 14 configurations of the original Marching Cubes algorithm were taken into ac-

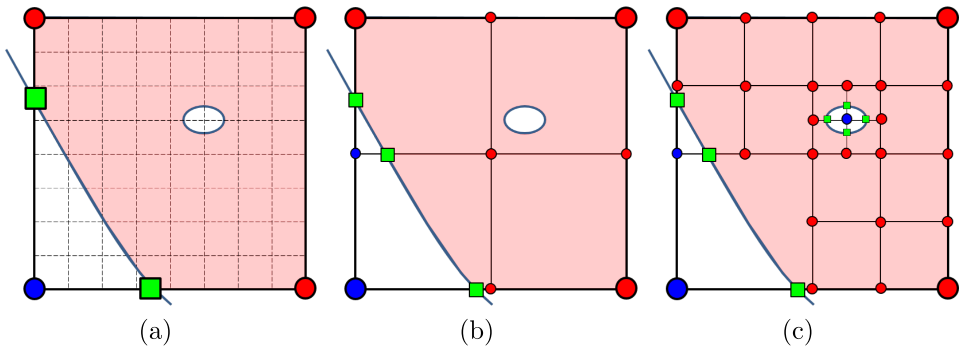


Figure 19: Detection of small features. (a) Coarse element. (b) First step of refinement. (c) Final step of refinement.

count because they refer to non-ambiguous configurations. In [30] we predicted the use of the ambiguous patterns to locate areas where refinement was necessary. Obviously there will be cases where ambiguities will appear, for instance with highly complicated geometries (see Figure 20a). In any case, as good discretization will be necessary, we will refine these elements to obtain simpler intersection patterns, as can be seen in Figure 20b. As explained in Section 4, in a multi-body environment where we will need to generate the integration subdomains exclusively for each body to ensure consistency with the parametric spaces, the ambiguous patterns can be handled as combinations of simple ones.

5.2. Error-based refinement

In FEM, the discretization error is defined as the difference between the exact and the approximate solutions obtained from the finite element analysis, without taking into account the round-off and modeling errors. It is commonly measured in terms of the energy norm, which represents the error as a scalar quantity. In terms of stresses, the error in the energy norm, $\|\mathbf{e}\|$, can be written as

$$\|\mathbf{e}\| = \sqrt{\int_\Omega (\boldsymbol{\sigma}_h - \boldsymbol{\sigma})^T \mathbf{D}^{-1} (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}) \mathrm{d}\Omega} \qquad (1)$$

where $\mathbf{D}$ is the material stiffness matrix, $\boldsymbol{\sigma}_h$ is the FE stress field and $\boldsymbol{\sigma}$ is the exact stress field.

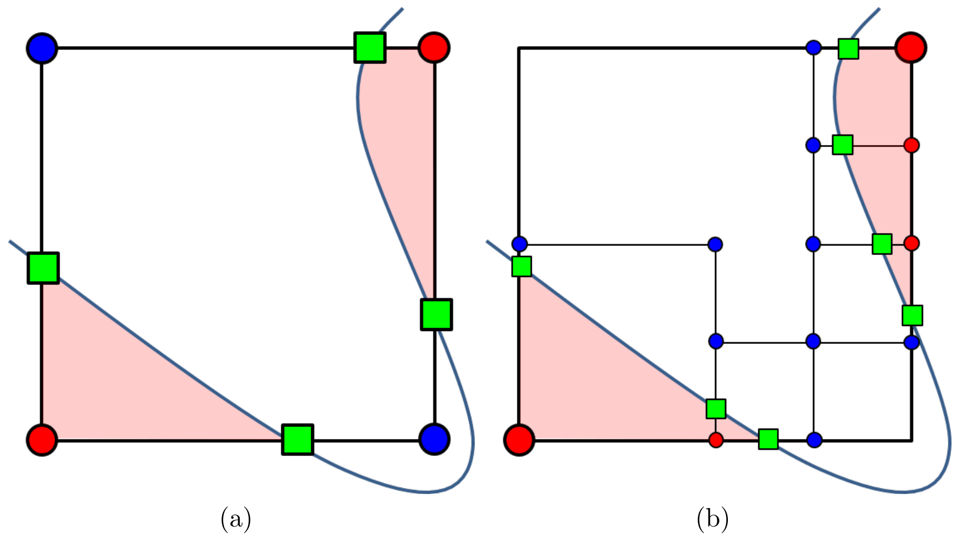


Figure 20: Undefined pattern and elimination through refinement. (a) 2D projection of the pattern. (b) Solution through refinement.

The error at each element can be evaluated by integrating (1) on each individual element of the mesh. Let $\|\mathbf{e}^{(i)}\|$ be the exact error in energy norm of the element i. The following equation, where M is the total number of elements in the mesh, relates the global and local errors

$$\|\mathbf{e}\|^2 = \sum_{i=1}^{M} \|\mathbf{e}^{(i)}\|^2 \tag{2}$$

Several types of error estimators have been proposed in the literature depending on the obtaining procedure: residual error estimators [1, 48, 49, 50] use the residuals of the approximate solution to evaluate the error, the Constitutive Relation Error (CRE) [51] consisting in the comparison of statically admissible stress fields with kinematically admissible stress fields, the estimators based on dual analysis [52, 53] and making use of two solutions of the problems. Finishing the classification, we find the recovery based error estimators. Proposed by Zienckevick and Zhu [54], these estimators use a recovered solution, $\boldsymbol{\sigma}^*$, instead of the exact solution $\boldsymbol{\sigma}$ to measure the error [55, 56].

Assuming that it is possible to write the previous convergence rates as a function of the estimated errors, we could use the Zienkiewicz and Zhu (ZZ) error estimator to reformulate (2), for the i^{th}-element, as

$$\|\mathbf{e}_{es}^{(i)}\| = \sqrt{\int_{\Omega^{(i)}} (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^*)^T \mathbf{D}^{-1} (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^*) \mathrm{d}\Omega} \tag{3}$$

where $\boldsymbol{\sigma}^*$ is a smoothed continuous stress field obtained using a 3D version of the recovery technique presented in [57, 28].

The refinement algorithm makes use of the estimated element errors to define a new mesh. The algorithm can be based on a type of optimality criterion to obtain new meshes of the prescribed accuracy level. In this work we propose a 3D generalization of the strategy presented in [54, 58] in which the optimality criterion is that of equidistributing the error on the elements of the new mesh. In [58] it was shown to be equivalent to the criterion of minimization of the number of elements in the new mesh to reach the prescribed error level with proper convergence rates. Let us assume that we are in mesh $n-1$ (current mesh) and we want to evaluate mesh n (new mesh), then:

$$h_n^{(i),n-1} \approx h_{n-1}^{(i)} \left[\frac{1}{M_{n-1}}\right]^{1/2(p+1)} \left[\frac{\|\mathbf{e}\|_n}{\|\mathbf{e}\|_{n-1}}\right]^{\frac{d}{2p^2+pd}} \left[\frac{\|\mathbf{e}\|_n}{\|\mathbf{e}^{(i)}\|_{n-1}}\right]^{\frac{2}{2p+d}} \tag{4}$$

where the quantities are:

$h_{n-1}^{(i)}$ is the size of the element i of the mesh $n-1$,

$h_n^{(i),n-1}$ is the new element size of the mesh n obtained by the subdivision of element i in the mesh $n-1$,

M_{n-1} is the number of elements in the mesh $n-1$,

$\|\mathbf{e}\|_n$ is the global error in energy norm of the mesh n.

$\|\mathbf{e}\|_{n-1}$ is the global error in energy norm of the mesh $n-1$,

$\|\mathbf{e}^{(i)}\|_{n-1}$ is the error of the element i of the mesh $n-1$,

p is the polynomial degree of the shape functions used,

d is the dimension of the problem (2 for 2D, 3 for 3D problems).

Replacing $\|\mathbf{e}^{(i)}\|$ in (4) by the estimation given in (3) we obtain the practical formula to evaluate the new element sizes. After obtaining the element size it is easy to find the refinement level necessary for the elements. Complete details of the procedure to reach this expression are given in Appendix A.

6. Numerical examples

This section gives a series of examples to demonstrate the applicability and the performance of the proposed methodology for 3D problems when the boundary of the domain is described by parametric surfaces. First, the proposed strategy is applied for the numerical solution of a linear-elastic problem, with an analytical solution and a simple geometry, to evaluate the convergence of the method. Then an example of a mechanical component will be described to show the applicability of the meshing procedure to more complex geometries given by CAD models, including trimmed surfaces.

6.1. Convergence analysis

For this study we consider a thick-wall cylinder loaded with internal pressure. The geometrical model for this problem is represented in Figure 21. A linear-elastic analysis is performed on a domain given by a CAD model that uses NURBS to represent the boundary. Only 1/4 of the section is modeled together with the appropriate symmetry and plain strain boundary conditions. The internal and external surfaces are of radius a and b with $a = 5$, $b = 20$. Young's modulus is $E = 1000$, Poisson's ratio is $\nu = 0.3$ and the applied load $P = 1$.

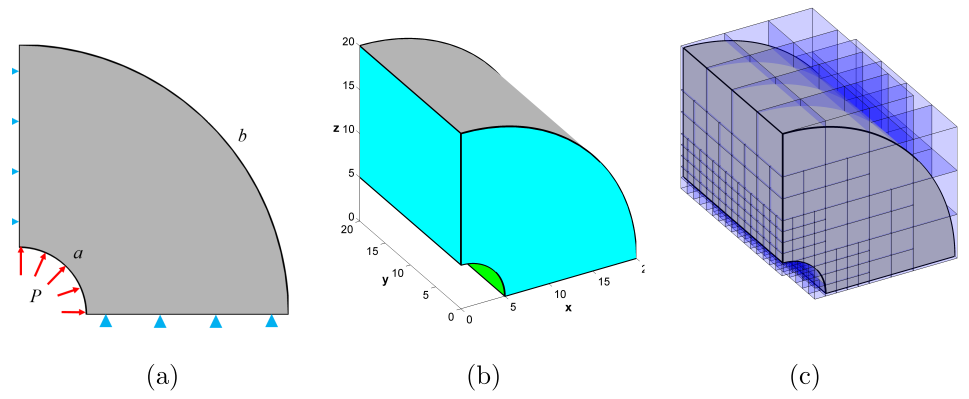


Figure 21: Model of a cylinder under internal pressure. (a) Front view with boundary conditions. (b) 3D model representation. (c) Example of analysis mesh.

The exact solution in cylindrical coordinates for displacements and stresses is given by:

$$u_r = \frac{P(1+\nu)}{E(k^2-1)} \left(r\,(1-2\nu) + \frac{b^2}{r} \right), \qquad u_l = 0 \tag{5}$$

$$\sigma_r = \frac{P}{k^2-1}\left(1 - \frac{b^2}{r^2}\right), \qquad \sigma_\phi = \frac{P}{k^2-1}\left(1 + \frac{b^2}{r^2}\right), \qquad \sigma_l = \nu\left(\sigma_r + \sigma_\phi\right) \tag{6}$$

where $k = b/a$, $r = \sqrt{x^2+z^2}$. Defining $\phi = \arctan(z/x)$ we transform to Cartesian coordinates:

$$u_x = u_r \cos(\phi), \qquad u_y = 0, \qquad u_z = u_r \sin(\phi) \tag{7}$$

$$\sigma_x = \sigma_r \cos(\phi)^2 + \sigma_\phi \sin(\phi)^2, \qquad \sigma_z = \sigma_r \sin(\phi)^2 + \sigma_\phi \cos(\phi)^2,$$

$$\sigma_y = \nu\left(\sigma_x + \sigma_z\right), \qquad \tau_{xz} = (\sigma_r - \sigma_\phi)\sin(\phi)\cos(\phi), \qquad \tau_{xz} = \tau_{yz} = 0$$

The quality of the results will be assessed by evaluating the relative error in the displacement field in energy norm, defined as

$$\eta_e = \left(\frac{\displaystyle\int_\Omega (\boldsymbol{\sigma}_h - \boldsymbol{\sigma})\,\mathbf{D}^{-1}\,(\boldsymbol{\sigma}_h - \boldsymbol{\sigma})\,\mathrm{d}\Omega}{\displaystyle\int_\Omega \boldsymbol{\sigma}\mathbf{D}^{-1}\boldsymbol{\sigma}\mathrm{d}\Omega} \right)^{1/2} \tag{8}$$

where $\boldsymbol{\sigma}_h$ and $\boldsymbol{\sigma}$ are the FE (approximated) and the analytical stresses respectively.

In the first analysis we will study the convergence for both tri-linear (L8) and tri-quadratic (Q20) elements in a set of uniformly refined meshes. For the linear analysis, we will compare the NURBS-enhanced (NeApprox) geometry approximation with a linear approximation (LinApprox) of the geometry. The linear approximation considers that the faces of the tetrahedrons, used to integrate the boundary elements, are represented by flat facets. On the other hand, for the analysis with tri-quadratic elements, we will compare NURBS-enhanced and quadratic approximations (QuadApprox) of the geometry. In this case, the facets lying on the boundary will be approximated with quadratic triangles.

Front views of the 3D meshes used in this simulation can be seen in Figure 22. Table 2 summarizes the main features of the five computational meshes. In particular, this table shows the number of elements that are interior to the embedded domain (A) and the number of elements intersecting the boundary of the embedded domain. Boundary elements can be separated into elements integrated by templates (B) and elements for which an exclusive tetrahedralization was necessary (C). In columns A to C we can find the proportion of those elements with respect to the total number of elements in the mesh. The last column shows the number of tetrahedra used to perform the numerical integration (D).

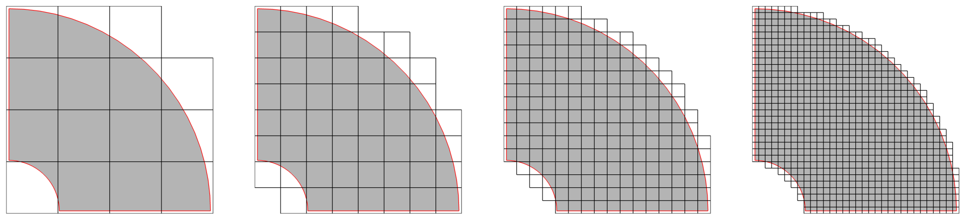


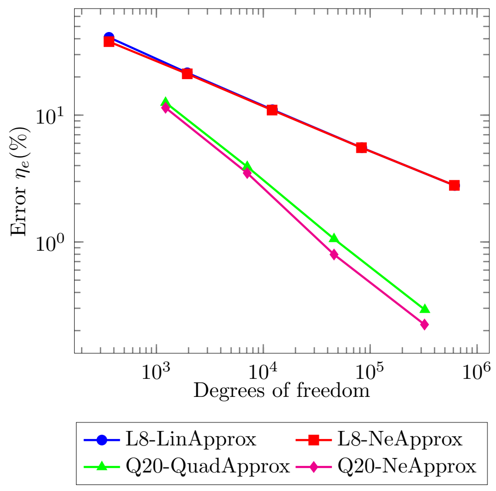
Figure 22: Front view of the first four meshes of the uniform refinement process (L8 and Q20).

Mesh	Internal (A)	Boundary temp. (B)	Boundary excl. (C)	Tetrahedra (D)
1	6 (10%)	22 (36.7%)	32 (53.3%)	569
2	162 (36.9%)	198 (45%)	80 (18.1%)	2225
3	2016 (61.7%)	1072 (32.9%)	176 (5.4%)	8369
4	19440 (78.7%)	4896 (19.9%)	368 (1.4%)	33393
5	171368 (88.8%)	20904 (10.1%)	752 (1.1%)	133603

Table 2: Topology of the approximation meshes in terms of different types of elements and subdomains.

Figure 23 shows the convergence of the relative error in energy norm for both tri-linear and tri-quadratic elements with the different geometry approximations. On the right plot in Figure 23 we show the convergence rate of the exact error in energy norm of the FE solution as a function of the number of degrees of freedom.

For problems with non-singular solution the theoretical predicted convergence rate in energy norm is $O(h^p)$. Therefore, following the rationale of [59], taking into account that the number of degrees of freedom (N) in 3D is approximately inversely proportional to h^3 the convergence rate can be written as $O(N^{-p/3})$. Hence, expressed in terms of the number of degrees of



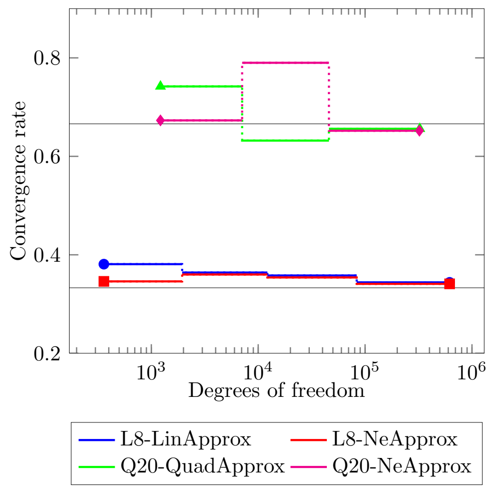


Figure 23: Cylinder convergence study. (Left) Relative error in energy norm. (Right) Convergence rate.

freedom, the convergence rate of the error in energy norm is $O(N^{-1/3})$ for tri-linear elements and $O(N^{-2/3})$ for tri-quadratic elements.

The values of the convergence show almost optimal rates for both tri-linear and tri-quadratic elements, and both isoparametric (linear/quadratic) interpolation of the geometry and the NURBS-Enhanced approach.

Table 3 shows the computational cost of the different modules of the FEA of the five meshes solved above with tri-linear elements (L8) and linear approximation to the geometry. In particular, this table shows the computational cost of the geometry-mesh intersection, the integration of elements (including the generation of mappings) and the solver step which gathers the assembly and the resolution of the system of equations.

We can observe how the intersection stage reduces its weight while refining. In addition, the proportion of local mesh generation reduces because it is related only to the interfaces between surfaces. Last, the computational cost related to the resolution of the system of equations increases until it becomes the most expensive part of the process.

It is interesting to note that both the isoparametric approach of the boundary and the NURBS-Enhanced integration of the domain yield the proper rates of convergence. Despite these similarities, in Figure 24 the geometrical errors obtained in the first four meshes can be compared. The geometrical error has been calculated as $\eta_V(\%) = \frac{|V_h-V|}{V} \cdot 100$, where V_h

Mesh	Intersection	Integration			Solver
		Internal	Boundary Temp.	Boundary Excl.	
1	0.157 (22.1%)	0.023 (3.2%)	0.076 (10.7%)	0.329 (46.3%)	0.125 (17.7%)
2	0.188 (13.1%)	0.016 (1.1%)	0.411 (28.7%)	0.644 (45.0%)	0.172 (12.1%)
3	0.281 (6.4%)	0.035 (0.8%)	1.909 (42.9%)	1.499 (33.7%)	0.718 (16.2%)
4	0.797 (3.4%)	0.063 (0.3%)	9.147 (39.1%)	3.364 (14.3%)	10.056 (42.9%)
5	3.641 (0.8%)	0.512 (0.2%)	60.844 (14.8%)	9.228 (2.3%)	336.261 (81.9%)

Table 3: Computational cost breakdown. Meshes of tri-linear elements (L8) and linear approximation of the geometry. Time measured in seconds.

is the volume integrated with the finite element mesh and V is the exact volume of the model shown in Figure 21. The global convergence of the geometrical error for linear and quadratic approximations agrees with the convergence rates expected. The oscillations observed during the analyses can be related to the fact that the discretization of the external cylindrical surface underestimates the volume and, in parallel, the discretization of the internal cylindrical surface overestimates the volume. Since the mesh is fixed, the refinement does not occur in the same uniform way in both surfaces thus making more pronounced this effect in coarse meshes. NURBS-Enhanced volume integration shows a very low almost constant error during the process, several orders of magnitude lower than approximating curved domains with low degree approximations. These results show that the proposed approach provides accurate integration of the domain, regardless of the refinement level of the models.

Table 4 shows a comparison in terms of accuracy and computational cost of different integration schemes for a given mesh. We use the mesh number 3 represented in Figure 22. We can observe that using the NEFEM approximation allows to reduce dramatically the geometrical error when increasing the number of Gauss points used to integrate the model. In exchange, the computational cost is increased by a factor close to 4 and it does not vary much independently of the number of Gauss points used for the NEFEM approach. This can be explained because most cost related to NEFEM is devoted to ensure the proper collocation of points while their number is almost irrelevant.

We also performed an h-adaptive refinement process guided by the local error estimation in energy norm, as explained in Section 5.2. Figure 25 shows the first four h-adapted meshes for tri-linear and tri-quadratic elements. In

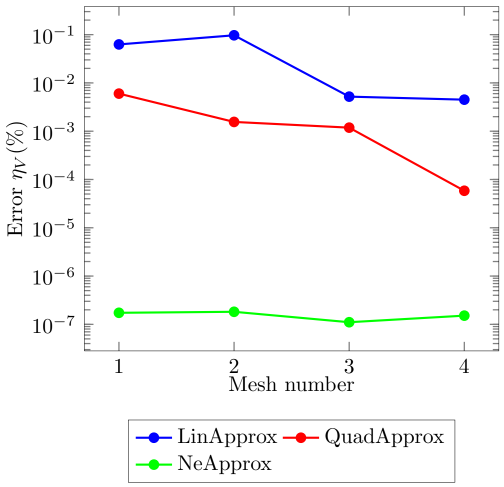


Figure 24: Cylinder volume convergence study.

Approx. Type	Num. Gauss points	Error η_V (%)	Integration time (s)
Linear	35524	0.0052	3.440
NEFEM	69226	1.4968e-5	14.746
NEFEM	76187	5.5125e-7	14.871
NEFEM	81754	4.2966e-9	15.549
NEFEM	94282	1.2212e-11	16.025

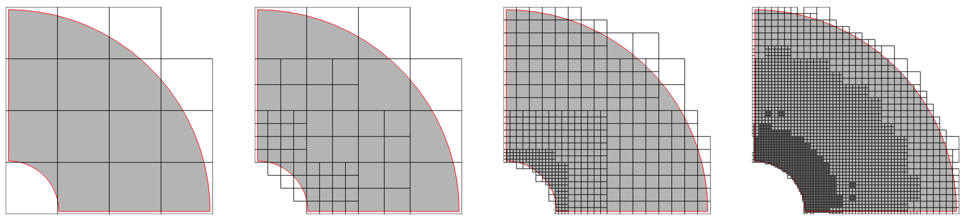
Table 4: Computational cost and accuracy of different integration schemes.

Figure 26 we compare the results of uniform meshes and h-adapted meshes, considering the NURBS-Enhanced approach. The figure shows the convergence of the relative error in energy norm for both tri-linear and tri-quadratic elements.

In the graphs it can be seen how the convergence for the h-adapted meshes in the first stages of the refinement processes (in the pre-assymptotical range) are above the optimal rate. This leads to reaching an specified error level using fewer degrees of freedom and, hence, to a lower computational cost.

6.2. Geometrical refinement sample

With this problem our purpose is to show the performance of the h-adaptive geometrical refinement process in more complex geometries. Naturally, in this type of problem there is no available exact solution, so our objective is to check whether the criteria proposed here provide a mesh suitably adapted to the geometrical features of the model.


(a) h-adaptive meshes for tri-linear elements (L8).

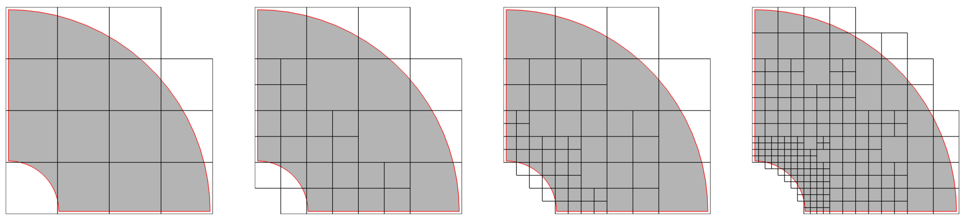

(b) h-adaptive meshes for tri-quadratic elements (Q20).

Figure 25: Front view of the first four meshes of the h-refinement process.

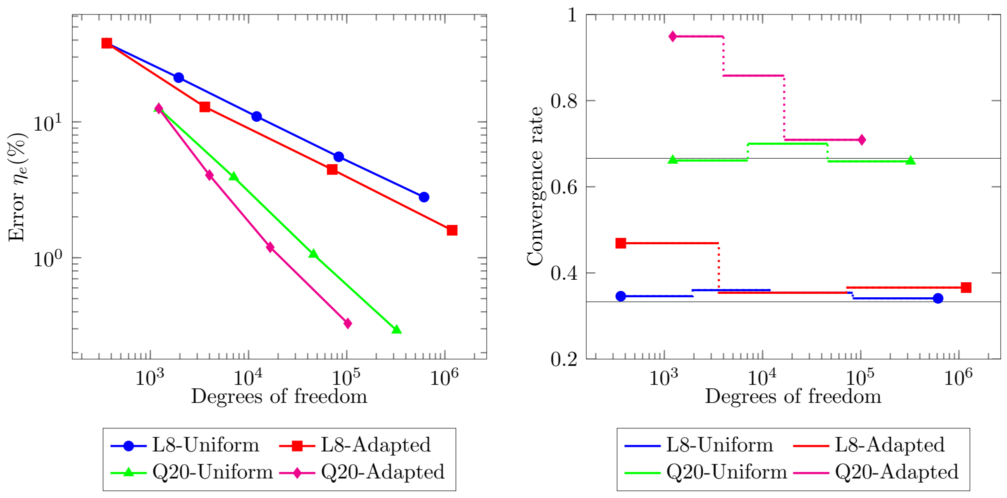
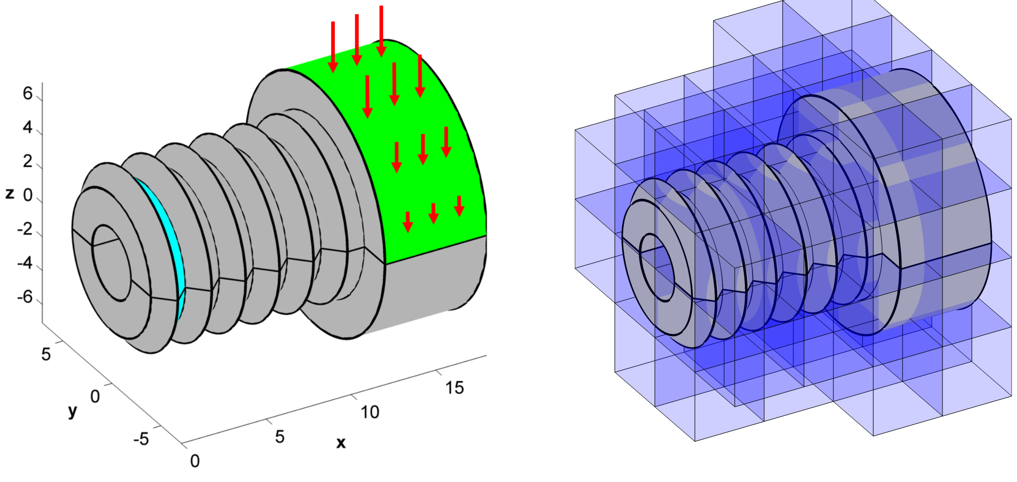


Figure 26: Cylinder convergence study. h-adaptive case. (Left) Relative error in energy norm. (Right) Convergence rate.

We used a complex model to test the proposed strategy. The model selected represents a perforated screw, as shown in Figure 27, with a topology as used in hydraulic applications. In this case, we restrained the displacements of the surfaces in blue and applied a variable vertical force per unit of area. The material was steel with Young's modulus $E = 2,1 \cdot 10^9 Pa$ and Poisson's ratio $\nu = 0,333$.



(a) (b)

Figure 27: Model of a hydraulic screw. (a) CAD model and boundary conditions. (b) Initial coarse mesh.

Figure 27b shows the coarse initial mesh used in the process. It can be seen that this element size is unlikely to properly capture the features of the screw threads. Figure 28a shows a refined mesh using the criteria proposed here. The refinement properly captures the features of the model and focuses on the re-entrant corners between surfaces. Figure 28b represents the Von Mises stress field, where the stress concentration can be seen along the singularities produced by the re-entrant corners of the model.

To make clear how boundary elements are treated, Figure 28c shows a section of the refined mesh, distinguishing between internal elements (green) and the integration subdomains of the cut elements conforming to the geometry (blue). Figure 28d gives a detailed view of the section.

After the geometrical refinement, we move forward in the simulation with a refinement based on the discretization error.

Table 5 summarizes the topological features of the meshes used for this

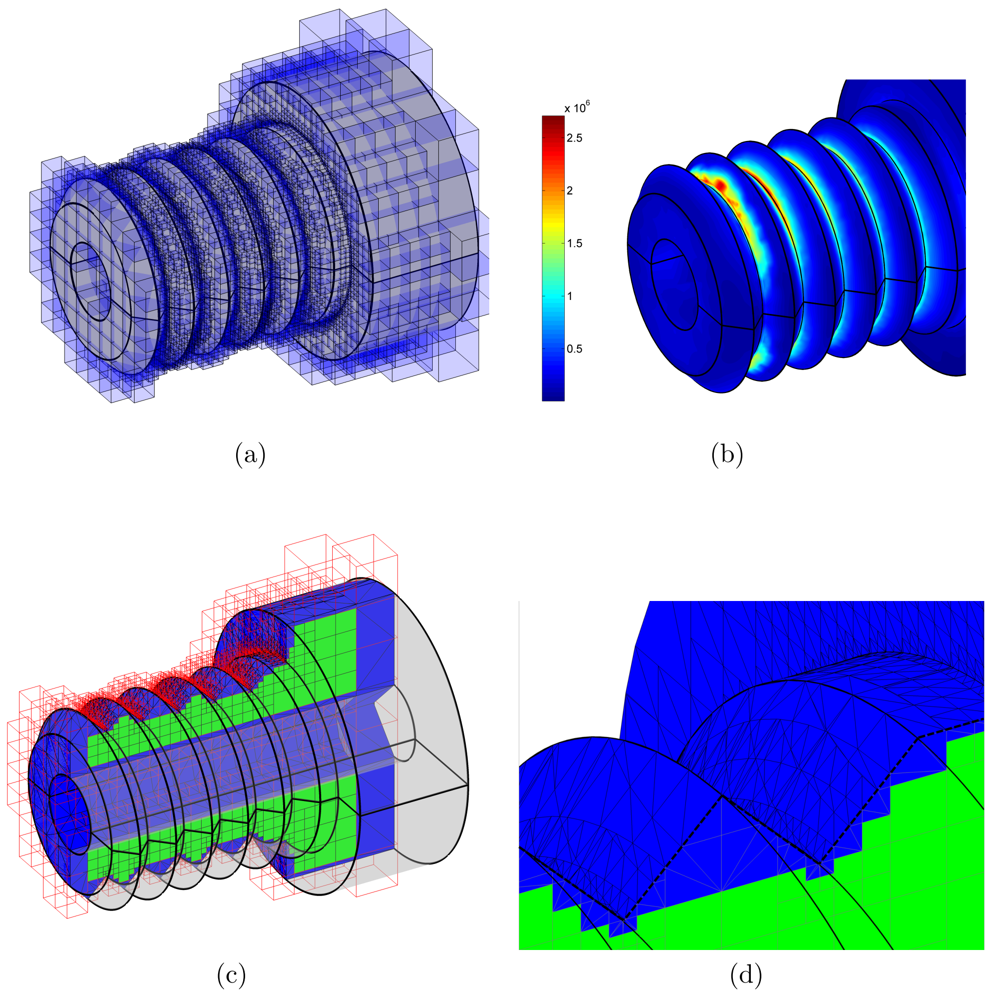


Figure 28: Geometrical h-refinement. (a) Approximation mesh, (b) Von Mises stress field, (c) section of the mesh and (b) detail of the integration subdomains.

analysis. The first mesh is the geometrically refined mesh shown in Figure 28d and the second mesh corresponds to the one obtained from the error-based refinement, see Figure 29b. It can be seen that, as in the previous example, the percentage of boundary elements decreases from a high value in the first mesh (obtained by geometrical refinement) to a considerably lower value after the error-based h-adaptive refinement. Despite of complexity of the model, with a high number of geometrical entities, the ratio of elements requiring exclusive tetrahedralization with respect to the total number of elements is only around 10% in the first mesh and further decreases to 2% in the second mesh.

Mesh	Internal	Boundary temp.	Boundary excl.	Tetrahedra
1	9356 (48.4%)	8109 (41.9%)	1851 (9.7%)	73866
2	60223 (73.2%)	21199 (25.7%)	720 (2.1%)	150633

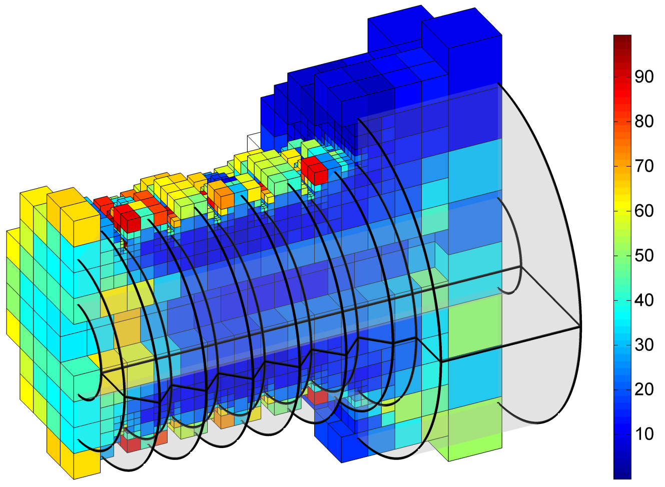
Table 5: Topology of the approximation meshes in terms of different types of elements and subdomains.

The global estimated error in energy norm for the first mesh is 20.86%. Figure 29a shows the element-wise relative estimated error in energy norm. For clarity we have plotted a section of the mesh to observe the distribution of error also in the internal elements. The error map shows that the error is larger along the singularities and the area where the Dirichlet conditions where applied. These errors will drive the h-refinement process that will result in the mesh shown in Figure 29b. We can observe higher density of elements in this mesh compared to the first one analyzed. Figure 29c represent the Von Mises stress field. We can also observe that the highest stress correspond to this mesh due to the better discretization. The estimated error in energy norm obtained for this mesh is 13.76%. Due to the presence of singularities in the problem the convergence rate is suboptimal, as expected.

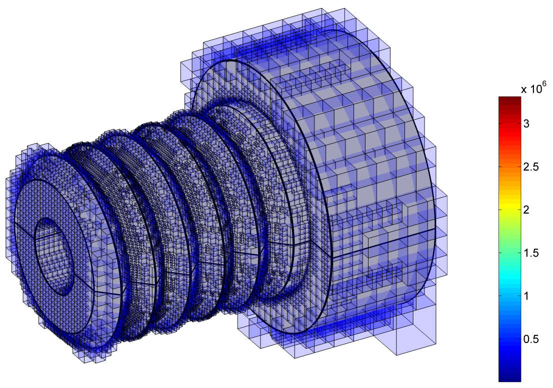
7. Conclusions

This paper proposes a novel method of generating h-adapted meshes, considering the exact 3D CAD boundary representation of the domain in an immersed boundary framework in which a Cartesian grid is used to mesh the embedding domain.

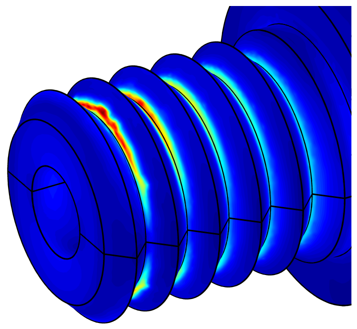
Details are included of the strategy used to compute the intersections between the Cartesian grid and the exact geometry of the boundary of the embedded domain. To perform the numerical integration in the region of the cut elements internal to the physical domain, we propose the creation of a submesh of tetrahedra in each of the elements cut by the boundary. For this, we developed a system to obtain all possible configurations from only seven basic patterns of tetrahedra. The algorithms used to refine the finite element mesh are also described. First, in a preprocess step, the geometrical criteria were chosen to obtain the proper refined mesh. Secondly, as a postprocess



(a)



(b)



(c)

Figure 29: Error-based h-refinement. (a) Element-wise error estimation, (b) approximation mesh and (c) Von Mises stress field.

tool, an error based algorithm was adapted to obtain a new mesh considering a prescribed error reduction and the error in the elements of the previous mesh.

Two examples were given to demonstrate the potential and applicability of the proposed methodology. The optimality of the approximation in terms of error convergence rate, for both linear and quadratic elements, was corroborated by the problem of a cylinder under internal pressure. The geometrical error of different boundary approximations was also compared, showing the accuracy and consistency of NURBS-Enhanced techniques and a model of a hydraulic screw showed its ability to obtain refined meshes from geometrical parameters.

Acknowledgements

The authors wish to thank the Spanish textitMinisterio de Economía y Competitividad for the financial support received through Project DPI2013-46317-R and the FPI program (BES-2011-044080), also the *Generalitat Valenciana* for the assistance received through Project PROMETEO/2016/007.

Appendix A. *h*-refinement criteria based on error estimation

This appendix contains further details on the process of finding the h-refinement criterion based on error estimation. This algorithm originally designed for 2D problems [58] has been adapted to 3D. The equations resulting from the asymptotic rates of convergence of the FEM must first be explained. During h-refinement, for a uniformly refined succession of meshes, the exact error, $\|\mathbf{e}\|$, is bounded as follows:

$$\|\mathbf{e}\| \leq C_1 h^{min(p,\lambda)} \approx C_2 N^{-\frac{1}{d}min(p,\lambda)} \tag{A.1}$$

where N is the number of degrees of freedom; h is the characteristic element size; p is the polynomial degree of the interpolation functions being used; C_1 and C_2 are positive independent constants; and λ represents the intensity of the singularities. The exponent of N is known as the asymptotic rate of convergence and for a sequence of h-adapted meshes the bound of the exact error takes the form[60]:

$$\|\mathbf{e}\| \leq C_2 N^{-\frac{1}{d}c} \tag{A.2}$$

yielding the theoretical optimal convergence rate of the h-version of the FEM.

At the global level the ratio of the error in the new mesh to be created (n) to the error in the current mesh ($n-1$) is, considering (A.1), almost equal to the ratio of the size of the elements to the power of c. In our study we consider $c = p$ but, theoretically, assuming c as a constant in the presence of stress singularities is not appropriate. However, we allow that these results are approximately valid, and use them in an adaptive refinement. That is,

$$\frac{\|\mathbf{e}\|_n}{\|\mathbf{e}\|_{n-1}} \approx \left[\frac{h_n}{h_{n-1}}\right]^p \tag{A.3}$$

In the following we assume that the convergence shown in (A.3) is also valid at the element level. We have assumed that the uniform refinement of element i of the current mesh, whose size is $h_{n-1}^{(i)}$, produces $M_n^{(i),n-1}$ elements of size $h_n^{(i),n-1}$ and that the following expression holds:

$$\frac{\|\mathbf{e}^{(i),n-1}\|_n}{\|\mathbf{e}^{(i)}\|_{n-1}} \approx \left[\frac{h_n^{(i),n-1}}{h_{n-1}^{(i)}}\right]^p \tag{A.4}$$

where $\|\mathbf{e}^{(i)}\|_{n-1}$ represents the error in the element i of the previous mesh, and $\|\mathbf{e}^{(i),n-1}\|_n$ represents the error in each of the new elements included in the element i of the previous mesh. Therefore, if $\|\mathbf{e}^{(i)}\|_n$ represents the error in the new elements, the following relation is correct:

$$\|\mathbf{e}^{(i),n-1}\|_n^2 = \sum_{i=1}^{M_n^{(i),n-1}} \|\mathbf{e}^{(i)}\|_n^2 \tag{A.5}$$

We use (A.4) to predict size of elements of mesh n created in each of the elements of mesh $n-1$ required to obtain the preset error in energy norm at each of the new elements.

$$h_n^{(i),n-1} \approx h_{n-1}^{(i)} \left[\frac{\|\mathbf{e}^{(i),n-1}\|_n}{\|\mathbf{e}^{(i)}\|_{n-1}}\right]^{1/p} \tag{A.6}$$

To use this expression, we need the error of all the elements of the new mesh contained in the space defined by element i in mesh $n-1$, $\|\mathbf{e}^{(i),n-1}\|_n$.

Besides the equations resulting from the asymptotic rates of convergence, we also need to estimate the number of elements in the new mesh during adaptive mesh refinement.

At the element level, if we consider a uniform refinement, the number of elements $M_n^{(i),n-1}$ of size $h_n^{(i),n-1}$ in the new mesh n that are contained in the element i of size $h_{n-1}^{(i)}$ of mesh $n-1$ can be estimated as

$$M_n^{(i),n-1} \approx \left(\frac{h_n^{(i)}}{h_n^{(i),n-1}} \right)^d \tag{A.7}$$

Hence, the total number of elements in the new mesh, M_n, will be

$$M_n = \sum_{i=1}^{M_{n-1}} M_n^{(i),n-1} \approx \sum_{i=1}^{M_{n-1}} \left[\frac{h_{n-1}^{(i)}}{h_n^{(i),n-1}} \right]^d \tag{A.8}$$

where M_{n-1} is the number of elements in mesh $n-1$.

As assumed in (A.1), at the global level and considering a uniform refinement, the number of elements in the meshes is inversely proportional to the sizes of the elements to the power of d. That is

$$M_{n-1} h_{n-1}^d \approx M_n h_n^d \tag{A.9}$$

Taking into account (A.3), we can estimate the number of elements in the new mesh as a function of the number of elements in the previous mesh:

$$M_n \approx M_{n-1} \left[\frac{\|\mathbf{e}\|_{n-1}}{\|\mathbf{e}\|_n} \right]^{d/p} \tag{A.10}$$

As previously mentioned, the strategy follows the idea of a nearly optimal mesh in which the estimated error must be equidistributed on each element. Instead of using the previous mesh, the error distribution is made on the new mesh using Equation (A.10). This procedure seeks the definition of the element sizes of the new mesh in such a way that we have the same absolute error in each of its elements.

Using $\|\mathbf{e}^{(i)}\|_n$ as the exact error in each of the new elements, the global absolute error of the new mesh can be written as

$$\|\mathbf{e}\|_n^2 = \sum_{i=1}^{M_n} \|\mathbf{e}^{(i)}\|_n^2 = M_n \|\mathbf{e}^{(i)}\|_n^2 \tag{A.11}$$

where M_n is the number of elements in the new mesh. Since on the new mesh, $\|\mathbf{e}^{(i)}\|_n$ is the same for each new element, we obtain $\|\mathbf{e}^{(i)}\|_n^2$ by the following expression:

$$\|\mathbf{e}^{(i)}\|_n = \left[\frac{1}{M_n} \right]^{1/2} \|\mathbf{e}\|_n \tag{A.12}$$

Moreover, since we have Equation (A.10) to estimate the number of elements in the new mesh, we estimate the error in each element of the new mesh as:

$$\|\mathbf{e}^{(i)}\|_n = \left[\frac{1}{M_{n-1}}\right]^{1/2} \frac{\|\mathbf{e}\|_n^{(d/2p)+1}}{\|\mathbf{e}\|_{n-1}^{d/2p}} \tag{A.13}$$

However, we need $\|\mathbf{e}^{(i),n-1}\|_n^2$. Taking into account that we must have the same absolute error in each element of the new mesh, we can express $\|\mathbf{e}^{(i),n-1}\|_n^2$ in the following manner:

$$\|\mathbf{e}^{(i),n-1}\|_n^2 = \sum_{i=1}^{M_n^{(i),n-1}} \|\mathbf{e}^{(i)}\|_n^2 = M_n^{(i),n-1}\|\mathbf{e}^{(i)}\|_n^2 \tag{A.14}$$

where $M_n^{(i),n-1}$ is the number of new elements contained in the subdomain defined by element i of the mesh $n-1$.

Next, taking into account that we can estimate $M_n^{(i),n-1}$ with Equation (A.7), we obtain the estimated error $\|\mathbf{e}^{(i),n-1}\|_n$ as

$$\|\mathbf{e}^{(i),n-1}\|_n^2 \approx \left[\frac{h_{n-1}^{(i)}}{h_n^{i,n-1}}\right]^d \|\mathbf{e}^{(i)}\|_n^2 \tag{A.15}$$

Therefore, using Equation (A.13) we obtain $\|\mathbf{e}^{(i),n-1}\|_n$ by the following expression:

$$\|\mathbf{e}^{(i),n-1}\|_n \approx \left[\frac{h_{n-1}^{(i)}}{h_n^{(i),n-1}}\right]^{d/2} \left[\frac{1}{M_{n-1}}\right]^{1/2} \frac{\|\mathbf{e}\|_n^{(d/2p)+1}}{\|\mathbf{e}\|_{n-1}^{d/2p}} \tag{A.16}$$

Finally using Equation (A.6) we obtain the new element size for each of the elements of the previous mesh as

$$h_n^{(i),n-1} \approx h_{n-1}^{(i)} \left[\frac{1}{M_{n-1}}\right]^{1/2(p+1)} \left[\frac{\|\mathbf{e}\|_n}{\|\mathbf{e}\|_{n-1}}\right]^{\frac{d}{2p^2+pd}} \left[\frac{\|\mathbf{e}\|_n}{\|\mathbf{e}^{(i)}\|_{n-1}}\right]^{\frac{2}{2p+d}} \tag{A.17}$$

where all quantities are well defined.

References

[1] I. Babuška, C. Rheinboldt, A-posteriori error estimates for the finite element method, International Journal for Numerical Methods in Engineering 12 (1978) 1597–1615.

[2] I. Babuška, C. Rheinboldt, Adaptive approaches and reliability estimates in finite element analysis, Computer Methods in Applied Mechanics and Engineering 17-18 (3) (1979) 519–540.

[3] M. Paraschivoiu, J. Peraire, A. T. Patera, A posteriori finite element bounds for linear-functional outputs of elliptic partial differential equations, Computer Methods in Applied Mechanics and Engineering 150 (1-4) (1997) 289–312.

[4] P. Ladevèze, P. Rougeot, P. Blanchard, J. P. Moreau, Local error estimators for finite element linear analysis, Computer Methods in Applied Mechanics and Engineering 176 (1-4) (1999) 231–246.

[5] M. Ainsworth, J. T. Oden, A posteriori Error Estimation in Finite Element Analysis, John Wiley & Sons, 2000.

[6] I. Babuška, B. Szabó, I. N. Katz, The p-version of the finite element method, SIAM Journal on Numerical Analysis 8 (3) (1981) 515–545.

[7] M. R. Dorr, The aproximation theory for the p-version of the finite element method, SIAM Journal on Numerical Analysis 21 (6) (1984) 1180–1207.

[8] B. Guo, I. Babuška, The h-p version of the finite element method, Computational Mechanics 1 (1) (1986) 21–41.

[9] J. E. Tarancón, F. J. Fuenmayor, L. Baeza, An a posteriori error estimator for the p- and hp-versions of the finite element method, International Journal for Numerical Methods in Engineering 62 (1) (2005) 1–18.

[10] C. O. Frederick, Y. C. Wong, F. W. Edge, Two-dimensional automatic mesh generation for structural analysis, International Journal for Numerical Methods in Engineering 2 (1) (1970) 133–144.

[11] W. C. Thacker, A brief review of techniques for generating irregular computational grids, International Journal for Numerical Methods in Engineering 15 (9) (1980) 1335–1341.

[12] M. A. Yerry, M. S. Shephard, Automatic three-dimensional mesh generation by the modified-octree technique, International Journal for Numerical Methods in Engineering 20 (11) (1984) 1965–1990.

[13] D. Meagher, Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer, Tech. Rep. IPL-TR-80-11 I, Rensselaer Polytechnic Institute (1980).

[14] C. L. Jackins, S. L. Tanimoto, Oct-tree and their use in representing three-dimensional objects, Computer Graphics and Image Processing 14 (3) (1980) 249–270.

[15] L. J. Doctor, J. G. Torborg, Display techniques for octree-encoded objects, IEEE Comput. Graph. Appl. 1 (3) (1981) 29–38.

[16] M. S. Shephard, W. J. Schroeder, A combined Octree/Delaunay method for fully automatic 3D mesh generation, International Journal for Numerical Methods in Engineering 29 (1) (1990) 37–55.

[17] M. S. Shephard, M. K. Georges, Automatic three-dimensional mesh generation by the finite octree technique, International Journal for Numerical Methods in Engineering 32 (4) (1991) 709–749.

[18] C. S. Peskin, Numerical Analysis of Blood Flow in the Heart, Journal of Computational Physics 25 (1977) 220–252.

[19] L. Zhang, A. Gerstenberger, X. Wang, W. K. Liu, Immersed Finite Element Method, Computer Methods in Applied Mechanics and Engineering 293 (21) (2004) 2051–2067.

[20] J. Parvizian, A. Düster, E. Rank, Finite Cell Method: h- and p- Extension for Embedded Domain Methods in Solid Mechanics, Computational Mechanics 41 (1) (2007) 121–133.

[21] A. Düster, J. Parvizian, Z. Yang, E. Rank, The finite cell method for three-dimensional problems of solid mechanics, Computer Methods in Applied Mechanics and Engineering 197 (45-48) (2008) 3768–3782.

[22] D. Schillinger, M. Ruess, The finite cell method: A review in the context of higher-order structural analysis of cad and image-based geometric models, Archives of Computational Methods in Engineering 22 (3) (2015) 391–455.

[23] J. Haslinger, D. Jedelsky, Genetic algorithms and fictitious domain based approaches in shape optimization, Struc. Optim. 12 (1996) 257–264.

[24] K. Kunisch, G. Peichl, Numerical gradients for shape optimization based on embedding domain techniques, Comput. Optim. 18 (1996) 95–114.

[25] W. K. Liu, Y. Liu, D. Darell, L. Zhang, X. S. Wang, Y. Fukui, N. Patankar, Y. Zhang, C. Bajaj, J. Lee, J. Hong, X. Chen, H. Hsu, Immersed Finite Element Method and its Applications to Biological Systems, Computer Methods in Applied Mechanics and Engineering 195 (13) (2006) 1722–1749.

[26] W. K. Liu, S. Tang, Mathematical Foundations of the Immersed Finite Element Method, Computational Mechanics 39 (3) (2007) 211–222.

[27] A. J. Gil, A. Arranz-Carreño, J. Bonet, O. Hassan, The Immersed Structural Potential Method for Haemodynamic Applications, Journal of Computational Physics 229 (22) (2010) 8613–8641.

[28] E. Nadal, J. J. Ródenas, J. Albelda, M. Tur, J. E. Tarancón, F. J. Fuenmayor, Efficient Finite Element Methodology based on Cartesian Grids: Application to Structural Shape Optimization, Abstract and Applied Analysis 2013.

[29] E. Nadal, Cartesian Grid FEM (cgFEM): High Performance h-adaptive FE Analysis with Efficient Error Control. Application to Structural Shape Optimization, PhD Thesis.

[30] O. Marco, R. Sevilla, Y. Zhang, J. J. Ródenas, M. Tur, Exact 3D boundary representation in finite element analysis based on Cartesian grids independent of the geometry, International Journal for Numerical Methods in Engineering 103 (2015) 445–468.

[31] L. Piegl, W. Tiller, The NURBS Book, Springer-Verlag, 1995.

[32] D. F. Rogers, An Introduction to NURBS: with Historical Perspective, Elsevier, 2001.

[33] T. W. Sederberg, J. Zheng, A. Bakenov, A. Nasri, T-splines and T-NURCCs, ACM Transactions on Graphics (TOG) 22 (3) (2003) 477–484.

[34] R. Sevilla, S. Fernández-Méndez, A. Huerta, NURBS-enhanced Finite Element Method (NEFEM): A Seamless Bridge Between CAD and FEM, Archives of Computational Methods in Engineering 18 (4) (2011) 441–484.

[35] R. Sevilla, S. Fernández-Méndez, A. Huerta, 3D-NURBS-enhanced Finite Element Method (NEFEM), International Journal for Numerical Methods in Engineering 88 (2) (2011) 103–125.

[36] L. Kudela, N. Zander, S. Kollmannsberger, E. Rank, Smart octrees: Accurately integrating discontinuous functions in 3d, Computer Methods in Applied Mechanics and Engineering 306 (1) (2016) 406–426.

[37] T.-P. Fries, S. Omerović, Higher-order accurate integration of implicit geometries, International Journal for Numerical Methods in Engineering 106 (5) (2016) 323–371.

[38] R. Sevilla, S. Fernández-Méndez, A. Huerta, Comparison of High-order Curved Finite Elements, International Journal for Numerical Methods in Engineering 87 (8) (2011) 719–734.

[39] M. Tur, J. Albelda, O. Marco, J. J. Ródenas, Stabilized Method to Impose Dirichlet Boundary Conditions using a Smooth Stress Field, Computer Methods in Applied Mechanics and Engineering 296 (2015) 352–375.

[40] J. T. Kajiya, Ray Tracing Parametric Patches, SIGGRAPH Comput. Graph. 16 (3) (1982) 245–254.

[41] D. L. Toth, On Ray Tracing Parametric Surfaces, SIGGRAPH Comput. Graph. 19 (3) (1985) 171–179.

[42] M. Sweeney, R. Bartels, Ray tracing free-form b-spline surfaces, IEEE Computer Graphics and Applications 6 (2) (1986) 41–49.

[43] T. Nishita, T. W. Sederberg, M. Kakimoto, Ray Tracing Trimmed Rational Surface Patches, SIGGRAPH Comput. Graph. 24 (4) (1990) 337–345.

[44] W. Barth, W. Stürzlinger, Efficient ray tracing for Bezier and B-spline surfaces, Computers & Graphics 17 (4) (1993) 423–430.

[45] W. E. Lorensen, H. E. Cline, Marching Cubes: A High Resolution 3D Surface Construction Algorithm, ACM SIGGRAPH Computer Graphics 21 (4) (1987) 163–169.

[46] J. F. Abel, M. S. Shephard, An algorithm for multipoint constraints in finite element analysis, International Journal for Numerical Methods in Engineering 14 (3) (1979) 464–467.

[47] C. Farhat, C. Lacour, D. Rixen, Incorporation of linear multipoint constraints in substructure based iterative solvers. Part 1: a numerically scalable algorithm, International Journal for Numerical Methods in Engineering 43 (6) (1998) 997–1016.

[48] M. Ainsworth, J. T. Oden, A posteriori Error Estimation in Finite Element Analysis, Vol. 142, John Wiley & Sons, Chichester, 2000.

[49] P. Díez, N. Parés, A. Huerta, Recovering lower bounds of the error by postprocessing implicit residual a posteriori error estimates, International Journal for Numerical Methods in Engineering 56 (10) (2003) 1465–1488.

[50] T. Gerasimov, M. Rüter, E. Stein, An explicit residual-type error estimator for Q 1 -quadrilateral extended finite element method in two-dimensional linear elastic fracture mechanics, International Journal for Numerical Methods in Engineering 90 (April) (2012) 1118–1155.

[51] P. Ladevèze, D. Leguillon, Error estimate procedure in the finite element method and applications, SIAM Journal on Numerical Analysis 20 (3) (1983) 485–509.

[52] O. J. B. Almeida Pereirea, J. P. Moitinho de Almeida, E. A. W. Maunder, Adaptive methods for hybrid equilibrium finite element models, Computational Methods in Applied Mechanics and Engineering 176 (1999) 19–39.

[53] O. J. B. Almeida Pereira, J. P. Moitinho de Almeida, A posteriori error estimation for equilibrium finite elements in elastostatic problems, Computer Assisted Mechanics and Engineering Sciences 8 (2-3) (2001) 439–453.

[54] O. C. Zienkiewicz, J. Z. Zhu, A Simple Error Estimator and Adaptive Procedure for Practical Engineering Analysis, International Journal for Numerical Methods in Engineering 24 (2) (1987) 337–357.

[55] O. C. Zienkiewicz, J. Z. Zhu, The superconvergent patch recovery and a posteriori error estimates. Part 1: The recovery technique, International Journal for Numerical Methods in Engineering 33 (7) (1992) 1331–1364.

[56] O. C. Zienkiewicz, J. Z. Zhu, The superconvergent patch recovery and a posteriori error estimates. Part 2: Error estimates and adaptivity, International Journal for Numerical Methods in Engineering 33 (7) (1992) 1365–1382.

[57] J. J. Rodenas, M. Tur, F. J. Fuenmayor, A. Vercher, Improvement of the superconvergent patch recovery technique by the use of constraint equations: the SPR-C technique, International Journal for Numerical Methods in Engineering 70 (6) (2007) 705–727.

[58] F. J. Fuenmayor, J. L. Oliver, Criteria to achieve nearly optimal meshes in the h-adaptive finite element mehod, International Journal for Numerical Methods in Engineering 39 (23) (1996) 4039–4061.

[59] O. C. Zienkiewicz, R. L. Taylor, J. Z. Zhu (Eds.), The Finite Element Method: Its Basis and Fundamentals, Butterworth-Heinemann, Oxford, 2013.

[60] I. Babuška, B. Szabó, On the rates of convergence of the finite element method, International Journal for Numerical Methods in Engineering 18 (3) (1982) 323–341.