Exact 3D Boundary Representation in Finite Element Analysis Based on Cartesian Grids Independent of the Geometry

Onofre Marco[1], Ruben Sevilla[2], Yongjie Zhang[3], Juan José Ródenas[1], Manuel Tur[1]

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

[2]*Civil and Computational Engineering Centre (C^{2}EC), College of Engineering, Swansea University, Singleton Park, Swansea, SA2 8PP, Wales, UK.*

[3]*Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA.*

Abstract

This paper proposes a novel Immersed Boundary Method where the embedded domain is exactly described by using its CAD boundary representation with NURBS or T-Splines. The common feature with other immersed methods is that the current approach substantially reduces the burden of mesh generation. In contrast, the exact boundary representation of the embedded domain allows to overcome the major drawback of existing immersed methods that is the inaccurate representation of the physical domain. A novel approach to perform the numerical integration in the region of the cut elements that is internal to the physical domain is presented and its accuracy and performance evaluated using numerical tests. The applicability, performance and optimal convergence of the proposed methodology is assessed by using numerical examples in three dimensions. It is also shown that the accuracy of the proposed methodology is independent on the CAD technology used to describe the geometry of the embedded domain.

Email addresses: `onmaral@upvnet.upv.es` (Onofre Marco[1]), `R.Sevilla@swansea.ac.uk` (Ruben Sevilla[2]), `jessicaz@andrew.cmu.edu` (Yongjie Zhang[3]), `jjrodena@mcm.upv.es` (Juan José Ródenas[1]), `matuva@mcm.upv.es` (Manuel Tur[1])

Keywords: Immersed Boundary Methods, Cartesian grids, NURBS, T-spline, Bézier extraction, NEFEM

1. Introduction

In the Finite Element Method (FEM) the domain, usually defined by a Computer-Aided Design (CAD) model, where the problem is actually solved, is partitioned in subdomains, or elements, of simple geometries (e.g., triangles or quadrilaterals in 2D and tetrahedra, hexahedra, prisms or pyramids in 3D). Despite mesh generation using simplexes (i.e., triangles and tetrahedra) is considered a mature technology, the generation of a fitted mesh for complex geometries with good-quality elements to avoid numerical errors due to presence of highly distorted elements, still requires a substantial effort. In addition, the computational mesh must be adapted to properly capture the local features of the solution such as stress concentrations in solid mechanics or boundary layers in fluid mechanics.

According to some studies, the process of creating an analysis-suitable geometry and the meshing of that geometry appropriate for Finite Element Analysis (FEA) takes 80% of the total time required to perform a finite element simulation. The bibliography about this subject shows several ways to decrease this 80%. Among them we can cite the Isogeometric Analysis (IGA) [?] and the techniques where the mesh is made independent of the geometry of the domain to be analyzed.

NURBS (Non-Uniform Rational B-Splines) are ubiquitous in CAD and have been successfully used as a basis for IGA where, instead of polynomials, the FE interpolation functions are those used to define the geometry. This new concept seeks to reduce errors by focusing on one, and only one, geometric model, which can be utilized directly as the analysis model. A newer CAD representation tool, the T-splines [?], which allows for the use of the so called T-junctions, ensures the possibility to create water-tight models (this was not always possible with a NURBS representation of the surface) and has helped to overcome the difficulties of the IGA to produce local refinements. IGA does not only require the NURBS discretization of the surface given by the CAD modeler but also a NURBS/T-spline analysis-suitable discretization of the volume. Progresses towards the automatic generation of this discretization can be found in [1? ? ? ? ?].

The second option analyzed in this paper to decrease the above mentioned analysis time, while maintaining the overall FEA environment, is to use a computational mesh that is completely independent of the geometry of the domain. This option is particularly attractive, for example, in an optimization framework, when the analysis requires continuous mesh adaptations and re-meshings.

The eXtended FEM (XFEM) [?] and the Generalized FEM (GFEM) [?] are two variations of the traditional FEM that reduce the burden of mesh generation. The main motivation of the XFEM was to deal with cracks without the need of re-meshing even if the cracks grow. Making use of the Partition of Unity Method (PUM) [2], this approach enriches the numerical solution to represent singular stress fields near the crack tip and discontinuities on the crack faces. The GFEM follows a similar rationale also using the PUM to include enrichment functions that describe the known behavior of the solution at specific locations. In both methods the mesh can be independent of the geometry, although, for integration purposes only, a boundary-fitted mesh, obtained by additional subdivision of elements cut by the boundary, has to be created so that the numerical integration considers the region of the element that actually lies within the domain.

Other variations of the FEM that were developed to reduce the burden of mesh generation are based on the idea of defining an auxiliary and easy to mesh domain Ω which embeds the problem domain $\Omega_{\texttt{Phys}}$. All these method were classified under the umbrella term of Finite Elements in Ambient Space in [?]. Examples of these analysis techniques are the Immersed Boundary Method (IBM) and the Immersed Finite Element Method (IFEM). The IBM was introduced by Peskin [?] to alleviate the cost associated with remeshing in body-fitted techniques when simulating the flow around heart valves. Later developments including the IFEM [?] were proposed in order to avoid the limitations associated to the assumption of the fiber (i.e., one-dimensional) nature of the immersed structure. Immersed boundary methods, often referred to as embedded methods, have been object of intensive research within the fluid mechanics community and several alternatives and modifications to the original method have been proposed, see [?] for a review. These methods have become very popular in the last decade within the computational bio-mechanics community, see for instance [? 3?]. As in the case of XFEM and GFEM, to numerically compute the integrals appearing in the weak formulation, these techniques rely on a submesh of the elements cut by the boundary that is used to perform the integration in the interior to the physical domain, $\Omega_{\mathtt{Phys}}$. Therefore these elements require a specific treatment.

As previously indicated, the geometry of the domain to be analyzed is usually defined by a CAD model. The accuracy of the geometric representation in FE computation is another issue to take into account which in the late 1990s motivated the incorporation of powerful CAD techniques into FE computations [4]. There is a big concern in the scientific community about the need to integrate CAD systems with the numerical analysis tools. Any attempt towards this integration will require the numerical analysis tools to be able to use the most modern techniques used in the CAD industry. Because of this, methods able to incorporate the most extended CAD technology, namely NURBS and more recently T-Splines, into the FE analysis stage such as IGA methods [5?] or the NURBS-enhanced Finite Element Method (NEFEM) [6? ?] have become very popular.

When the IBM-type methods are applied to complex geometries, it is common to substitute the exact geometry of the embedded domain by an approximated description using a faceted representation in three dimensions. The errors introduced by an approximated geometry representation can be termed as geometrical modeling errors that will translate into numerical integration errors that will negatively influence the accuracy of the numerical analysis because the submesh employed for integration purposes is directly constructed using the approximated embedded geometry. The importance of the geometrical model in body-fitted FE simulations has been pointed out by several authors, see [6] and references therein, but it has been rarely accounted for in immersed techniques until very recently [? ?]. In this work, the CAD description of the boundary of the physical domain is considered.

In addition, new strategies for treating boundaries and interfaces have been developed recently. Within the scope of the IGA, a two dimensional NURBS-based IGA with trimming technique [7] in which the auxiliary domain Ω is defined as a NURBS parametric space has been proposed. Another interesting approach is the Finite Cell Method (FCM) [8] which uses the p-version of the FEM to perform adaptive analysis over a mesh of regular quadrilaterals (2D) or cubes (3D). One of the main features of the FCM is that, for integration purposes, it uses a highly refined integration mesh into each of the elements cut by the boundary to appropriately capture the limits of the domain, hence, the resolution of the boundary is related to the refined integration mesh. In exchange, our approach will consist of using high-order quadratures over the integration subdomains of the coarse mesh to capture the boundary of the problem.

The *Cartesian grid Finite Element Method* (cgFEM) presented by Nadal *et al.* [? ?] is a computationally efficient FE methodology for the resolution of 2D linear elasticity problems that makes use of a Cartesian grid in which the problem domain is embedded. A hierarchical data structure relates the different refinement levels of the Cartesian grid allowing for the definition of h-refined meshes for h-adaptive analysis and for the simple data transfer and re-utilization between elements of different refinement levels. The Superconvergent Patch Recovery technique for displacements (SPR-CD) uses constrain equations to obtain accurate recovered displacement and stress fields that locally satisfies the equilibrium equations and the Dirichlet boundary conditions. These fields are used as the standard output of cgFEM instead of the raw FE solution and as part of the information required by the Zienkiewick-Zhu error estimator [?] that drives the h-adaptive refinement process. Dirichlet boundary conditions are imposed using a stabilized Lagrange multipliers approach [9], where the stabilization term is provided by the FE tractions along the Dirichlet boundaries evaluated in a previous mesh. In cgFEM, a procedure, only valid for the 2D case, based on the use of transifinite mapping functions can be used in the elements cut by the boundary in order to consider the exact geometry of the domain in the evaluation of the required volume integrals. This avoids integration errors due to an inaccurate representation of the domain that could even lead to an error convergence rate of the FE solution smaller than the expected theoretical optimum.

An extension of cgFEM to 3D, called FEAVox, is under development. One of the most challenging aspects of the development of FEAVox is to consider the exact boundary of the domain in the evaluation of volume integrals. Therefore, this paper presents a methodology that incorporates the exact boundary representation of the 3D computational domain $\Omega_{\texttt{Phys}}$ embedded in the domain Ω meshed with a Cartesian grid composed of regular hexahedra. Instead of simplifying the embedded geometry to perform the numerical integration, we propose efficient techniques to perform the numerical integration over the true computational domain $\Omega_{\texttt{Phys}}$. The proposed technique follows the NEFEM rationale although new developments are needed in order to find the intersections between the Cartesian grid and the boundary of the physical domain. In addition, this paper considers not only NURBS but also T-Splines.

The paper is organized as follows: A brief review of NURBS and T-spline representations will be shown in Section 2, then Section 3 will be devoted to explain how to capture exact geometries within a Cartesian grid framework. Section 4 will present the formulation of the problem the procedure used to impose Dirichlet boundary conditions considering meshes not conforming to the geometry. Numerical results showing the behavior of the proposed technique will be presented in Section 5. This contribution ends with the conclusions in Section 6.

2. Geometrical Representation: from NURBS to T-spline

Different options are available for representing surfaces in CAD such as B-Splines [?], NURBS [10, 11], subdivision surfaces [?] or T-Spline [?]. In this paper, we consider NURBS and T-splines for the geometrical representation of three dimensional models. This section covers succinctly the main features of these two technologies.

2.1. NURBS fundamentals

NURBS are a generalization of B-splines, which in turn are piecewise polynomial curves composed of B-spline basis functions. The basis functions are defined in parametric space on a so-called knot vector Ξ. This is a set of non-decreasing real numbers representing coordinates (knots) in the parametric space:

$$\Xi = \{\xi_1, \dots, \xi_{n+p+1}\}, \tag{1}$$

where p is the order of the B-spline and n the number of basis functions. The interval $[\xi_1, \xi_{n+p+1}]$ is called a patch, whereas the interval $[\xi_1, \xi_{i+1})$ is called a knot span. A knot vector is said to be uniform if its knots are uniformly spaced and non-uniform otherwise. Moreover, it is said to be open if its first and last knots are repeated $p+1$ times.

The B-spline basis functions $N_i^{(p)}(\xi)$ of order $p \geq 0$ are defined recursively on the corresponding knot vector as follows:

$$N_i^{(0)}(\xi) = \begin{cases} 1 & \xi_i \leq \xi \leq \xi_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$N_i^{(q)}(\xi) = \frac{(\xi - \xi_i)\, N_i^{(q-1)}(\xi)}{\xi_{i+q} - \xi_i} + \frac{(\xi_{i+q+1} - \xi)\, N_{\xi+1}^{(q-1)}(\xi)}{\xi_{i+q+1} - \xi_{i+1}} \tag{3}$$

for $q = 1, \dots, p$ and with $i = 1, \dots, n + p + 1$. They are $\mathcal{C}^{p-1}$-continuous if internal knots are not repeated. If a knot has multiplicity k, the basis is $\mathcal{C}^{p-k}$-continuous at that knot. Further properties of the basis functions are

- B-spline basis functions formed from open knot vectors constitute a partition of unity, that is,
  $\sum_{i=1}^{n} N_i^{(p)}(\xi) = 1 \;\forall\, \xi$.
- The support of each $N_i^{(p)}(\xi)$ is compact and contained in the interval $[\xi_1, \xi_{i+p+1})$.
- B-spline basis functions are non-negative: $N_i^{(p)}(\xi) \geq 0 \;\forall\, \xi$.

B-spline curves of order p are linear combinations of B-spline basis functions of order p, $N_i^{(p)}$, and of points $\mathbf{P}_i$. These points, $\mathbf{P}_i$, referred to as control points, are given in d-dimensional space $\mathbb{R}^d$. E.g. in three dimensions this means $\mathbf{P}_i = (x_i, y_i, z_i)^T$. Hence B-splines are given as:

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} N_i^{(p)}(\xi)\, \mathbf{P}_i \tag{4}$$

The control points define the control polygon. B-spline curves interpolate the control points just at their start and end points. In between, interpolation can be achieved by a certain multiplicity of control points or knots, respectively.

NURBS are rational B-spline curves which are the projection of a non-rational B-spline curve $C^w(\xi)$, defined in $(d+1)$-dimensional homogeneous coordinate space, back onto the d-dimensional physical space $\mathbb{R}^d$. Homogeneous (weighted) $(d+1)$-dimensional control points are

$$\mathbf{P}_i^w = (w_i x_i, w_i y_i, w_i z_i, w_i)^T \tag{5}$$

The non-rational $(d+1)$-dimensional B-spline curve $\mathbf{C}^w$ then reads

$$\mathbf{C}^w(\xi) = \sum_{i=1}^{n} N_i^{(p)}(\xi)\, \mathbf{P}_i^w \tag{6}$$

Projecting onto $\mathbb{R}^d$ by dividing through the additional coordinate yields the rational B-spline curve

$$\mathbf{C}(\xi) = \frac{\sum_{i=1}^{n} N_i^{(p)}(\xi)\, w_i \mathbf{P}_i}{\sum_{i=1}^{n} w_i N_i^{(p)}(\xi)} = \sum_{i=1}^{n} R_i^{(p)}(\xi)\, \mathbf{P}_i \tag{7}$$

Here $\mathbf{P}_i$ are the control points in $\mathbb{R}^d$, $R_i^{(p)}$ are rational B-spline basis functions and w_i is referred to as the i-th weight, typically $w_i \geq 0 \quad \forall i$. Through this projection all common shapes, especially conic sections such as circles and ellipses, can be represented exactly.

Finally, NURBS surfaces are constructed from a tensor product through two knot vectors $\Xi = \{\xi_1, \ldots, \xi_{n+p+1}\}$ and $\Gamma = \{\eta_1, \ldots, \eta_{m+q+1}\}$. The $n \times m$ control points $\mathbf{P}_{i,j}$ form a control net. For the geometric description of NURBS surfaces the typical arrangement is a $(n \times m)$-dimensional matrix with elements (i, j). The NURBS surface $\mathbf{S}(\xi, \eta)$ is defined on the one-dimensional basis functions $N_i^{(p)}$ and $M_i^{(q)}$ (with $i = 1, \ldots, n$ and $j = 1, \ldots, m$) of order p and q, respectively, as

$$\mathbf{S}(\xi, \eta) = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{N_i^{(p)}(\xi)\, M_j^{(q)}(\eta)\, w_{i,j}}{\sum_{i=1}^{n} \sum_{j=1}^{m} N_i^{(p)}(\xi)\, M_j^{(q)}(\eta)\, w_{i,j}} \mathbf{P}_{i,j} \tag{8}$$

In the case of surfaces, we refer to the $[\xi_1, \xi_{n+p+1}] \times [\eta_1, \eta_{m+q+1}]$ as patch and $[\xi_i, \xi_{i+1}) \times [\eta_j, \eta_{j+1})$ as knot span. NURBS surfaces examples are shown in Figure 1a and Figure 1b, where in blue we can see the control points and in red the projections of the knot vectors onto the surface. These models will be analyzed in the section devoted to numerical comparisons.

NURBS have been used as a basis for IGA where the interpolation functions are those used to define the geometry. This approach brought to light some drawbacks of NURBS surfaces due to its tensor product nature. For instance, to model complicated designs requires multiple NURBS patches, which are often discontinuous across patch boundaries. Even achieving $\mathcal{C}^0$ continuity across patches requires special techniques. The joining of two patches that were created separately may require the insertion of many knots and nonlinear reparameterization of one or both patches. Furthermore, all NURBS refinement operations are global. In other words, when we refine by inserting knots into the knot vectors of a surface, the knot lines extend throughout the entire domain. Global refinement introduces an unnecessary cost when NURBS as used as basis for the analysis. Finally, to add features, such as holes it is common to use trimming curves. The application of trimming curves destroys the tensor product nature of the geometry thus the geometric basis no longer describes the geometry and cannot be used directly in FE analysis.

2.2. T-spline fundamentals

To achieve a tight integration of design and analysis requires a technology built on the smooth B-spline basis which can be locally refined and is capable of representing domains of arbitrary topological complexity as a single watertight geometry. All of these capabilities are present in a generalization of NURBS called T-splines. In this work we are not interested in the characteristics of this technology from the IGA point of view but in its representation power as a state-of-the-art technology.

T-spline basis functions are defined on local knot vectors, and its control nets allow T-junctions which are introduced during local refinement. T-spline does not have the superfluous control points in NURBS model to satisfy topological constrains. A T-spline surface is defined as

$$\mathbf{S}(\xi,\eta) = \frac{\sum_{i=0}^{n} B_i(\xi,\eta) w_i \mathbf{T}_i}{\sum_{i=0}^{n} w_i B_i(\xi,\eta)}, \qquad (\xi,\eta) \in \Omega, \tag{9}$$

where

$$B_i(\xi,\eta) = N_i^{\xi}(\xi) N_i^{\eta}(\eta) \tag{10}$$

and $\mathbf{T}_i$ are the T-spline control points, w_i are the respective weights, $N_i^{\xi}(\xi)$ and $N_i^{\eta}(\eta)$ are B-spline basis functions defined by two local knot vectors ξ_i and η_i. If the degree is 3, we have $\xi_i = [\xi_{i_0}, \xi_{i_1}, \xi_{i_2}, \xi_{i_3}, \xi_{i_4}]$ and $\eta_i = [\eta_{i_0}, \eta_{i_1}, \eta_{i_2}, \eta_{i_3}, \eta_{i_4}]$ [**? ?**]. The algorithm used to infer knot vectors from a T-mesh is introduced in [**?**].

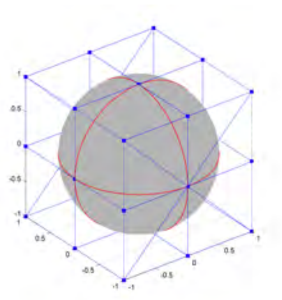
Neither NURBS nor T-spline can be directly used for analysis, since they are defined to represent the whole domain. To obtain the discretized finite element representation of a NURBS or T-spline, we can use Bézier extraction operator to decompose the domain into Bézier elements. The Bézier extraction operator maps a piecewise Bernstein polynominal basis onto a B-spline basis [12]. A Bézier extraction operator $\mathbf{E}$ is a linear operator such that

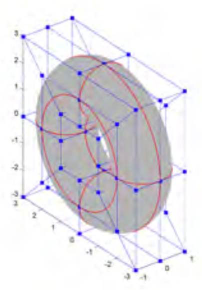
$$N(s) = \mathbf{E}B(s) \tag{11}$$

where $N(s)$ is a B-spline basis function and $B(s)$ is a set of Bézier basis functions. The operator $\mathbf{E}$ is constructed from the repeated knot insertion of knot vector which defines $N(s)$ and it is independent from the control points and the basis functions. A similar extraction operator $\mathbf{M}$ can be defined to transform T-spline basis functions to Bézier basis functions [**?**]. In a T-spline framework, for each parametric domain which can extract one Bézier element, we can first find all the control points with nonzero basis functions. Then we have

$$B_t^e = \mathbf{M}^e B_b^e \tag{12}$$

where B_t^e is the vector formed by all the T-spline basis functions with nonzero function values, and B_b^e is the vector formed by the Bézier basis functions. For each Bézier element, $\mathbf{M}^e$ can be calculated using the Oslo knot insertion algorithm [13]. This algorithm can obtain the extraction operator from all the related T-spline basis functions to the Bézier basis functions in a single step. In this work we will use this by-product to be able to represent T-spline geometries, see Figure 1c.


(a) NURBS sphere model.


(b) NURBS torus model.

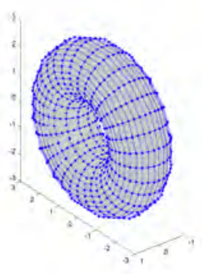
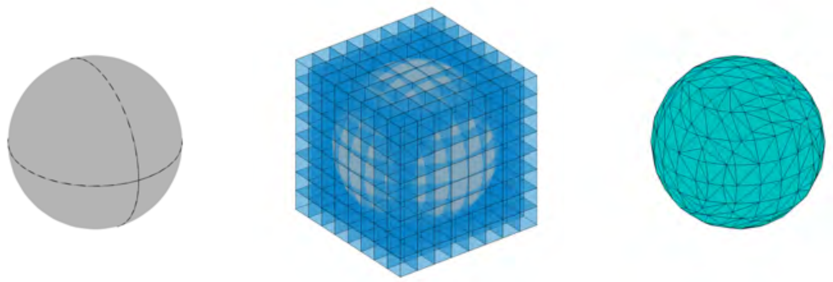

(c) T-spline torus model.

Figure 1: Geometrical models used in this contribution.

3. Cartesian Grids with Exact Representation of the Immersed Geometry

As previously mentioned, in the classical FEM, the most extended approach is to employ unstructured meshes that conform to the boundary of the physical domain. Mesh generation and, especially, mesh adaptation techniques such as mesh refinement, mesh movement or remeshing are time consuming and require a substantial amount of human hours [**?** **?**]. Expertise is required in order to refine the mesh appropriately to accurately represent both the geometry of the physical domain and the local characteristics of the solution of the problem under consideration.

Given an open bounded domain $\Omega_{\mathtt{Phys}} \subset \mathbb{R}^3$, see Figure 2a, with boundary $\Gamma_{\mathtt{IB}} = \partial\Omega_{\mathtt{Phys}}$, the key principle of FEAVox, or any other IBM, consists in



(a) Geometrical model of an sphere, $\Omega_{\texttt{Phys}}$. (b) Cartesian grid, Ω. (c) Approximation of $\Omega_{\texttt{Phys}}$, $\Omega^h_{\texttt{Phys}}$.

Figure 2: Typical Immersed Boundary Method environment.

defining an embedding domain Ω such that $\Omega_{\texttt{Phys}} \subset \Omega$ and with a much more simpler geometry than the physical domain. Therefore, Ω is extremely easy to mesh compared to the domain of interest $\Omega_{\texttt{Phys}}$. In FEAVox, we consider Ω to be a cuboid, and a Cartesian grid is used to mesh the domain Ω as represented in Figure 2b. In order to represent the geometry of the physical domain in IBM, it is common to use a linear triangular mesh to discretize the boundary $\Gamma_{\texttt{IB}}$. This option allows for the implementation of simple algorithms to find the intersections between the discretized immersed boundary and the mesh of the embedding domain, but this also means that the problem is solved not in the physical domain $\Omega_{\texttt{Phys}}$ but in an approximation of $\Omega_{\texttt{Phys}}$, namely $\Omega^h_{\texttt{Phys}}$, Figure 2c. The effect of this approximation can be very important if high order elements are used. This method can be used to obtain good results from an engineering point of view in many problems. However, the optimal convergence rate of the FEM can be compromised because of the rate of convergence of the integration error when the mesh is refined. To overcome this problem, in this work, the CAD description of the boundary of the physical domain $\Gamma_{\texttt{IB}}$ is considered using high-order quadratures over the integration subdomains to capture the boundary of the problem.

The translation of complex CAD-based geometrical models into conforming FE discretizations is computationally expensive, difficult to fully automate and often leads to error-prone meshes, which have to be improved manually by the user. Immersed boundary methods do not require body-fitted meshes, but embed the domain into a Cartesian grid (i.e., a regular grid of axis-aligned elements), which is generated irrespective of the geometric complexity of the physical domain. In this work, we present an strategy that exploits the advantages of Cartesian grids and uses specific strategies for the numerical integration over the exact physical domain, with a CAD boundary representation.

3.1. Generation of the analysis mesh

FEAVox is based on the use of a sequence of uniformly refined Cartesian meshes where hierarchical relations between the different mesh levels have been defined.

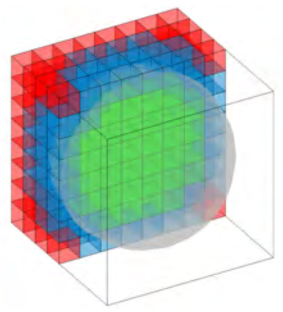
The sequence of m meshes used to discretize the embedding 3D domain Ω is called the *Cartesian grid pile* and is denoted by $\{\mathcal{Q}_h^i\}_{i=1,\ldots,m}$. For each level of refinement, the embedding domain Ω is partitioned in $\mathtt{n}_{\mathtt{el}}^i$ disjoint cubes of uniform size, where $\mathtt{n}_{\mathtt{el}}^{i+1} = 8\mathtt{n}_{\mathtt{el}}^i$. A hierarchical data structure for FE analysis based on element splitting was presented in [?]. This data structure takes into account the hierarchical relations between the elements of different levels, obtained during the element splitting process, to accelerate FE computations. The data structure has been adapted to the particular case of a sequence of meshes given by the Cartesian grid pile, where all elements are geometrically similar to the element used in the coarsest level of the Cartesian grid pile, called the reference element. One important benefit of the data structure adopted here is that the mapping between an element in the Cartesian grid and the reference element is affine and, therefore, its Jacobian is constant. This property can be exploited to dramatically speed up the computation of the elemental matrices. For instance, the analysis presented in [?] shows that the number of operations required to compute the elemental matrices can be reduced by a factor of 10 when a mapping with a constant Jacobian is considered with low-order hexahedral elements. This and other hierarchical relations considered in the data structure allow for a simplification of the mesh refinement process and the precomputation of most of the information used by the FE implementation, remarkably influencing the efficiency of the code. The implementation uses functions that directly provide the nodal coordinates, mesh topology, hierarchical relations, neighborhood relations, and other geometric information, in an efficient manner, when required. Therefore, there is no need to store this information in memory, making the proposed algorithm more efficient, not only in terms of computing time but also in terms of memory requirements.

3.2. Element classification and geometry-mesh intersection

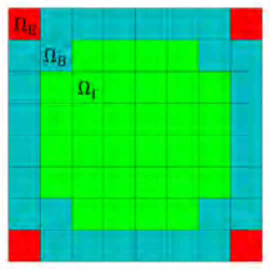
The first step of the proposed strategy consists of creating the analysis mesh used to obtain the FE solution of the boundary value problem. In order to obtain the analysis mesh, the elements of the Cartesian grid are classified as:

- Boundary elements: elements cut by the boundary of the physical domain, this is, elements $\Omega_{\mathtt{B}}$ such that $\Omega_{\mathtt{B}} \cap \Gamma_{\mathtt{IB}} \neq \emptyset$.
- Internal elements: elements inside the physical domain, thus, elements $\Omega_{\mathtt{I}}$ such that $\Omega_{\mathtt{I}} \subset \Omega_{\mathtt{Phys}}$, and
- External elements: elements outside the physical domain, elements $\Omega_{\mathtt{E}}$ such that $\Omega_{\mathtt{E}} \subset \Omega \setminus \Omega_{\mathtt{Phys}}$,

as illustrated in Figure 3 where a sphere is embedded in a Cartesian grid, Figure 2b.



(a) Perspective view.



(b) 2D section.

Figure 3: Section of a three dimensional Cartesian grid showing the three different types of elements: (1) In red, external elements, $\Omega_{\mathtt{E}}$, not considered in the analysis, (2) in blue, elements, $\Omega_{\mathtt{B}}$, intersected by the boundary of the embedded domain and (3) in green interior elements, $\Omega_{\mathtt{I}}$.

The analysis mesh is formed by the internal and the boundary elements intersected by the geometry. The external elements are not considered in the analysis stage. Internal elements are treated as standard FE elements and the affinity with respect to the reference element is exploited in order to speed-up the computational cost of the element matrices. For those elements cut by the boundary of the physical domain, and since we are working with meshes completely independent of the embedded geometry, it is necessary to determine the relative position of the elements with respect to the physical boundary, so specific strategies are required to find the intersection with the boundary and to perform the numerical integration. Efficient strategies to perform these two operations are proposed in the remaining of this section.

The strategy considered in this work to classify the elements consists of three steps:

1. Find the intersections of the physical boundary with the edges of the Cartesian grid elements,
2. Classify the grid nodes as internal or external, relative to the physical domain, and
3. Classify the elements as internal, boundary or external.

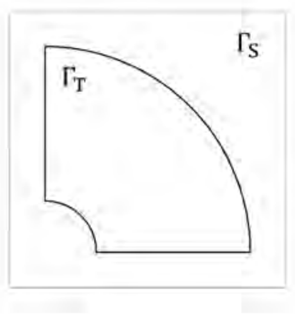
It is therefore only necessary to describe in detail the proposed strategy to compute the intersections of the physical boundary with the edges of the Cartesian grid elements There are several methods available in the literature to evaluate the intersection between parametric surfaces and a Cartesian grid. Most of them were developed for ray-tracing tasks related to the animation industry, where rendering CAD models is mandatory, see [14] for a review of the methods.

We employ a Newton-Raphson algorithm to find the intersections between the edges of the Cartesian grid elements of the analysis mesh and the parametric surfaces describing the boundary of the physical domain. Efficiency is guaranteed by using, as an initial approximation, the intersection of the edges of the Cartesian grid elements with an auxiliary triangular surface mesh of the boundary of the physical domain. We have to remark that this triangular mesh is only an approximation of the exact CAD description of the physical boundary. However, this approximation is only used to compute a good initial guess for the Newton-Raphson algorithm.

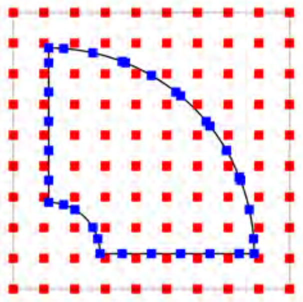
If the embedded domain is represented by trimmed surfaces, we need to create the auxiliary triangulation, used during the Newton-Raphson procedure, only over the trimmed surface of the NURBS. To illustrate this situation we consider the example depicted in Figure 4. Figure 4a shows the parametric space of a NURBS surface $\Gamma_{\mathtt{S}}$. The immersed body $\Gamma_{\mathtt{T}}$ is assumed to be the image of the trimmed space where NURBS will be used to define the boundary curves (trimming curves). We will define a triangulation of $\Gamma_{\mathtt{T}}$

as shown in Figure 4d. To generate this triangulation we will use a set of arbitrary points distributed over the parametric space of the NURBS surface (red squares in Figure 4b) but we will also add points located over the boundary curves defining Γ_T, (blue squares in Figure 4b). We have to ensure that the intersections between the Cartesian grid edges and the NURBS are correctly identified as intersections on Γ_T or outside Γ_T. Obviously, for an appropriate representation, the points located over the boundary curves of Γ_T must include the extremes of these curves, but additional points must also be included to properly define the boundary. To do this, the additional points will correspond to the intersections of the trimming curves with all the Cartesian planes that define the faces of the elements of the Cartesian grid. The evaluation of these intersections only introduces a marginal extra computational cost because, although NURBS are rational curves, their homogeneous description, see Section 2.1, is employed in the intersection process. A triangulation of the NURBS surface will be created using this cloud of points and the efficient 2D Delaunay triangulation procedure, see Figure 4c. The triangles lying outside Γ_T will be discarded to obtain the final auxiliary triangulation shown in Figure 4d. This figure also shows the point of intersection between the NURBS and the edges of the elements of the Cartesian grid, correctly classified as internal (red dots) or external (blue dots) with respect to Γ_T.

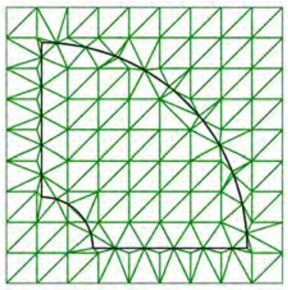
The above procedure requires the definition of an auxiliary triangulation in the parametric space of the NURBS surfaces defining the embedded domain. In order to guarantee convergence of the Newton-Raphson algorithm to the desired intersection point, it is advantageous to define a triangulation with a triangle size related to the size of the elements of the analysis mesh. If the auxiliary triangulation is too coarse, the axis of the Cartesian grid can intersect the same triangle several times. This situation will prevent the convergence of the Newton-Raphson algorithm in some cases as the same initial guess will be considered for the computation of two different roots. This situation is illustrated in Figure 5. In Figure 5a a NURBS surface is represented together with two planes corresponding to sections of the Cartesian grid. Figure 5b shows the parametric space of the NURBS surface with a coarse auxiliary triangulation. The two curves correspond to the intersections of the planes in Figure 5a with the NURBS surface in the parametric space. It can be observed that, for this coarse auxiliary triangulation, the axis of the Cartesian grid can intersect several times the triangle highlighted in light blue. Figure 5c shows a finer triangulation that fixes this issue. To



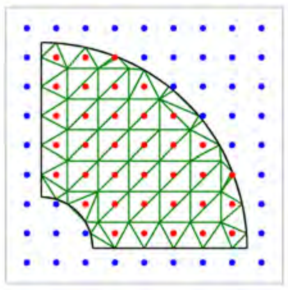
(a) Parametric space of a NURBS surface, Γ_S, and subspace to define a trimmed NURBS, Γ_T.



(b) Points used to define an arbitrary auxiliary triangulation on the parametric space.



(c) Triangulation over the parametric space.

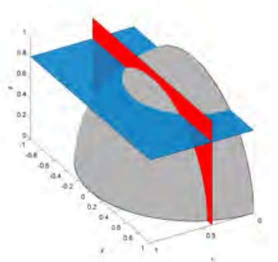


(d) Final auxiliary triangulation that ensures the correct classification of all intersection points.

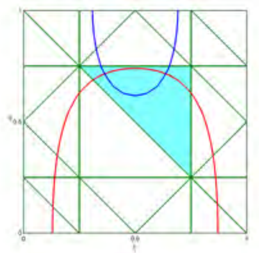
Figure 4: Identification of intersections between a trimmed NURBS surface and the edges of the Cartesian grid.

avoid the problem the size of the triangles has to be selected depending on the refinement of the Cartesian grid.

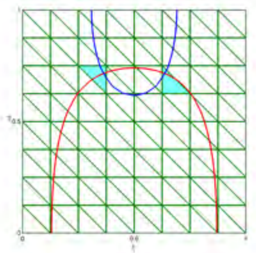
Assuming that the intersections of the physical boundary with the edges of the Cartesian grid elements are computed, it is easy to classify the element nodes as internal or external just marching along the edges of the Cartesian grid. Once the grid nodes are classified, it is straight forward to classify the elements as internal, boundary or external, just by counting the number of internal and external nodes in each element.



(a) NURBS surface in grey and two planes corresponding to sections of the Cartesian grid.



(b) Parametric space of the NURBS with a coarse auxiliary triangulation and the projection of the two planes in Figure 5a.



(c) Parametric space of the NURBS with a refined auxiliary triangulation and the projection of the two planes in Figure 5a.

Figure 5: Automatic definition of the size of the auxiliary triangulation to avoid multiple intersections of a single triangle with the planes of the Cartesian grid.

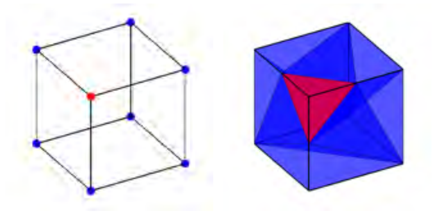
3.3. Integration over subdomains

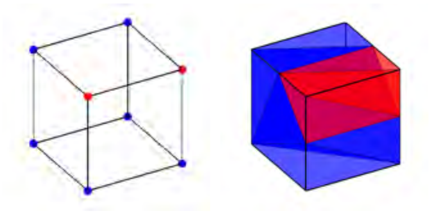
The FEM requires the computation of integrals over the domain of interest. When a body-fitted mesh is employed, the integrals on the domain interior are computed by adding the contribution of the integrals over each element and, analogously, the boundary integrals are computed by adding the contribution of the integrals over each element face on the boundary of the physical domain. The numerical integration in IBM require special attention as the mesh is completely independent of the geometry of the physical domain.

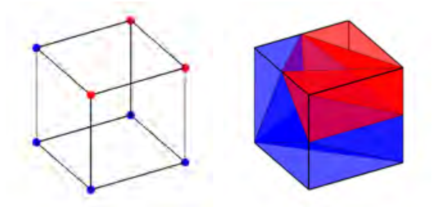
Internal elements are treated as standard finite elements and the integration is performed using a tensor product of one-dimensional Gauss quadratures with the desired number of points in each direction. However, the contribution from the boundary element $\Omega_{\mathtt{B}}$ requires special attention as the integral must be computed only over the portion of the boundary elements that lies inside the physical domain, namely $\Omega_{\mathtt{B}}^{\mathtt{Phys}} = \Omega_{\mathtt{B}} \cap \Omega_{\mathtt{Phys}}$. In fact, the independent generation of the Cartesian grid with respect to the embedded geometry implies that the region of elements intersected by the mesh lying inside the computation domain, $\Omega_{\mathtt{B}}^{\mathtt{Phys}}$ can be extremely complex. The strategy proposed to perform the integration over $\Omega_{\mathtt{B}}^{\mathtt{Phys}}$ consists in employing a tetrahedralization of this region that incorporates the exact boundary representation of $\Omega_{\mathtt{Phys}}$.

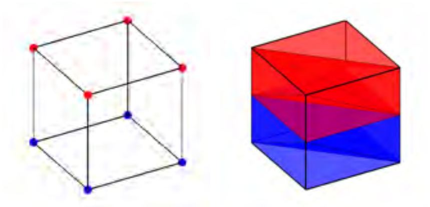
The proposed approach is inspired on the Marching Cubes (MC) algorithm [15], which uses a set of templates for the intersection between surfaces and the edges of cubes. The MC algorithm is widely used in computational graphics to represent approximations of surfaces as it is very efficient sorting out basic intersection patterns and creating linear surfaces between them. We have taken the basic intersection patterns of the MC algorithm to identify the most common intersection patterns between the embedded geometry and the Cartesian grid, then a parametrized tetrahedralization of each one of these patterns is generated and stored. To facilitate the implementation, and without loss of generality, we assume that the Cartesian elements are intersected, at most, once by the boundary of the physical domain. This condition can be easily relaxed and it is employed here only to simplify the presentation and to facilitate the implementation. From this premise, we need only seven out of fourteen templates of the original MC algorithm (1, 2, 5, 8, 9, 11 and 14, see [15]). It is in fact possible to use the remaining templates to identify regions of particular geometric complexity where extra mesh refinement can be introduced to properly capture them. The seven patterns considered are depicted in Figure 6. In the figures we can see the nodal topologies and the set of tetrahedra used for each pattern. Colors identify internal and external subdomains (or different materials if the case of multimaterial problems).

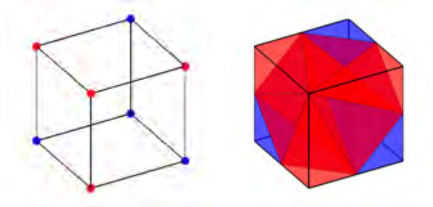
Numerical integration over the region $\Omega_{\mathtt{B}}^{\mathtt{phys}}$ is then accomplished by integrating over each subdomain of the tetrahedralization. In order to perform the integration over the subdomains, the strategy proposed within the NEFEM [**?**] is adopted. This methodology was designed to incorporate the exact boundary of the computational domain into body-fitted FE simulations

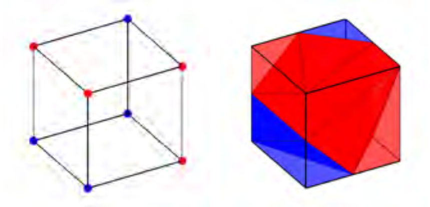

(a) Configuration 1.


(b) Configuration 2.


(c) Configuration 3.


(d) Configuration 4.


(e) Configuration 5.


(f) Configuration 6.

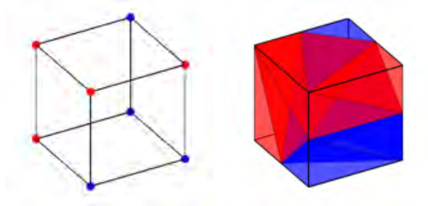

(g) Configuration 7.

Figure 6: Intersection patterns inspired on the MC algorithm. Nodal topology (left) and tetrahedralization (right).

and the advantages with respect to the classical finite element method were demonstrated for a variety of problems, see [16]. A tetrahedral subdomain T_e^F with a face on the physical boundary is parametrized using the mapping

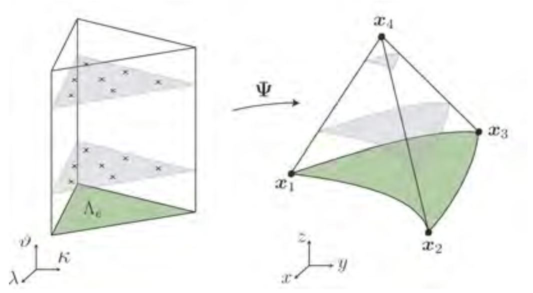


Figure 7: Integration over a curved tetrahedron with a face over the physical domain.

$$\begin{aligned} \mathbf{\Psi} \ &: \Lambda_e \times [0,1] \longrightarrow T_e^F \\ &(\xi,\eta,\zeta) \longmapsto \mathbf{\Psi}(\xi,\eta,\zeta) := (1-\zeta)\mathbf{S}(\xi,\eta) + \zeta\mathbf{x}_4, \end{aligned}$$

where $\mathbf{S}(\Lambda_e)$ denotes the curved face of T_e^F on the boundary of the physical domain and $\mathbf{x}_4$ is the internal vertex of T_e^F. Analogously, a tetrahedral subdomain T_e^E with an edge on the physical boundary is parametrized using the mapping

$$\begin{aligned} \mathbf{\Phi} \ &: [\xi_1,\xi_2] \times [0,1]^2 \longrightarrow T_e^E \\ &(\xi,\eta,\zeta) \longmapsto \mathbf{\Phi}(\xi,\eta,\zeta) := (1-\zeta)(1-\eta)\mathbf{C}(\xi) + (1-\zeta)\eta\mathbf{x}_3 + \zeta\mathbf{x}_4. \end{aligned}$$

where $\mathbf{C}([\xi_1,\xi_2])$ denotes the curved edge of T_e^E on the boundary of the physical domain and $\mathbf{x}_3$ and $\mathbf{x}_4$ are the two internal vertices of T_e^E.

The most salient properties of the mappings used by NEFEM is the ability to decouple the directions of the surface definition, Λ_e and $[\xi_1,\xi_2]$ in the mappings $\mathbf{\Psi}$ and $\mathbf{\Phi}$ respectively, with respect to the interior directions. In addition, the mappings are linear in the interior directions, guaranteeing that the required number of integration points is minimum, compared to other options such as the transfinite mappings [17].

Given these parametrizations, it is possible to perform the numerical integration over all the curved tetrahedral elements that form $\Omega_{\mathtt{B}}^{\mathtt{phys}}$. To this end, we consider tensor products of triangle quadratures [18] and one-dimensional Gaussian quadratures for the tetrahedrons with a face on the boundary of the physical domain, see Figure 7.

For the tetrahedra with an edge on the boundary of the physical domain, tensor products of one-dimensional Gaussian quadratures are employed. The number of integration points required in the parametric space of the parametric boundary representation depends on the CAD technology employed. In [19], the number of integration points required to integrate polynomial functions over domains with a NURBS or B-spline boundary description is studied numerically. The conclusions show that, compared to traditional FE, NEFEM requires the same, or just one integration point more, in order to ensure that the numerical error due to the numerical integration is lower than the interpolation error. In addition, the ideas supporting this approach are valid not only when the boundary of the domain is parametrized by NURBS, but for any piecewise boundary parametrization.

Remark 1. *It is important to note that, when non trimmed surfaces are considered, NEFEM defines the triangular face in the parametric space, Λ_e in Figure 7 as a stright-sided triangle [19]. This is always possible due to the boundary fitted nature of the NEFEM approach. In contrast, the approach proposed in this paper requires the faces of the tetrahedral elements on the boundary of the embedded domain are defined as the anti-image (by the NURBS surface) of the intersections of Cartesian planes and the NURBS surfaces. Therefore, in general these faces are curved triangles in the parametric space of the NURBS. It is worth noting that the mapping depicted in Figure 7 is still valid to perform the numerical integration, even if the boundary face is a curved triangle in the parametric space.*

4. Problem Formulation and Numerical Solution

Let us consider an open bounded domain $\Omega_{\texttt{Phys}} \subset \mathbb{R}^3$ with closed boundary $\Gamma_{\texttt{IB}} = \partial\Omega_{\texttt{Phys}}$. The boundary of the domain is partitioned into the Neumann boundary Γ_N and the Dirichlet boundary Γ_D, with $\Gamma_{\texttt{IB}} = \Gamma_N \cup \Gamma_D$ and $\Gamma_N \cap \Gamma_D = \emptyset$. The strong form of the equilibrium equations and the boundary conditions are

$$\begin{aligned} -\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}) = \mathbf{b} \quad & \text{in} \quad \Omega_{\texttt{Phys}} \\ \boldsymbol{\sigma}(\mathbf{u}) \cdot \mathbf{n} = \mathbf{t} \quad & \text{on} \quad \Gamma_N \\ \mathbf{u} = \widetilde{\mathbf{u}} \quad & \text{on} \quad \Gamma_D \end{aligned} \tag{13}$$

where $\mathbf{u}$ is the displacement field, $\boldsymbol{\sigma}(\mathbf{u})$ is the Cauchy stress tensor, $\mathbf{b}$ is the body force vector, $\mathbf{n}$ is the outward unit normal vector to Γ_N, $\mathbf{t}$ is the imposed traction on Γ_N and $\widetilde{\mathbf{u}}$ is the imposed displacement on Γ_D.

The weak variational formulation associated to the strong form of the equilibrium equations can be expressed as: find $\mathbf{u} \in [\mathcal{H}^1(\Omega_{\texttt{Phys}})]^3$ such that

$$a(\mathbf{u}, \mathbf{v}) = l(\mathbf{v}) \qquad \forall \mathbf{v} \in \left[\mathcal{H}^1(\Omega_{\texttt{Phys}})\right]^3 \tag{14}$$

where

$$\begin{aligned} a(\mathbf{u}, \mathbf{v}) &= \int_{\Omega_{\texttt{Phys}}} \boldsymbol{\sigma}(\mathbf{u}) : \boldsymbol{\epsilon}(\mathbf{v})\, \mathrm{d}\Omega \\ l(\mathbf{u}) &= \int_{\Omega_{\texttt{Phys}}} \mathbf{b} \cdot \mathbf{v} \mathrm{d}\Omega + \int_{\Gamma_N} \mathbf{t} \cdot \mathbf{v} \mathrm{d}\Gamma \end{aligned} \tag{15}$$

In the above expressions $\boldsymbol{\epsilon}$ is the strain tensor that satisfies

$$\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\epsilon}, \tag{16}$$

where $\mathbf{D}$ is the Hooke's tensor.

4.1. Boundary conditions

One major difficulty associated to the use of immersed boundary methods with Cartesian grids is the fact that, in general, the mesh nodes are not placed on the boundary of the domain, increasing the difficulty to impose Dirichlet boundary conditions in strong form. In this paper, Dirichlet boundary conditions are imposed by means of stabilized Lagrange multipliers. More precisely, the procedure chosen to impose the constraints (i.e., Dirichlet boundary conditions) follows the technique proposed by [9, 20]. This method is suitable for h-refinement based on the use of hierarchical Cartesian grids, where a functional is added to the initial formulation of the problem that has the effect of stabilizing the problem. The stabilization term uses the FE stress field from a previous mesh [9] or a recovered stress field from a previous mesh or the current one [20]. In the second case, an iterative method is defined to solve the problem. In both cases, the definition of the Lagrange multipliers field will allow us to directly condense the degrees of freedom of the Lagrange multipliers at an element level. For the model problem of Equation (13), the weak formulation with Lagrange multipliers reads: find $(\mathbf{u}, \boldsymbol{\lambda}) \in [\mathcal{H}^1(\Omega_{\texttt{Phys}})]^3 \times \left[\mathcal{H}^{-1/2}(\Gamma_D)\right]^3$ such that

$$\begin{aligned} a(\mathbf{u}, \mathbf{v}) + b(\boldsymbol{\lambda}, \mathbf{v}) &= l(\mathbf{v}) \qquad \forall \mathbf{v} \in \left[\mathcal{H}^1(\Omega_{\texttt{Phys}})\right]^3 \\ b(\boldsymbol{\mu}, \mathbf{u}) &= b(\boldsymbol{\mu}, \bar{\mathbf{u}}) \qquad \forall \boldsymbol{\mu} \in \left[\mathcal{H}^{-1/2}(\Gamma_D)\right]^3 \end{aligned} \tag{17}$$

where

$$b(\boldsymbol{\lambda}, \mathbf{v}) = \int_{\Gamma_D} \boldsymbol{\lambda} \cdot \mathbf{v} d\Gamma$$

The stabilized formulation can be derived from a constrained minimization problem solved using the Lagrange multipliers method. Applying the FE discretization and considering the discrete subspaces $\mathscr{U}^h \subset [\mathcal{H}^1(\Omega_{\texttt{Phys}})]^3$ and $\mathscr{L}^h \subset \left[\mathcal{H}^{-1/2}(\Gamma_D)\right]^3$, he problem consist of finding the saddle point of the following functional:

$$\mathscr{L}_s\left(\mathbf{v}^h, \boldsymbol{\mu}^h\right) = \frac{1}{2}a\left(\mathbf{v}^h, \mathbf{v}^h\right) - c\left(\mathbf{v}^h\right) + b\left(\boldsymbol{\mu}^h, \mathbf{v}^h\right) - \frac{1}{2}s\left(\boldsymbol{\mu}^h - \mathbf{T}, \boldsymbol{\mu}^h - \mathbf{T}\right)$$
$$\text{with} \quad s\left(\boldsymbol{\phi}^h, \boldsymbol{\theta}^h\right) = \kappa \sum_e h_e \int_{\Gamma_D^e} \boldsymbol{\phi}^h \cdot \boldsymbol{\theta}^h d\Gamma \tag{18}$$

where h_e is the size of the Dirichlet boundary faces and κ is a positive penalty parameter that is selected to accurately impose the boundary conditions without affecting the convergence rate of the method. The different stabilization methods are obtained by selecting different terms $\mathbf{T}$ in the modified Lagrangian. $\mathbf{T}^*$ can be defined as the traction obtained from the FE stress field $\boldsymbol{\sigma}^*$ of a previous mesh [9], i.e. $\mathbf{T}^* = -\boldsymbol{\sigma}^*\left(\boldsymbol{u}_{i-1}^h\right) \cdot \mathbf{n}$, where $\mathbf{n}$ is the vector normal to the Dirichlet boundary, and $\boldsymbol{u}_{i-1}^h$ are the displacements evaluated in mesh $i-1$. The alternative implemented in this work is to use the recovered solution of the current mesh, and define an iterative process to update the solution [20]. In both cases, the stabilization term prevents excessive oscillations of the Lagrange multipliers solution and κ can be chosen to maintain the optimal convergence rate of the method.

5. Numerical Examples

This section presents a series of examples to demonstrate the applicability and the performance of the proposed methodology for three dimensional problems when the boundary of the domain is described by NURBS or T-spline. The models where previously presented in the section devoted to geometrical, see Figure 1. First, the error associated to the proposed strategy to perform the numerical integration of polynomial functions over NURBS surfaces is studied. Then, the proposed strategy is applied for the numerical solution of linear elastic problems.

5.1. Numerical integration

We first evaluate the accuracy of the proposed approach to perform the integrals of the weak formulation. In fact, only the boundary integrals are of interest because the strategy to perform the integrals on the element interiors use a mapping that is linear in the interior direction and exact integration in this direction is feasible, see Section 3.3.

Let us consider a sphere of unit radius embedded in a coarse mesh with only eight Cartesian elements, as depicted in Figure 1. Let S be the surface integral of a polynomial function f defined as

$$S = \int_{\Gamma} f(x,y,z) \mathrm{d}\Gamma \tag{19}$$

where $\Gamma = \{(x,y,z) \mid x,y,z \geq 0,\ x^2+y^2+z^2=1\}$ represents the surface of the sphere. The numerical result computed with the strategy proposed in this paper, $S_h(f)$, is compared to the analytical result $S_e(f)$. The accuracy is evaluated by defining the relative error in percentage as $100 \times (S_e(f) - S_h(f))/S_e(f)$. To test the performance of the proposed approach we consider constant, linear and quadratic functions. It is worth noting that when a linear approximation of the solution is considered, the elemental stiffness matrix requires the integration of constant functions whereas with quadratic approximations the stiffness matrix requires the integration of constant, linear and quadratic functions. The analytical results are reported here for completeness

$$S_e(f=1) = \frac{\pi}{2}, \qquad S_e(f=x) = S_e(f=y) = S_e(f=z) = \frac{\pi}{4},$$

$$S_e(f=x^2) = S_e(f=y^2) = S_e(f=z^2) = \frac{\pi}{6}, \qquad S_e(f=xy) = S_e(f=xz) = S_e(f=yz) = \frac{1}{3}.$$

Table 2 shows the result of the numerical integration the constant function $f(x,y,z) = 1$ and the linear functions $f(x,y,z) = x$, $f(x,y,z) = y$ and $f(x,y,z) = z$. The percentage error is also reported. These results show how increasing the number of integration points allows us to reduce the error towards machine accuracy. For the constant function $f(x,y,z) = 1$, with 48 integration points in each of the 8 elements used in the analysis (192 integration points in total), the error due to numerical integration is less than 1%. The distribution of integration point is shown in Figure 8a. If we increase the number of integration points to 448 in each element (i.e., 1792 integration points in total) the error due to numerical integration goes down to 9×10^{-10}%. The distribution of integration points in this case is displayed in Figure 8b. It is worth remarking that for the linear functions $f(x,y,z) = x$ and $f(x,y,z) = z$ a comparable accuracy is obtained whereas slightly less accurate results are attained from the linear function $f(x,y,z) = y$.

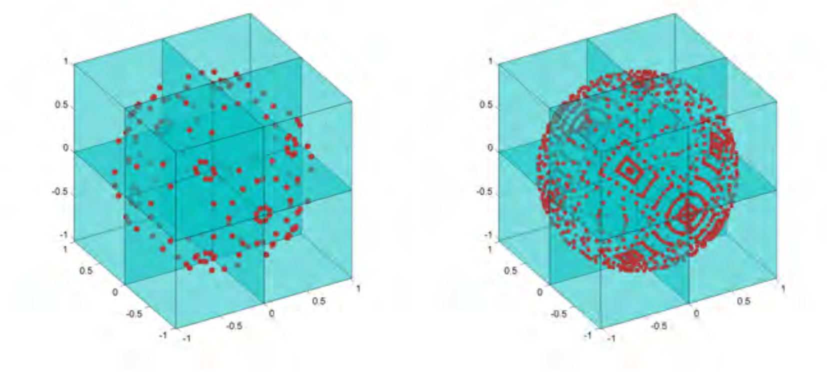
Gauss points	$f = 1$	Error (%)	$f = x$	Error (%)
32	1, 7847357662	13, 6198	0, 8923678831	13, 6198
64	1, 6538967700	5, 2903	0, 8269483850	5, 2903
192	1, 5672909334	0, 2231	0, 7836454667	0, 2231
384	1, 5708636716	0, 0042	0, 7854318358	0, 0042
640	1, 5707956384	$4,38 \cdot 10^{-5}$	0, 7853978192	$4,38 \cdot 10^{-5}$
1344	1, 5707963271	$2,03 \cdot 10^{-8}$	0, 7853981636	$2,03 \cdot 10^{-8}$
1792	1, 5707963268	$9,45 \cdot 10^{-10}$	0, 7853981634	$9,445 \cdot 10^{-10}$

Table 1: Sphere defined by NURBS: Integration error on the surface integral for constant and linear function $f = x$.

Gauss points	$f = y$	Error (%)	$f = z$	Error (%)
32	1, 1085106556	41, 1399	0, 8923678831	13, 6198
64	0, 9164009972	16, 6797	0, 8269483850	5, 2903
192	0, 7754902482	1, 2615	0, 7836454667	0, 2231
384	0, 7855902456	0, 0244	0, 7854318358	0, 0042
640	0, 7853963976	$2,25 \cdot 10^{-4}$	0, 7853978192	$4,38 \cdot 10^{-5}$
1344	0, 7853981668	$4,29 \cdot 10^{-7}$	0, 7853981636	$2,03 \cdot 10^{-8}$
1792	0, 7853981632	$2,04 \cdot 10^{-8}$	0, 7853981634	$9,45 \cdot 10^{-10}$

Table 2: Sphere defined by NURBS: Integration error on the surface integral for linear functions $f = y$ and $f = z$.

Tables 3 and 4 show the result of the numerical integration of quadratic functions and the associated percentage error. Again, it can be observed that with 48 integration points per element (i.e., 192 integration points in total), all the integrals are computed with an error of less than 2% and, in some cases, the error is lower than 1%. Increasing the number of integration points per element the error converges rapidly to machine accuracy. For instance, with 448 in each element (i.e., 1792 integration points in total) the error due to numerical integration is of the order of 2×10^{-6}% or lower.



(a) 8 elements and 192 Gauss points. (b) 8 elements and 1792 Gauss points.

Figure 8: Sphere defined by NURBS: Examples of the mesh used to evaluate the integration error with different number of quadrature points on the surface.

Gauss points	$f = x^2$	Error (%)	$f = y^2$	Error (%)	$f = z^2$	Error (%)
32	$0,5135503309$	$1,9191$	$0,7576351044$	$44,6976$	$0,5135503309$	$1,9191$
64	$0,5137816653$	$1,8749$	$0,6263334393$	$19,6208$	$0,5137816653$	$1,8749$
192	$0,5275430006$	$0,7532$	$0,5122049323$	$2,1760$	$0,5275430006$	$0,7532$
384	$0,5235375836$	$0,0116$	$0,5237885044$	$0,0362$	$0,5235375836$	$0,0116$
640	$0,5235986049$	$3,25 \cdot 10^{-5}$	$0,5235984286$	$6,62 \cdot 10^{-5}$	$0,5235986049$	$3,25 \cdot 10^{-5}$
1344	$0,5235987699$	$1,08 \cdot 10^{-6}$	$0,5235987873$	$2,23 \cdot 10^{-6}$	$0,5235987699$	$1,08 \cdot 10^{-6}$
1792	$0,5235987759$	$5,56 \cdot 10^{-8}$	$0,5235987750$	$1,142 \cdot 10^{-7}$	$0,5235987759$	$5,56 \cdot 10^{-8}$

Table 3: Sphere defined by NURBS: Integration error on the surface integral for quadratic functions.

It is worth emphasizing that the overhead caused by the numerical integration with the exact geometry is restricted to the elements of the Cartesian grid that are cut by the boundary of the embedded geometry. For interior elements the number of integration points is chosen a priori to be the minimum number required to exactly compute the integrals of the weak formulation. For instance, if linear elements are considered, a quadrature with only one integration point guarantees exact integration of the elemental stiffness matrix terms. Analogously, with a quadratic approximation of the solution a tensor product of one-dimensional Gauss quadratures with two points in each

Gauss points	$f = xy$	Error (%)	$f = xz$	Error (%)	$f = yz$	Error (%)
32	$0,5086633254$	$52,5989$	$0,4401458247$	$32,0437$	$0,5086633254$	$52,5989$
64	$0,3946299371$	$18,3889$	$0,3901829374$	$17,0548$	$0,3946299371$	$18,3889$
192	$0,3269680495$	$1,9095$	$0,3293966424$	$1,1810$	$0,3269680495$	$1,9095$
384	$0,3335154896$	$0,0546$	$0,3333287772$	$0,0013$	$0,3335154896$	$0,0546$
640	$0,3333318950$	$4,31 \cdot 10^{-4}$	$0,3333329888$	$1,03 \cdot 10^{-4}$	$0,3333318950$	$4,31 \cdot 10^{-4}$
1344	$0,3333333403$	$2,09 \cdot 10^{-6}$	$0,3333333304$	$8,88 \cdot 10^{-7}$	$0,3333333403$	$2,09 \cdot 10^{-6}$
1792	$0,3333333330$	$1,01 \cdot 10^{-7}$	$0,3333333334$	$2,02 \cdot 10^{-8}$	$0,3333333330$	$1,01 \cdot 10^{-7}$

Table 4: Sphere defined by NURBS: Integration error on the surface integral for quadratic functions.

direction (i.e., eight integration points per hexahedral element) guarantees exact integration of the elemental stiffness matrix terms.

5.2. Discretization error

In this section, a linear elastic analysis is performed on two domains given by a CAD boundary representation with NURBS and T-Splines. The computation is performed with the proposed approach by embedding the CAD geometry in a Cartesian grid and a refinement study is performed in order to evaluate the accuracy of the proposed approach.

In all the examples the Young's modulus and the Poisson ratio are $E = 1000$ and $\nu = 0.3$ respectively. The analytical solution of the problem is the cubic displacement field $\mathbf{u} = (u_x, u_y, u_z)$ with

$$u_x = x+x^2-2xy+x^3-3xy^2+x^2y, \qquad u_y = -y-2xy+y^2-3x^2y+y^3-xy^2, \qquad u_z = 0 \tag{20}$$

Dirichlet boundary conditions, corresponding to the analytical displacement field are considered in the whole boundary.

The exact expression of the stress tensor, obtained by using the constitutive relation is

$$\sigma_x = \frac{E}{1+\nu}\left(2x - 2y + 3x^2 - 3y^2 + 2xy\right), \qquad \sigma_y = \frac{-E}{1+\nu}\left(2x - 2y + 3x^2 - 3y^2 + 2xy\right),$$

$$\sigma_z = \nu\left(\sigma_x + \sigma_y\right), \qquad \tau_{xy} = \frac{E}{1+\nu}(-x-y+\frac{x^2}{2}-\frac{y^2}{2}-6xy), \qquad \tau_{xz} = \tau_{yz} = 0$$

and the volumetric forces required to satisfy the internal equilibrium equation are given by $\mathbf{b} = (b_x, b_y, b_z)$ with

$$b_x = \frac{-E}{1+\nu}(1+y), \qquad b_y = \frac{-E}{1+\nu}(1-x), \qquad b_z = 0 \tag{21}$$

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

$$\eta_e = \left(\frac{\int_\Omega (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}_e) \mathbf{D}^{-1} (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}_e) \, \mathrm{d}\Omega}{\int_\Omega \boldsymbol{\sigma}_e \mathbf{D}^{-1} \boldsymbol{\sigma}_e \mathrm{d}\Omega} \right)^{1/2} \tag{22}$$

where $\boldsymbol{\sigma}_h$ and $\boldsymbol{\sigma}_e$ are the FE (approximated) stress tensor and the analytical stress tensor respectively. In all the numerical examples we select the number of integration points to be enough in order to guarantee that the error due to numerical integration is lower than the spatial discretization error.

5.2.1. Sphere defined by NURBS

The first example considers a sphere of unit radius. A set of four meshes is employed, where uniform refinement is considered. Table 5 summarizes the main properties of the computational meshes used. In particular, this Table shows the number of active elements in each mesh. The number of elements that are interior to the embedded domain and the number of elements intersecting the boundary of the embedded domain is also detailed, together with the number of tetrahedra used to perform the numerical integration. Finally, this table shows the number of degrees of freedom used in the numerical simulation when 8-noded (L8) or 20-noded (Q20) isoparametric hexahedral elements are considered, corresponding to a linear or quadratic approximation of the solution respectively.

Table 6 shows the relative error in the displacement field in energy norm when linear and quadratic elements are used in the four meshes detailed in Table 5. The theoretical optimal convergence rate of the error in energy norm of the FE solution is 1 for the case of linear elements and 2 if quadratic elements are used. The values of the convergence rate of the error in energy norm also displayed in the table show the optimal rate for both linear and quadratic approximations.

The rate of convergence is also displayed, showing the optimal rate for both linear and quadratic approximations.

Mesh	Elements	Internal elems.	Boundary elems.	Tetrahedra	DOFs L8	DOFs Q20
1	8	0(0%)	8(100%)	8	81	243
2	64	8(12.5%)	56(87.5%)	224	375	1275
3	408	136(33.3%)	272(66.7%)	1392	1839	6663
4	2632	1472(55.9%)	1160(44.1%)	6432	10059	37923

Table 5: Sphere defined by NURBS: Information about the calculation meshes.

Mesh	η_e (%) L8	Rate L8	η_e (%) Q20	Rate Q20
1	$52,2879$	–	$9,1053$	–
2	$28,7238$	0.9	$2,9562$	1.6
3	$15,1344$	0.9	$0,8055$	1.9
4	$7,7817$	1.0	$0,2046$	2.0

Table 6: Sphere defined by NURBS: discretization errors and convergence rates using linear (L8) and quadratic (Q20) elements.

The results shown in Table 6 can be seen in Figure 9 as a function of the total number of degrees of freedom. The superiority of quadratic elements is clearly observed, as expected for problems with smooth analytical solution, see for instance [**?**]. In particular, the comparison in Figure 9 shows that the error attained with linear elements in the finest mesh (with 2632 elements) is almost the same as the error attained by using quadratic elements in the coarsest mesh (with only 8 elements).

The displacement field represented over the surface of the sphere is displayed in Figure 10. The result corresponds to a computation using the mesh number 4 with linear elements. The displacement in the z direction is represented to illustrate the error due to the imposition of the Dirichlet boundary condition in weak form by using the technique described in Section 4 because the analytical displacement in this direction is exactly zero, as detailed in 20.

It is worth remarking that the geometry of the sphere has been exactly represented using one quadratic NURBS surface with 27 control points, as represented in Figure 1a. As mentioned earlier in the introduction one of the main advantages of NURBS is the ability to exactly represent conics, which is not possible if a polynomial representation of the geometry is considered.

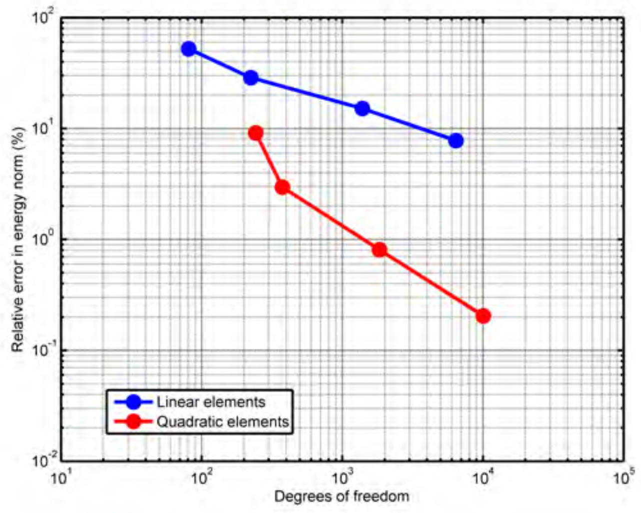
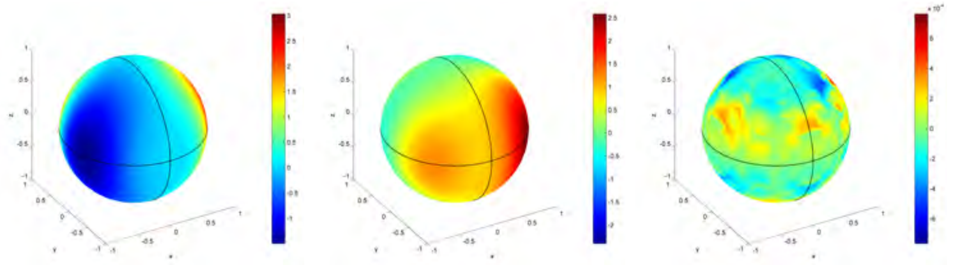


Figure 9: Sphere defined by NURBS: discretization error vs. degrees of freedom for linear and quadratic elements.



(a) (b) (c)

Figure 10: Sphere defined by NURBS: Computed displacement field. (a) x direction, (b) y direction and (c) z direction

5.2.2. Torus defined by NURBS

The second example considers a torus exactly defined by a NURBS surface, see Figure 1b. A set of four meshes is employed, where uniform refinement is considered. Table 7 summarizes the main features of these four computational meshes.

Table 8 shows the relative error in the displacement field in energy norm when linear and quadratic elements are used in the four meshes detailed in

Mesh	Elements	Internal elems.	Boundary elems.	Tetrahedra	DOFs L8	DOFs Q20
1	32	0(0%)	32(100%)	110	225	735
2	216	16(7.4%)	200(92.6%)	992	1101	3891
3	1128	384(34%)	744(66%)	4016	4860	17844
4	7136	3968(55.6%)	3168(44.4%)	18072	26892	101832

Table 7: Torus defined by NURBS: Information about the calculation meshes.

Table 7. The convergence rate of the error in energy norm is also displayed, showing the optimal rate for both linear and quadratic approximations.

Mesh	η_e (%) L8	Rate L8	η_e (%) Q20	Rate Q20
1	$37,3394$	–	$4,1987$	–
2	$21,9811$	0.8	$1,4862$	1.5
3	$11,3050$	1.0	$0,3904$	1.9
4	$5,7553$	1.0	$0,0986$	2.0

Table 8: Torus defined by NURBS: discretization errors and convergence rates using linear (L8) and quadratic (Q20) elements.

Figure 11 represents the relative error in the displacement field in energy norm as a function of the total number of degrees of freedom, both for linear and quadratic elements. The conclusions are similar to the ones obtained in the previous example, showing that the performance of the proposed methodology does not depend on the geometry considered. The superiority of quadratic elements is again clearly observed. both in terms of accuracy and error convergence rate.

The displacement field represented over the surface of the torus is displayed in Figure 12. The result corresponds to a computation using the mesh number 4 with linear elements. Again, the displacement in the z direction is represented to illustrate the error due to the imposition of the Dirichlet boundary condition in weak form by using the technique described in Section 4.

5.2.3. Torus defined by T-spline

The last example considers a torus defined by T-spline, see Figure 1c. The same meshes used in the previous computations are employed, see Table 7. Table 9 shows the relative error in the displacement field in energy norm when linear and quadratic elements are used in the four meshes detailed in Table 7.

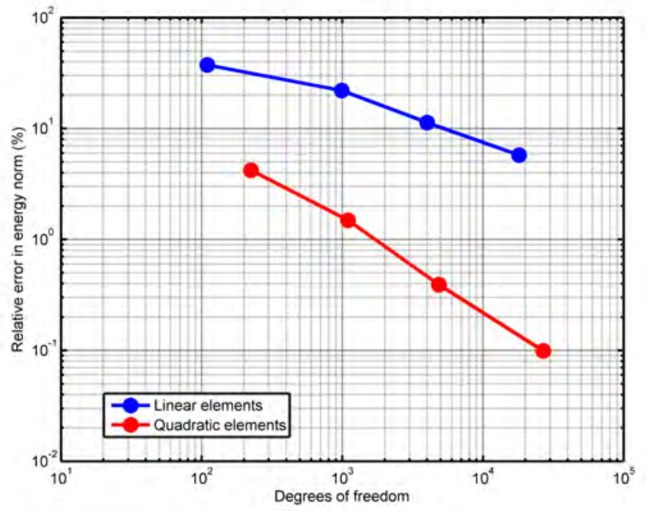


Figure 11: Torus defined by NURBS: discretization error vs degrees of freedom.

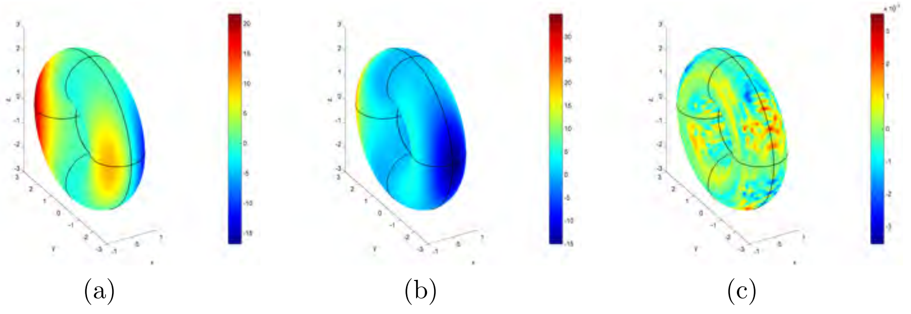


Figure 12: Torus defined by NURBS: computed displacement field. (a) x direction, (b) y direction and (c) z direction

The convergence rate of the error in energy norm is also displayed, showing, once more, the optimal rate for both linear and quadratic approximations.

Figure 13 represents the relative error in the displacement field in energy norm as a function of the total number of degrees of freedom, both for linear and quadratic elements. The conclusions are identical to the ones discussed before, when the torus was represented with NURBS. This illustrates, once more, that the performance of the proposed methodology is independent on

Mesh	η_e (%) L8	Rate L8	η_e (%) Q20	Rate Q20
1	$39,2256$	–	$5,1908$	–
2	$22,7172$	0.8	$1,5881$	1.7
3	$11,7909$	0.9	$0,4172$	1.9
4	$5,9671$	1.0	$0,1056$	2.0

Table 9: Torus defined by T-spline: discretization errors and convergence rate using linear (L8) and quadratic (Q20) elements.

the CAD technology employed to represent the geometry of the embedded domain.

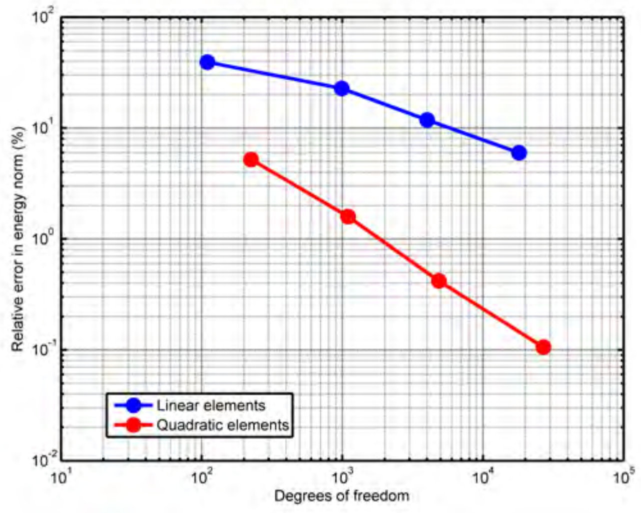


Figure 13: Torus defined by T-spline: discretization error vs degrees of freedom.

6. Conclusions

This papers proposes a novel immersed boundary method where an exact boundary representation of the embedded domain is considered. The method is capable of employing any boundary representation of the embedded domain but the presentation is focused in the most extended CAD technology, namely NURBS, and a novel approach with T-Splines. The proposed technique removes the geometric errors that are associated to standard immersed boundary methods due to the approximation of the embedded geometry by a faceted representation.

The strategy to compute the intersections between the Cartesian grid employed to mesh the embedding domain and the exact geometry of the boundary of the embedded domain is detailed. Then, a novel approach to perform the numerical integration in the region of the cut elements that is internal to the physical domain is developed. The strategy consists on defining a tetrahedral submesh of the region of interest where the tetrahedral subelements have one face or one edge in contact with the boundary of the embedded domain. Then, specifically designed numerical quadratures are defined in the subelements by following the rationale of the NURBS-enhanced finite element method. The performance and accuracy of the proposed technique to compute the integrals appearing in the weak formulation is analyzed using numerical examples.

One crucial aspect in immersed boundary methods is the imposition of essential boundary conditions. As mesh nodes do not lie on the boundary of the physical domain it is not possible to strongly impose such conditions. The technique adopted here consists on using stabilized Lagrange multipliers to impose essential boundary conditions.

Three numerical examples have been considered in order to show the potential and applicability of the proposed methodology. The optimality of the approximation for both linear and quadratic elements has been corroborated. The method shows the same performance on problems where the embedded geometry is represented using NURBS or T-Splines, showing independence on the CAD technology utilized. Finally, all the numerical examples have shown the potential of the proposed approach when quadratic elements are considered.

The present method looks very promising for problems involving large deformations, where traditional mesh-based methods will need mesh adaptation and re-meshing. As future work, the authors consider the implementation of recovery based error estimators, very efficient when Cartesian grids are used. This approach will enable the development of a robust h-refinement strategy and he use improved solutions from finite element analyses.

References

[1] J. M. Escobar, J. M. Cascón, E. Rodríguez, R. Montenegro, A new approach to solid modeling with trivariate T-splines based on mesh op-

timization, Computer Methods in Applied Mechanics and Engineering 200 (45-46) (2011) 3210–3222.

[2] J. M. Melenk, I. Babuska, The Partition of Unity Finite Element Method : Basic Theory and Applications The Partition of Unity Finite Element Method :, Tech. Rep. 96 (1996).

[3] J. Liu, S. Zhu, B. He, Finite Element Modeling of Human Head from Medical Images last ~~~~~~~~~~~~~ mit mid l alongest andrepatSteedge (2007) 133–135.

[4] F. Cirak, M. Ortiz, P. Schröder, Subdivision surfaces: a new paradigm for thin-shell finite-element analysis, Internat. J. Numer. Methods Engng. 47 (12) (2000) 2039–2072.

[5] K. Inoue, Y. Kikuchi, T. Masuyama, A NURBS finite element method for product shape design, J. Engrg. Design 16 (2) (2005) 157–174.

[6] 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.

[7] H. J. Kim, Y. D. Seo, S. K. Youn, Isogeometric analysis with trimming technique for problems of arbitrary complex topology, Computer Methods in Applied Mechanics and Engineering 199 (4548) (2010) 2796 – 2812. doi:http://dx.doi.org/10.1016/j.cma.2010.04.015.

[8] 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.

[9] M. Tur, J. Albelda, E. Nadal, Imposing Dirichlet boundary conditions in hierarquical cartesian meshes by means of stabilized Lagrange multipliers, International Journal for Numerical Methods in Engineering (2013) 1–18doi:10.1002/nme.

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

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

[12] M. J. Borden, M. A. Scott, J. A. Evans, T. J. Hughes, Isogeometric finite element data structures based on Bézier extraction of NURBS, International Journal for Numerical Methods in Engineering 87 (2011) 15–47.

[13] R. Goldman, T. Lyche, Knot Insertion and Deletion Algorithms for B-spline Curves and Surfaces, Society for Industrial and Applied Mathematics, Philadelphia, 1993.

[14] W. Martin, E. Cohen, R. Fish, P. Shirley, Practical ray tracing of trimmed nurbs surfaces, Journal of Graphics Tools 5 (2000) 27–52.

[15] W. E. Lorensen, H. E. Cline, Marching cubes: A high resolution 3d surface construction algorithm, ACM SIGGRAPH Computer Graphics 21 (4) (1987) 163–169.

[16] 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.

[17] W. J. Gordon, C. A. Hall, Transfinite element methods: Blending-function interpolation over arbitrary curved element domains, Numer. Math. 21 (2) (1973) 109–129.

[18] S. Wandzura, H. Xiao, Symmetric quadrature rules on a triangle, Comput. Math. Appl. 45 (12) (2003) 1829–1840.

[19] R. Sevilla, S. Fernández-Méndez, Numerical integration over 2D NURBS shaped domains with applications to NURBS-enhanced FEM, Finite Elements in Analysis and Design 47 (10) (2011) 1209–1220.

[20] 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, Submitted.