UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Department of Mechanical and Materials Engineering

Ph.D. Thesis

Structural shape optimization based on the use of Cartesian grids

---

Presented by: Onofre Marco Alacid M.Sc.

Supervised by: Dr. Juan José Ródenas García

Dr. Manuel Tur Valiente

Valencia, July, 2017

Ph.D. Thesis

Structural shape optimization based on the use of Cartesian grids

for the degree of

Doctor in Industrial Engineering and Production

presented by

Onofre Marco Alacid M.Sc.

at the

Department of Mechanical and Materials Engineering

of Universitat Politècnica de València

Supervised by

Dr. Juan José Ródenas García
Dr. Manuel Tur Valiente

Valencia, July, 2017

Ph.D. Thesis

Structural shape optimization based on the use of Cartesian grids

Presented by: Onofre Marco Alacid M.Sc.

Supervised by: Dr. Juan José Ródenas García

Dr. Manuel Tur Valiente

QUALIFYING TRIBUNAL

President: Dr. _____

Vocal: Dr. _____

Secretary: Dr. _____

Valencia, July, 2017

Abstract

As ever more challenging designs are required in present-day industries, the traditional trial-and-error procedure frequently used for designing mechanical parts slows down the design process and yields suboptimal designs, so that new approaches are needed to obtain a competitive advantage. With the ascent of the Finite Element Method (FEM) in the engineering community in the 1970s, structural shape optimization arose as a promising area of application.

However, due to the iterative nature of shape optimization processes, the handling of large quantities of numerical models along with the approximated character of numerical methods may even dissuade the use of these techniques (or fail to exploit their full potential) because the development time of new products is becoming ever shorter.

This Thesis is concerned with the formulation of a 3D methodology based on the Cartesian-grid Finite Element Method (cgFEM) as a tool for efficient and robust numerical analysis. This methodology belongs to the category of embedded (or fictitious) domain discretization techniques in which the key concept is to extend the structural analysis problem to an easy-to-mesh approximation domain that encloses the physical domain boundary.

The use of Cartesian grids provides a natural platform for structural shape optimization because the numerical domain is separated from a physical model, which can easily be changed during the optimization procedure without altering the background discretization. Another advantage is the fact that mesh generation becomes a trivial task since the discretization of the numerical domain and its manipulation, in combination with an efficient hierarchical data structure, can be exploited to save computational effort.

However, these advantages are challenged by several numerical issues. Basically, the computational effort has moved from the use of expensive meshing algorithms towards the use of, for example, elaborate numerical integration schemes designed to

capture the mismatch between the geometrical domain boundary and the embedding finite element mesh. To do this we used a stabilized formulation to impose boundary conditions and developed novel techniques to be able to capture the exact boundary representation of the models.

To complete the implementation of a structural shape optimization method an adjunct formulation is used for the differentiation of the design sensitivities required for gradient-based algorithms. The derivatives are not only the variables required for the process, but also compose a powerful tool for projecting information between different designs, or even projecting the information to create h-adapted meshes without going through a full h-adaptive refinement process.

The proposed improvements are reflected in the numerical examples included in this Thesis. These analyses clearly show the improved behavior of the cgFEM technology as regards numerical accuracy and computational efficiency, and consequently the suitability of the cgFEM approach for shape optimization or contact problems.

Resumen

La competitividad en la industria actual impone la necesidad de generar nuevos y mejores diseños. El tradicional procedimiento de prueba y error, usado a menudo para el diseño de componentes mecánicos, ralentiza el proceso de diseño y produce diseños subóptimos, por lo que se necesitan nuevos enfoques para obtener una ventaja competitiva. Con el desarrollo del Método de los Elementos Finitos (MEF) en el campo de la ingeniería en la década de 1970, la optimización de forma estructural surgió como un área de aplicación prometedora.

El entorno industrial cada vez más exigente implica ciclos cada vez más cortos de desarrollo de nuevos productos. Por tanto, la naturaleza iterativa de los procesos de optimización de forma, que supone el análisis de gran cantidad de geometrías (para las se han de usar modelos numéricos de gran tamaño a fin de limitar el efecto de los errores intrínsecamente asociados a las técnicas numéricas), puede incluso disuadir del uso de estas técnicas.

Esta Tesis se centra en la formulación de una metodología 3D basada en el *Cartesian-grid Finite Element Method* (cgFEM) como herramienta para un análisis numérico eficiente y robusto. Esta metodología pertenece a la categoría de técnicas de discretización *Immersed Boundary* donde el concepto clave es extender el problema de análisis estructural a un dominio de aproximación, que contiene la frontera del dominio físico, cuya discretización (mallado) resulte sencilla.

El uso de mallados cartesianos proporciona una plataforma natural para la optimización de forma estructural porque el dominio numérico está separado del modelo físico, que podrá cambiar libremente durante el procedimiento de optimización sin alterar la discretización subyacente. Otro argumento positivo reside en el hecho de que la generación de malla se convierte en una tarea trivial. La discretización del dominio numérico y su manipulación, en coalición con la eficiencia de una estructura jerárquica de datos, pueden ser explotados para ahorrar coste computacional.

Sin embargo, estas ventajas pueden ser cuestionadas por varios problemas numéricos. Básicamente, el esfuerzo computacional se ha desplazado. Del uso de costosos algoritmos de mallado nos movemos hacia el uso de, por ejemplo, esquemas de integración numérica elaborados para poder capturar la discrepancia entre la frontera del dominio geométrico y la malla de elementos finitos que lo embebe. Para ello, utilizamos, por un lado, una formulación de estabilización para imponer condiciones de contorno y, por otro lado, hemos desarrollado nuevas técnicas para poder captar la representación exacta de los modelos geométricos.

Para completar la implementación de un método de optimización de forma estructural se usa una formulación adjunta para derivar las sensibilidades de diseño requeridas por los algoritmos basados en gradiente. Las derivadas no son sólo variables requeridas para el proceso, sino una poderosa herramienta para poder proyectar información entre diferentes diseños o, incluso, proyectar la información para crear mallas h-adaptadas sin pasar por un proceso completo de refinamiento h-adaptativo.

Las mejoras propuestas se reflejan en los ejemplos numéricos presentados en esta Tesis. Estos análisis muestran claramente el comportamiento superior de la tecnología cgFEM en cuanto a precisión numérica y eficiencia computacional. En consecuencia, el enfoque cgFEM se postula como una herramienta adecuada para la optimización de forma.

Resum

Actualment, amb la competència existent en la industria, s'imposa la necessitat de generar nous i millors dissenys . El tradicional procediment de prova i error, que amb freqüència es fa servir pel disseny de components mecànics, endarrereix el procés de disseny i produeix dissenys subòptims, pel que es necessiten nous enfocaments per obtindre avantatge competitiu. Amb el desenvolupament del Mètode dels Elements Finits (MEF) en el camp de l'enginyeria en la dècada de 1970, l'optimització de forma estructural va sorgir com un àrea d'aplicació prometedora.

No obstant això, a causa de la natura iterativa dels processos d'optimització de forma, la manipulació dels models numèrics en grans quantitats, junt amb l'error de discretització dels mètodes numèrics, pot fins i tot dissuadir de l'ús d'aquestes tècniques (o d'explotar tot el seu potencial), perquè al mateix temps els cicles de desenvolupament de nous productes s'estan acurtant.

Esta Tesi se centra en la formulació d'una metodologia 3D basada en el *Cartesian-grid Finite Element Method* (cgFEM) com a ferramenta per una anàlisi numèrica eficient i sòlida. Esta metodologia pertany a la categoria de tècniques de discretització *Immersed Boundary* on el concepte clau és expandir el problema d'anàlisi estructural a un domini d'aproximació fàcil de mallar que conté la frontera del domini físic.

L'utilització de mallats cartesians proporciona una plataforma natural per l'optimització de forma estructural perquè el domini numèric està separat del model físic, que podria canviar lliurement durant el procediment d'optimització sense alterar la discretització subjacent. A més, un altre argument positiu el trobem en què la generació de malla es converteix en una tasca trivial, ja que la discretització del domini numèric i la seua manipulació, en coalició amb l'eficiència d'una estructura jeràrquica de dades, poden ser explotats per estalviar cost computacional.

Tot i això, estos avantatges poden ser qüestionats per diversos problemes numèrics. Bàsicament, l'esforç computacional s'ha desplaçat. De l'ús de costosos algoritmes de mallat ens movem cap a l'ús de, per exemple, esquemes d'integració numèrica

elaborats per poder capturar la discrepància entre la frontera del domini geomètric i la malla d'elements finits que ho embeu. Per això, fem ús, d'una banda, d'una formulació d'estabilització per imposar condicions de contorn i, d'un altra, desevolupem noves tècniques per poder captar la representació exacta dels models geomètrics

Per completar la implementació d'un mètode d'optimització de forma estructural es fa ús d'una formulació adjunta per derivar les sensibilitats de disseny requerides pels algoritmes basats en gradient. Les derivades no són únicament variables requerides pel procés, sinó una poderosa ferramenta per poder projectar informació entre diferents dissenys o, fins i tot, projectar la informació per crear malles h-adaptades sense passar per un procés complet de refinament h-adaptatiu.

Les millores proposades s'evidencien en els exemples numèrics presentats en esta Tesi. Estes anàlisis mostren clarament el comportament superior de la tecnologia cgFEM en tant a precisió numèrica i eficiència computacional. Així, l'enfocament cgFEM es postula com una ferramenta adient per l'optimització de forma.

Acknowledgements

First of all, I would like to thank my supervisors Professor Juan José Ródenas and Professor Manuel Tur for their advice and support during these years. They not only provided the technical guidance necessary to pull this Thesis off but a work environment based on patience, trust and comprehension.

I would like to extend my gratitude to all my colleagues and staff of the Department of Mechanical and Materials Engineering. It was my pleasure to share this stage of my life with them.

I have to thank very particularly Professor Yongjie Zhang for letting me be part of his research group in the Computational Biomodeling Lab at Carnegie Mellon University. In addition, I will never forget the contributions made by Professor Rubén Sevilla. Definitely his help has made my work more powerful.

I would also like to express my gratitude to the Spanish society for funding the public education system which provided me the opportunity of growing as a scientist in the framework of the FPI scholarship program.

Finally, I would like to express my very special thanks to my parents, family and friends, for bringing me to this world, providing the best life I could ever imagined and supporting me unconditionally always.

Thank you very much.

Contents

Abstract vii

Resumen ix

Resum xi

Acknowledgements xiii

I Thesis overview 1

1 Introduction 3
- 1.1 Motivation . . . 3
- 1.2 State of the art . . . 6
  - 1.2.1 Structural optimization . . . 6
  - 1.2.2 Immersed Boundary Method numerical integration . . . 7
  - 1.2.3 Boundary conditions in the Immersed Boundary Method . . . 8
  - 1.2.4 Shape sensitivity analysis . . . 9
- 1.3 Objective . . . 9
- 1.4 Thesis layout . . . 11

2 Problem description 13
- 2.1 Governing equations . . . 13
- 2.2 Finite Element approximation using Cartesian grids . . . 15
- 2.3 Structural shape optimization . . . 18

3 Contributions **19**
3.1 Exact representation of an immersed boundary 19
3.1.1 Mesh-geometry intersection 22
3.1.2 Integration over subdomains 26
3.2 Essential boundary conditions 33
3.3 h-adapted meshing 41
3.3.1 Geometrical refinement 42
3.3.2 Error-based refinement 45
3.4 Shape sensitivities and optimization 49
3.4.1 Design velocity fields 51
3.4.1.1 Generation of boundary velocity fields 52
3.4.1.2 Generation of domain velocity fields 56
3.4.2 Calculating shape sensitivities with FEAVox 60
3.4.2.1 Evaluation of derivatives 61
3.4.3 Optimization using Cartesian grids 62
3.4.3.1 Data sharing 62
3.4.3.2 Nested domain reordering 65
3.4.3.3 Automatic h-adaptive mesh projection 67

4 Closure **73**
4.1 Summary 73
4.2 Open research lines 74

Bibliography **77**

II Articles **89**

Paper A: Exact 3D boundary representation in finite element analysis based on Cartesian grids independent of the geometry **91**

Paper B: Stabilized method of imposing Dirichlet boundary conditions using a recovered stress field **131**

Paper C: Robust h-adaptive meshing strategy considering exact arbitrary CAD geometries in a Cartesian grid framework **173**

Paper D: An extension of shape sensitivity analysis to an Immersed Boundary Method based on Cartesian grids **221**

Paper E: Structural shape optimization using Cartesian grids and automatic h-adaptive mesh projection **265**

Part I

Thesis overview

Chapter 1

Introduction

1.1. Motivation

The Finite Element Method (FEM) is a numerical technique for finding approximate solutions to Boundary Value Problems (BVPs) described by Partial Differential Equations (PDEs). It originated around the middle of the last century from the need to solve complex elasticity and structural analysis problems in civil and aeronautical engineering. But it was in the 1960s and 1970s when the method gained a rigorous mathematical basis and was adopted by a wide variety of physical and engineering disciplines. Nowadays, the *momentum* of the method is still growing and we can find FEM applied to chemistry, biology, plasma dynamics or weather prediction, to name just a few examples.

FEM has become predominant over other methods of analysis and simulation for Computer Aided Engineering (CAE) for the design of structural components. Traditionally, when a component is being designed for a structure or mechanism, the geometry is first defined by a Computer Aided Design (CAD) system and then analyzed by CAE software to predict its behavior under certain load situations. After simulation with the CAE software, should the component not behave as expected, the user has to modify its geometry, mostly by a manual or poorly automatized process. This trial-and-error process has been the most widely used in the design of structural components over the last 50 years. As the final result strongly depends on the designer's experience, optimal designs is by no means guaranteed. Another serious drawback is that it requires many hours of analysis to obtain the final optimized ge-

ometry since the user has to keep on checking the results and modifying the geometry until the final or, at least an acceptable, geometry is obtained. This process can be very slow and so is little suited to highly competitive industrial environments, such as the automotive and aerospace industries, where reducing design cycle time is crucial.

Today's high-tech industries need more efficient design processes able to provide, not only suitable, but the optimal solutions within a reasonable time, with human intervention restricted to the initial steps of the optimization and unnecessary during the iterative process. In order to achieve this objective, we need to couple the optimization process with the component design, which means parameterizing the geometry of the component and then running an optimization algorithm over the parameters. Theoretically, this will provide the combination of parameters with the optimal geometry for the application under a set of prescribed constraints.

However, the evaluation of design parameters during an optimization analysis is not necessarily a trivial step. Although, some phenomena are defined by simple easily managed functions, most engineering problems are usually defined by PDEs that can only be solved using numerical tools such as FEM. In this area, however, there are several practical problems that are not already solved. For instance, a large number of computationally expensive analyses are required to obtain a component's optimal geometry, making the process prohibitive for many practical applications. Additionally, a robust and efficient meshing method for very complex geometries is required for the traditional FEM and still has not been fully achieved in commercial codes. As a result, FE analysts need to check each mesh before running the analysis. The aim of this work is thus to advance in three main areas: (i) the robustness of FE models obtained from CAD geometries, (ii) the accuracy of the FE analyses and (iii) the efficiency of the overall optimization process.

This Thesis proposes FEAVox, a 3D implementation of the methodology based on the Cartesian-grid Finite Element Method (cgFEM) [1], which has the appropriate combination of techniques to achieve our objective. The most important feature of cgFEM is that it does away with the geometry-conforming mesh of traditional FEM, as the mesh used to solve the FE problem is independent of the component's geometry. The cgFEM framework has two domains, the problem domain $\Omega_{\texttt{Phys}}$ and the meshing domain Ω, which is a parallelepiped, trivial to mesh, embedding $\Omega_{\texttt{Phys}}$.

This framework thus makes the tedious meshing process of the traditional FEM unnecessary, but obviously, does not come free of charge. In fact, we slide the computational cost from standard mesh generation to the treatment of the elements intersected by the boundary. Basically the task will consist of finding the part of the cut elements that belong to the problem domain $\Omega_{\texttt{Phys}}$. In this regard, we propose robust strategies to find the intersections between the Cartesian grid and the model and techniques that generate the integration subdomains necessary to properly capture the domain in the intersected region.

Once the robustness issue has been addressed, the next concern is accuracy, which is the main objection to Immersed Boundary Methods (IBM). The argument is ba-

sically that: with an immersed boundary we do not necessarily have nodes lying on the boundary, thus (i) the discretization does not properly capture the geometry, and (ii) the imposition of boundary conditions needs special treatment. These well-known drawbacks were in fact the inspiration for the contributions of this work. As will be explained in detail later, we propose novel techniques to allow our numerical models to exactly capture the geometry of the components to be analyzed, together with a formulation to impose essential boundary conditions.

Last but not least, efficiency is the keystone of the cgFEM methodology. In this Thesis, we extend the ideas for shape optimization proposed in [1] to 3D, adding new tools to improve the overall performance of the optimization analysis. We will rely on the Cartesian mesh and on its specifically implemented hierarchical data structure to reduce the size of the calculations. For instance, the integration of the stiffness matrices for structural analysis can be easily performed taking into account that the internal Cartesian elements can be identical in shape but only different in size. In this scenario, we can evaluate a large proportion of the domain using only one element and a scaling factor.

Also, when using gradient-based optimization algorithms, information on the derivatives of objective functions and constraints is needed. In this regard, we propose a shape sensitivity analysis formulation adapted to the immersed boundary environment. In addition, the combination of shape sensitivity analysis and the same meshing domain for all the geometries to be analyzed during an optimization process would provide the following advantages:

- the possibility of transfering calculations (element stiffness matrices, discretization errors, etc.) performed on previously analyzed geometries to other geometries, considerably reducing the computational burden throughout the analysis;
- reduce the resolution time of the system of equations using a Cartesian-based domain decomposition technique;
- the ability to project information found during the optimization process between evolving geometries, which will significantly diminish the computational cost associated with the generation of h-refined meshes.

1.2. State of the art

1.2.1. Structural optimization

The optimization of structural components through the use of the FE analysis is an important and active research field of significant interest in engineering. There are two common approaches to representing the domain which we seek to optimize:

i. A density function: this approach, common in topology optimization [2, 3], is very general and computationally convenient, but the boundary representation is not sharp and thus typically fine grids and low order approximation spaces are employed.

ii. An implicit or explicit representation of the boundary: given the boundary representation we need to generate a discretization of the domain when it is updated.

We will only consider the second case in this Thesis. There are different approaches and many codes to solve the optimization problem. However, some problems in this context, identified a long time ago [4, 5], still remain unsolved, like the incorporation of robust parametrization techniques for the definition of each design and the unwanted variations in the structural response due to mesh-dependency effects. Also, in many optimization processes based on the use of finite element analyses, there is no control of the accuracy of the numerical solution. As a result, when the process comes to an end there is no guarantee that the final outcome will be practicable; a more accurate analysis would reveal that the final design is unfeasible if it contravenes one or more of the constraints imposed. The control of the error associated with FE computation and its influence on the solution of the optimization problem was analyzed in [6].

Two main concepts have been developed to bypass these drawbacks. On the one hand, the design update procedure can be assigned to a geometry model [7, 8]. This avoids manipulating the nodal coordinates within the finite element discretization, thus removing impracticable patterns that cannot be defined by any combination of the design variables.

On the other hand, the finite element model could be updated through the optimization procedure to improve the accuracy of the numerical simulation results or to enhance the element quality [9, 10, 11]. This may for instance take the form of geometric mesh smoothing operations or more elaborate mesh updating procedures, such as the technique developed in [12] and can also include adaptive mesh refinement based on error estimation in the energy norm [13], goal oriented adaptivity[14] or error controlled adaptive mesh refinement [6]. However, when it comes to complicated geometries or to large shape changes, these strategies may still necessitate computationally expensive re-meshing algorithms.

One way of removing these *frictions* between geometrical and numerical models, would be to integrate CAD representations with FEM. A recent trend in this regard has been the Isogeometric Analysis (IGA) [15, 16]. The idea is to improve the geometrical accuracy of the models by integrating CAD representations with the FEM solvers. However, in its finite element form, generating an analysis-suitable solid discretization is an open topic [17, 18, 19]. Some works on IGA shape optimization can be found in [20, 21, 22, 23, 24].

1.2.2. Immersed Boundary Method numerical integration

From this perspective, the so-called Immersed Boundary (IB) discretization techniques seem the most appropriate choice for structural shape optimization. For these methods, the main idea is to extend the structural analysis problem to an easy-to-mesh approximation domain that encloses the physical domain boundary. Then, it suffices to generate a discretization based on the approximation domain subdivision, rather than a geometry-conforming finite element mesh. Moreover, when the structural component is allowed to evolve, the physical points move through the fixed discretization created from the embedding domain where there will be no mesh distortion. There are plenty of alternatives within the scope of IB. Among many other names used to describe these FE techniques, where the mesh does not match the domain's geometry, we have the Immersed Boundary Method (IBM) [25] and the Immersed Finite Element Method (IFEM) [26]. These methods have been applied to a wide range of problems including, of course, shape optimization [27, 28, 29, 30, 31].

Nevertheless, the attractive advantages of IBM come together with numerical challenges. Basically, the computational effort has moved from the use of expensive meshing algorithms towards the use of, for example, elaborate numerical integration schemes able to capture the mismatch between the geometrical domain boundary and the embedding finite element mesh. All the intersected elements have to be integrated properly in order to account for the volume fractions interior to the physical domain. Studies on the domain integration for these methods can be found in the literature.

The first idea, is to homogenize the material properties within intersected elements based on the actual volume fraction of these elements covered by the domain. This approach is straightforward and could be computationally efficient, but provides low accuracy for the structural analysis [32, 33].

A more selective approach is employed in the Finite Cell framework [34, 35, 36] in which a number of integration points are provided for all intersecting elements employing hierarchical *octree* data structures [37, 38, 39] for their distribution. Only the integration points interior to the physical domain are considered in the respective

integral contribution, although, regardless of the number of integration points, the exact representation of the geometry is not possible.

Considering an approximation to the exact boundary involves geometry-modeling errors. If the convergence rate of these errors is lower than the convergence rate of the FE discretization error, then the optimal convergence rate of the FE solution will be lost as it will be dominated by the convergence rate of the geometry modeling errors. Very recently, several methodologies have emerged to perform high-order integration in embedded methods, such as the so-called *smart octrees* tailored for Finite Cell approaches [40] or those in which the geometry is defined implicitly by level sets [41]. Even in the former, where the isoparamentric reoriented elements only provide an approximate FE description of the boundary, the exact geometry is not taken into account in the integration stage.

Considering the exact geometry when integrating would improve accuracy and retain the optimal convergence rate of the FE solution. In the embedded domain framework this can be done by using a separate element-wise tetrahedralization for integration purposes only [42, 43].

1.2.3. Boundary conditions in the Immersed Boundary Method

Boundary conditions have to be imposed to complete an FE formulation. Since no degrees of freedom are assigned directly to the boundary, the procedures used in the standard FEM cannot be used to apply the boundary conditions. The case of the Neumann boundary conditions can be easily tackled by simply taking into account that the integration surface can cut the element and does not necessarily have to coincide with the element faces. However, the case of the Dirichlet boundary conditions is much more complex. To solve the problem, a common alternative is to use the Lagrange multipliers technique. It can be difficult to find compatible discretizations of displacements and multipliers that satisfy the *InfSup* condition [44] but 2D [45] and 3D [46] methods of doing this can be found in the bibliography. The naive choice for the multiplier interpolation based on the element edge intersection does not satisfy the *InfSup* condition. The main problem appears because the number of Lagrange multipliers is too high, which can cause undesired oscillations in the Lagrange multipliers field. Thus, one of the alternatives is to stabilize the solution [47, 48].

One of the most popular ways of stabilizing the solution is Nitsche's method, which can be derived from the Barbosa-Hughes stabilization [47], see for example [49, 50, 51] for early implementations of the method in immersed boundaries or [46, 52] for a comparison with other methods. The stabilizing term in Nitsche's method has an algorithmic constant to be defined that affects its stability. As has been pointed

out[46, 53, 54], this constant depends on how the boundary intersects the underlying mesh and can become unbounded for certain configurations. At the same time very large values of the penalty constant result in an ill-conditioned system [55]. This issue has been approached for interface and X-FEM problems by choosing appropriate values for each intersected element [56, 57, 58, 59]. In these works the stabilizing term is the finite element stress field, as in the Barbosa-Hughes stabilization, but computed in a different way. In [56] the stress is computed in the largest element partition. In the weighted Nitsche's method [57, 59] the stress fields of both partitions are weighted from the partition sizes.

1.2.4. Shape sensitivity analysis

In optimal structural design, sensitivity analysis calculates the derivatives of the structural response (displacements, stresses, etc.) with respect to design variables. The initial developments in this field focused on size variables of size, such as thickness or cross-sectional areas of structural components. However, in many structural problems it was necessary to consider shape as a design variable. Currently, we can distinguish four different approaches: global finite differences, continuum or discrete derivatives and computational differentiation. Overall finite differences [60, 61, 62, 63] consist of the use of two FE analyses to obtain the derivatives with respect to a design parameter. Continuum derivatives [64, 65, 66, 67, 68, 69, 70] are obtained differentiating the governing elasticity equations. This process leads to a set of continuum sensitivity equations that are then discretized and solved. For discrete derivatives [71, 72, 73, 74] the derivation-to-discretization procedure is reversed and the components of the discretized system of equations are differentiated with respect to the design variables. Computational differentiation [75, 76] is related to the automatic differentiation of the routines within the computational code, for which several tools were developed [77, 78, 79].

The different approaches can be evaluated from the point of view of accuracy, relation to discretization, computational cost and implementation effort [80, 81, 82, 83, 84].

1.3. Objective

The present Thesis aims to contribute to the research field of Immersed Boundary Methods applied to shape optimization problems by proposing solutions to the traditional drawbacks associated with this type of approach, namely: a) the approximation

of the boundary on cut elements and b) Dirichlet boundary conditions imposition. Moreover, new tools have been developed to take advantage of the strong hierarchical structure inherent in the cgFEM methodology, applied to shape optimization problems.

In line with these ideas, the partial objectives are as follow:

1. **Generation of robust *h*-adapted FE 3D models.**

   It is necessary to define a robust procedure to evaluate the relative position of the elements with respect to the physical boundary, so specific strategies are required to find the intersection of the elements with the boundary and to perform the numerical integration only over the region of the intersected elements lying inside the computation domain. The intersection algorithms have to be able to take into consideration not only approximations of the geometry (i.e. linear tessellations) but parametric surfaces such as NURBS or T-spline. In addition, the generation of the integration submesh within intersected elements has to be both efficient and general enough to ensure the flawless discretization of the models. The numerical model generation has to take into account the need to allow h-adaptation of the meshes to improve the accuracy of the analyses. The strategies followed to generate robust models will be described in Section 3.1 and 3.3, and discussed in detail in Paper A and Paper C.

2. **Enhancing the quality of the domain integration allowing the exact consideration of the geometry.**

   One of the most challenging aspects of IBM is to consider the exact boundary of the domain in the evaluation of volume integrals. Achieving greater accuracy and ensuring the optimal convergence rate of the FE solutions involves performing the numerical integration over the true computational domain instead of simplifying the embedded geometry. As explained in Section 1.2, State of the art, this topic has attracted the attention of the fictitious-domain community, who are trying to improve the accuracy along the boundary. Our solution for the integration problem is highlighted in Section 3.1, and an in-depth explanation is given in Paper A.

3. **Improving the imposition of Dirichlet boundary conditions.**

   As indicated in the previous section, imposing Dirichlet boundary conditions on immersed boundary meshes is no trivial task. Although there have been many contributions in this area in recent years, the quest for a formulation that solves the ill-conditioning of the matrices and gives stable results for any arbitrary cutting configuration is still going on. Section 3.2 will be devoted to explaining a stabilized formulation that tries to improve the behavior of these methods. A detailed description of the method and its features is given in Paper B.

4. **Developing a shape sensitivity strategy adapted to immersed boundary environments.**

   Gradient-based optimization processes of mechanical components require information on the derivatives of the magnitudes of interest. Writing this Thesis provided the opportunity to implement an adjoint formulation that meets the requirements of the Cartesian grid setting, regardless of the geometry. This formulation is described in Section 3.4 and in detail in Paper D.

5. **Merging all the features of our Cartesian-grid methodology to solve shape optimization problems.**

   Due to the nature of shape optimization processes there is always room for new advances, either in the optimization algorithms themselves or in the numerical solver used to evaluate the different geometries. Regarding the latter, embedded methods provide a set of features that engage smoothly with the requirements of optimization procedures. With the goal of saving computational cost always in mind, several tools based on the Cartesian hierarchical structure are proposed in Section 3.4 and Paper E.

1.4. Thesis layout

The remainder of this document is divided into two parts: the first is an overview of the Thesis and includes Chapter 2, with the formal definition of the equations of structural analysis, including the formulation for non-conforming meshes, and also states the shape optimization problem. Chapter 3 is an overview of the main contributions of the Thesis and Chapter 4 is devoted to the conclusions and further research.

In Chapter 3 the overview of the contributions does not strictly follow the order of the publication or submission of the papers. Even though the chronological and conceptual order of the papers is correct, in a consistent and non-repetitive overview similar ideas regarding the same concept must be explained together. In order to clarify this situation, the author will indicate the origin of the contributions in terms of publications related to the present work. For instance, the mesh-geometry intersection is given in Papers A and C, so both papers will be referred to.

The second part of the Thesis consists of five papers in which the scientific contributions of this work are discussed in detail. All the papers will be presented without journal editing, and the cover of each one will include the citation of the corresponding journal or information on the submission.

Paper A describes a NURBS-enhanced integration technique for an immersed boundary environment. Paper B discusses a stabilized method of imposing Dirichlet boundary conditions in non-conforming meshes. Paper C improves the procedures for creating robust models using h-adapted Cartesian meshes. Paper D introduces a sensitivity analysis formulation adapted to immersed environments. Finally, Paper E applies the 'natural' advantages of Cartesian grids and the previous contributions to shape optimization problems.

Chapter 2

Problem description

2.1. Governing equations

Let us consider the linear elastic problem. Let $\Omega_{\texttt{Phys}} \in \mathbb{R}^d$, with $d = 2$ or $d = 3$ be a bounded domain with a sufficiently smooth boundary Γ. The contour can be divided into two non-overlapping parts, Γ_{D} and Γ_{N}, where Dirichlet and Neumann conditions are respectively imposed. The aim is to find the displacement field $\mathbf{u} \in \mathscr{U}$ that fulfills the internal equilibrium equation in the domain and the Dirichlet and Neumann boundary conditions on the boundary, which can be written as follows:

$$\begin{aligned} \nabla \boldsymbol{\sigma}(\mathbf{u}) + \mathbf{t}_v &= 0 && \text{in} \quad \Omega \\ \boldsymbol{\sigma}(\mathbf{u})\,\mathbf{n} &= \mathbf{t}_s && \text{on} \quad \Gamma_{\mathrm{N}} \\ \mathbf{u} &= \mathbf{g} && \text{on} \quad \Gamma_{\mathrm{D}} \\ \boldsymbol{\epsilon}(\mathbf{u}) &= \mathbf{D}\,\boldsymbol{\sigma}(\mathbf{u}) \end{aligned} \tag{2.1}$$

In the above expression $\mathbf{t}_v \in [L^2(\Omega)]^d$ are the volume forces, $\mathbf{t}_s \in [L^2(\Omega)]^d$ are the tractions imposed on the Neumann boundary, $\mathbf{g}$ are the displacements imposed on the Dirichlet boundary and $\mathbf{n}$ is the unit vector normal to the surface. $\mathscr{U} \equiv [H^1(\Omega)]^d$ is the Hilbert space of functions whose integral of the first derivative over the domain is bounded. In linear elasticity, the strain tensor is defined from displacement field by

$$\boldsymbol{\epsilon}(\mathbf{u}) = \big(\nabla \mathbf{u} + \nabla^{\mathrm{T}} \mathbf{u}\big)/2 \tag{2.2}$$

The constitutive equation, which relates the strains with the stresses by means of the tensor $\mathbf{D}$, can be expressed in the case of isotropic behavior using two material-dependent constants, the Young's modulus E and the Poisson's ratio ν. This relationship can be written as

$$\boldsymbol{\epsilon} = \left(\boldsymbol{\sigma} - \nu \left(\operatorname{tr}\left(\boldsymbol{\sigma}\right)\mathbf{I} - \boldsymbol{\sigma}\right)\right)/E \tag{2.3}$$

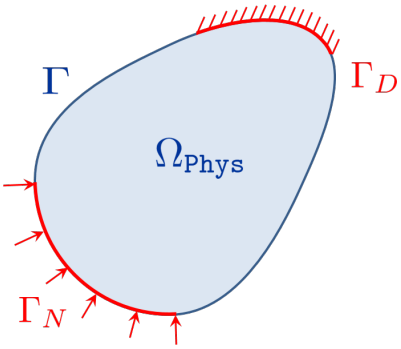


Figure 2.1: Elastic body and boundary conditions.

It is straightforward to show the following property concerning the constitutive equation, which will be used below.

Property 1. *The scalar product of the tractions can be bounded by the energy per unit volume with a constant C_E, which depends on the material properties, as*

$$\|\boldsymbol{\sigma}(\mathbf{u})\|^2 \leq C_E \left(\boldsymbol{\sigma}(\mathbf{u}) : \boldsymbol{\epsilon}(\mathbf{u})\right) \quad \text{with} \quad C_E = \frac{E}{1-2\nu} \tag{2.4}$$

The weak variational formulation of the elastic problem allows different approaches to imposing the Dirichlet boundary conditions. The most common procedure is to impose a constraint in the space of virtual displacement $\mathscr{V}$, i.e. the virtual displacement is zero on the Dirichlet boundary. The virtual work of the elastic forces is in equilibrium with the virtual work of the external forces applied, as follows:

$$a\left(\mathbf{u}, \mathbf{v}\right) = c\left(\mathbf{v}\right) \quad \forall \mathbf{v} \in \mathscr{V} \tag{2.5}$$

where

$$\begin{aligned} a\left(\mathbf{u}, \mathbf{v}\right) &= \int_{\Omega} \boldsymbol{\sigma}\left(\mathbf{u}\right) : \boldsymbol{\epsilon}\left(\mathbf{v}\right) \, \mathrm{d}\Omega \\ c\left(\mathbf{v}\right) &= \int_{\Omega} \mathbf{v} \cdot \mathbf{t}_v \, \mathrm{d}\Omega + \int_{\Gamma_{\mathrm{N}}} \mathbf{v} \cdot \mathbf{t}_s \, \mathrm{d}\Gamma \end{aligned} \tag{2.6}$$

This method is simple to implement and effective in the context of the standard finite element method, in which the boundary of the geometry is properly represented by the mesh. However, this method is not valid for Cartesian meshes, since it is very difficult to get a null field on the Dirichlet boundary if the contour of the geometry does not match with the edges of the elements.

For this reason it seems more appropriate to seek another formulation, which involves raising the elastic problem to a minimization problem with constraints. This means finding the displacement field $\mathbf{u}$ that minimizes the total potential energy, subject to the constraints imposed by Dirichlet boundary conditions. The problem can be expressed as

$$\min \left(\frac{1}{2} a\left(\mathbf{v}, \mathbf{v}\right) - c\left(\mathbf{v}\right) \right) \quad \text{with} \quad \mathbf{v} = \mathbf{g} \text{ in } \Gamma_{\mathrm{D}} \tag{2.7}$$

One approach to solving this minimization problem is to use the Lagrange multipliers method. In addition to the displacement field, a new field of Lagrange multipliers $\boldsymbol{\lambda}$ associated with the reaction forces is added. Formally, the problem is to find the saddle point $[\mathbf{u}, \boldsymbol{\lambda}] \in \mathscr{U} \times \mathscr{M}$ of the following Lagrangian

$$\mathscr{L}\left(\mathbf{v}, \boldsymbol{\mu}\right) = \frac{1}{2} a\left(\mathbf{v}, \mathbf{v}\right) + b\left(\boldsymbol{\mu}, \mathbf{v} - \mathbf{g}\right) - c\left(\mathbf{v}\right) \tag{2.8}$$

The Lagrange multipliers belong to the Hilbert space $\mathscr{M} = [H^{-\frac{1}{2}}(\Gamma_{\mathrm{D}})]^d$ and then the following functional is defined

$$\begin{aligned} b\left(\cdot, \cdot\right) &: \mathscr{M} \times \mathscr{U} \to \mathbb{R} \\ b\left(\boldsymbol{\mu}, \mathbf{u}\right) &= \int_{\Gamma_{\mathrm{D}}} \boldsymbol{\mu} \cdot \mathbf{u} \, d\Gamma \end{aligned} \tag{2.9}$$

2.2. Finite Element approximation using Cartesian grids

The domain is subdivided into finite elements by a Cartesian mesh in which the boundary of the domain does not necessarily coincide with the edges of the elements, but can pass through them. For the approximate solution of the virtual work statement in Equation (2.8), we consider a collection of n_e finite elements Ω^e (paral-

lelepipeds, cubes in most cases) to establish a discrete domain representation $\Omega_{\texttt{Approx}}$ of $\Omega_{\texttt{Phys}}$

$$\Omega_{\texttt{Phys}} \subseteq \Omega_{\texttt{Approx}} = \bigcup_{e=1}^{n_e} \Omega^e \tag{2.10}$$

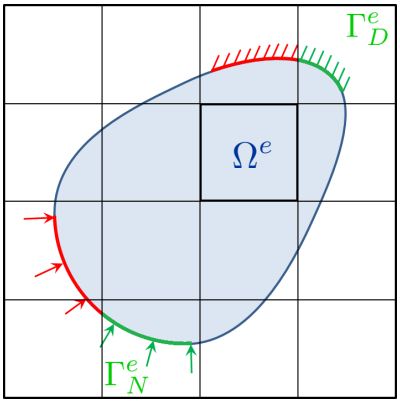


Figure 2.2: Immersed Boundary Finite Element discretization of the elastic body.

The spaces of the finite element solution are denoted as $\mathscr{U}^h \subset \mathscr{U}$ for displacements and $\mathscr{M}^h \subset \mathscr{M}$ for multipliers. Substituting the finite element fields in Equation (2.8) and optimizing the Lagrangian we obtain the following system of equations:

$$\begin{aligned} a\left(\mathbf{u}^h, \mathbf{v}^h\right) + b\left(\boldsymbol{\lambda}^h, \mathbf{v}^h\right) &= c\left(\mathbf{v}^h\right) \qquad && \forall \mathbf{v}^h \in \mathscr{U}^h \\ b\left(\boldsymbol{\mu}^h, \mathbf{u}^h\right) &= b\left(\boldsymbol{\mu}^h, \mathbf{g}\right) && \forall \boldsymbol{\mu}^h \in \mathscr{M}^h \end{aligned} \tag{2.11}$$

where $\mathbf{v}^h$ and $\boldsymbol{\mu}^h$ are the variations of the displacement and multiplier fields and $[\mathbf{u}^h, \boldsymbol{\lambda}^h]$ is the solution. Standard shape functions[85] are used to interpolate the displacement as $\mathbf{u}^h = \mathbf{N}\mathbf{u}^e$, where $\mathbf{u}^e$ is the vector of nodal displacements. The discrete system (2.11) yields a linear system of equations that will give as a result the displacements at nodes and also the discrete Lagrange multipliers field:

$$\left(\bigcup_{e=1}^{n_e} \begin{bmatrix} \mathbf{k}^e & \mathbf{B}^{eT} \\ \mathbf{B}^e & \mathbf{0} \end{bmatrix}\right) \begin{Bmatrix} \mathbf{u}^e \\ \lambda^e \end{Bmatrix} = \bigcup_{e=1}^{n_e} \begin{Bmatrix} \mathbf{f}^e \\ \mathbf{g}^e \end{Bmatrix} \tag{2.12}$$

The stiffness matrix $\mathbf{k}^e$ and force vector $\mathbf{f}^e$ in (2.12) are built as in the standard FE formulation.

The stiffness matrix of each element is computed by

$$\mathbf{k}^e = \int_{\Omega^e} \mathbf{B}^T \mathbf{D} \mathbf{B} |\mathbf{J}| \mathrm{d}\Omega \tag{2.13}$$

where

Ω^e is the domain in local element coordinates,

$\mathbf{B}$ is the nodal strains-displacements matrix,

$\mathbf{D}$ is the stiffness matrix that relates stresses with strains. We consider linear elasticity where, under isotropic behavior, this matrix depends only on E, the Young's modulus, and ν, the Poisson's ratio of the material,

$|\mathbf{J}|$ is the determinant of the matrix $\mathbf{J}$, which represents the Jacobian matrix of transformation of the global coordinates (x, y, z) to the local element coordinates (ξ, η, τ).

The vector $\mathbf{f}_q$ is the standard FE vector due to point forces, volumetric forces, forces distributed over the Neumann surface of the element, evaluated assembling the contribution $\mathbf{f}_q^e$ of every element e on the domain:

$$\mathbf{f}_q^e = \int_{\Gamma_N^e} \mathbf{N}^T \mathbf{t} |\mathbf{J}| \mathrm{d}\Gamma + \int_{\Omega^e} \mathbf{N}^t \mathbf{b} |\mathbf{J}| \mathrm{d}\Omega + \mathbf{p} \tag{2.14}$$

where vectors $\mathbf{t}$, $\mathbf{b}$ and $\mathbf{p}$ correspond to the surface, body and point loads, respectively.

Remark 1. *The choice of the interpolation of the Lagrange multipliers determines the optimal convergence of the formulation and allows* $\mathbf{B}$ *and* $\mathbf{g}$ *to be evaluated. In particular, in order to achieve this convergence the Ladyzhenskaya-Babuška-Breezi (LBB or InfSup) condition [44] must be fulfilled. Another alternative is the use of stabilization methods, as in this Thesis.*

2.3. Structural shape optimization

The structural shape optimization problem can be tackled as the minimization of a real function f, which depends on some variables and is subjected to several constraints. The generic form of such problem is:

$$\begin{array}{cll} \text{minimize} & f(\mathbf{a}) & \\ \text{where} & \mathbf{a} = \{a_i\} & i = 1, \ldots, n \\ \text{verifying} & \mathbf{g}(\mathbf{a}) \leq 0 & j = 1, \ldots, m \\ \text{and} & \mathbf{h}(\mathbf{a}) = 0 & k = 1, \ldots, l \end{array} \tag{2.15}$$

f being the objective function, a_i are the design variables, g_j are inequality constraints and the values h_k define equality constraints. The vector $\mathbf{a}$ defines a specific structural shape and the task consists of finding the $\mathbf{a}$ values which define the optimum design.

There are plenty large number of optimization algorithms to solve the problem stated in Equation 2.15. Gradient-based algorithms are based on the next iterative expression:

$$\mathbf{a}^q = \mathbf{a}^{q-1} + \alpha \mathbf{S}(\mathbf{a})^q \tag{2.16}$$

where q is the iteration number, $\mathbf{S}(\mathbf{a})^q$ is the search direction vector and α is a parameter related to the step size.

The physical interpretation of the last term is that $\mathbf{S}(\mathbf{a})^q$ is a direction vector defining the optimal combination of the simultaneous variations of the design variables $\mathbf{a}$, and α represents the magnitude of the movement along the vector $\mathbf{S}(\mathbf{a})^q$. Together, the terms $\alpha \mathbf{S}(\mathbf{a})^q$ represent the value of the perturbation $\delta \mathbf{a}$ to be applied to the design in the present iteration. The difference between the different gradient-based optimization algorithms lies in the way of choosing the search direction $\mathbf{S}(\mathbf{a})^q$, and how to determine the value of α.

In order to solve the optimization problem, most of the algorithms require the computation of the objective function, the constraints and their derivatives (sensitivities) with respect to the design variables for each geometry considered during the process. The gradient is a vector whose components are the derivatives of $f(\mathbf{a})$ with respect to the design variables with respect to every one of the design variables:

$$\nabla f(\mathbf{a}) = \left\{ \frac{\delta f}{\delta a_1}, \frac{\delta f}{\delta a_2}, \cdots, \frac{\delta f}{\delta a_n} \right\}^T \tag{2.17}$$

Geometrically, the gradient vector of a function defines the direction in which the function grows most rapidly.

Chapter 3

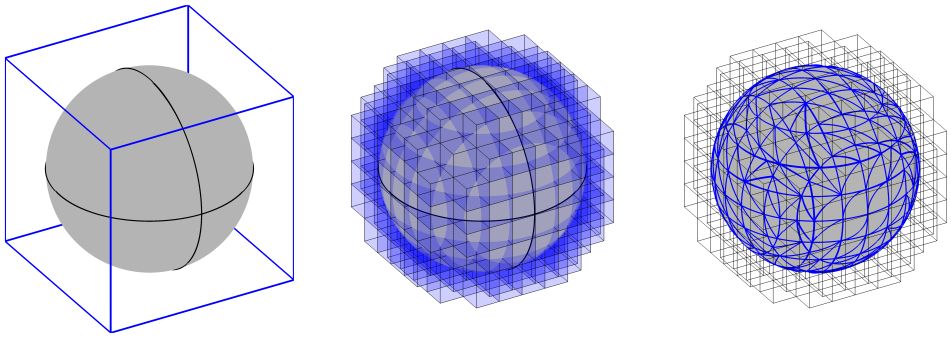
Contributions

This section is an overview of the major Thesis contributions, indicating their novel aspects and key ideas.

3.1. Exact representation of an immersed boundary

As has been previously mentioned, the most widely used approach in classical FEM involves 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 re-meshing are costly and require a substantial amount of human hours [86, 87]. And a certain amount of experience is required to refine the mesh so as 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 $\Omega_{\texttt{Phys}} \subset \mathbb{R}^3$ (see Figure 3.1a) with boundary $\Gamma_{\texttt{IB}} = \partial\Omega_{\texttt{Phys}}$, the key principle of FEAVox, or any other IBM, consists of defining an embedding domain Ω such that $\Omega_{\texttt{Phys}} \subset \Omega$ (with a much simpler geometry than $\Omega_{\texttt{Phys}}$), being Ω extremely easy to mesh compared to the domain of interest $\Omega_{\texttt{Phys}}$. In FEAVox, we consider Ω to be a cuboid, and use a Cartesian grid to mesh it, as shown in Figure 3.1b. 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 the



(a) Geometrical model $\Omega_{\texttt{Phys}}$ within embedding domain, Ω.

(b) Discretization $\Omega_{\texttt{Approx}}$ of the embedding domain.

(c) Integration domain, $\Omega^h_{\texttt{Phys}}$.

Figure 3.1: Typical Immersed Boundary Method environment.

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 $\Omega_{\texttt{Phys}}$, named $\Omega^h_{\texttt{Phys}}$ (Figure 3.1c). The effect of this approximation can be considerable if high-order elements are used, or if the accurate representation of the geometry plays a key role, as in shape optimization or contact problems. This method is often used to obtain good results from an engineering point of view, although the solution of the mesh refinement processes will not converge to the solution of the problem defined in $\Omega_{\texttt{Phys}}$, but instead to the solution of the problem defined in $\Omega^h_{\texttt{Phys}}$ if the approximation $\Omega^h_{\texttt{Phys}}$ is maintained throughout the mesh refinement. Even in the case in which $\Omega^h_{\texttt{Phys}}$ is refined together with the mesh, the optimal convergence rate of the error of the FE solution can be compromised if the convergence rate of the geometry-modeling errors is lower than the FE discretization error. To overcome this problem the CAD description of the boundary of the physical domain $\Gamma_{\texttt{IB}}$ is here considered during the integration process using numerical integration techniques that account for the exact geometry defined by NURBS or T-spline. The result of this is that the volume integrals are as exact as the accuracy of the numerical quadrature used in their evaluation.

FEAVox is based on the use of a sequence of uniformly refined Cartesian meshes. The different levels of the Cartesian meshes are connected by predefined hierarchical relationships. The present implementation is an extrapolation of the 2D data structure based on the element subdivision shown in [88]. The data structure considers the hierarchical relationship between the elements of different refinement levels, obtained during the element subdivision process, to accelerate FE computations. In FEAVox, the element used on the coarsest level of the Cartesian grid pile is called

the *reference element*. The data structure has been modified to the particular case in which all the elements are geometrically identical to the reference element. One of the main benefits of this data structure 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 evaluation of the elemental matrices. For instance, the analysis proposed in [89] shows that the number of operations required to compute the elemental matrices can be reduced by a factor of 10 when a mapping with constant Jacobian is considered with low-order hexahedral elements. The hierarchical relationships considered in the data structure simplify the mesh refinement and the precomputation of most of the data used during the analyses and remarkably improves the efficiency of the FE implementation. In addition, the hierarchical structure allows the algorithmic evaluation of nodal coordinates, mesh topology, hierarchical relationships, and neighborship patterns, among other geometrical information. Therefore, there is no need to store this information in the memory, which makes the proposed algorithm more efficient, not only in terms of computational expense but also in terms of memory requirements.

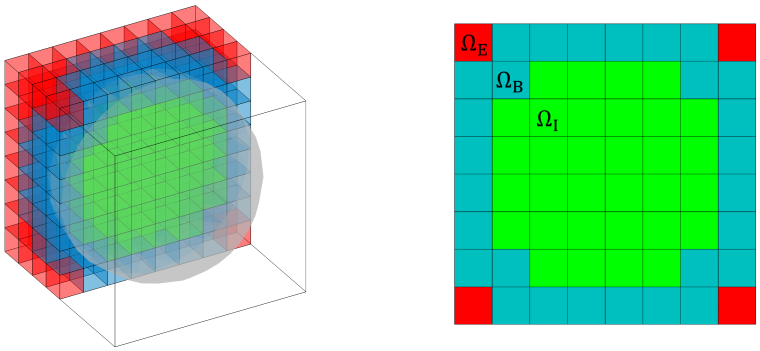
The first step in the proposed strategy is the creation of the FE analysis mesh used to solve the boundary value problem, which can be obtained by classifying the elements of the Cartesian grid as follows:

- Boundary elements: elements cut by the boundary of the physical domain, i.e., the 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: outside the physical domain, elements $\Omega_{\mathtt{E}}$ such that $\Omega_{\mathtt{E}} \subset \Omega \setminus \Omega_{\mathtt{Phys}}$,

Figure 3.2 shows these three types of elements for the sphere embedded in a Cartesian grid represented in Figure 3.1b.

The analysis mesh is formed by the internal and 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 reduce the computational cost of the element matrices.

Since we are working with meshes completely independent of the embedded geometry, it is necessary to determine the relative position of the elements cut by the boundary 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 execute these two operations are proposed in the remainder of this section.



(a) Perspective view. (b) 2D section.

Figure 3.2: Section of a three-dimensional Cartesian grid showing the three different types of elements: (1) In red, external elements, Ω_E, not considered in the analysis, (2) in blue, boundary elements, Ω_B, intersected by the embedded domain and (3) in green, interior elements, Ω_I.

3.1.1. Mesh-geometry intersection

As a logical consequence of the independence of the analysis mesh with the geometrical model, the problem of discriminating which part of mesh is inside or outside the target model arises. 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 with a tessellated boundary, but since we here use the exact parametric representation of the boundary the solution required needs to be more sophisticated.

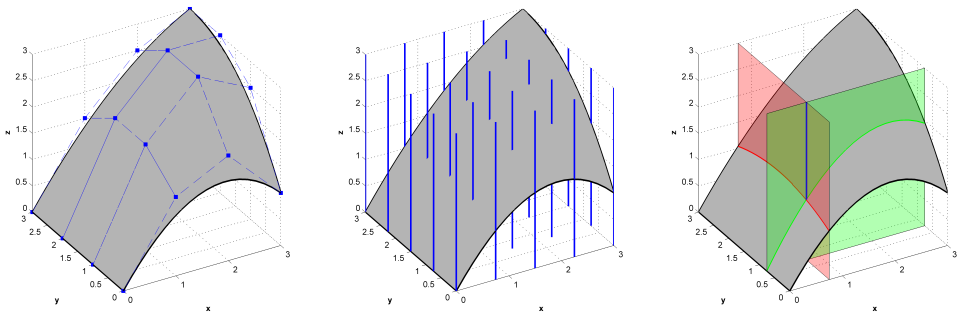
There are several methods available in the literature to evaluate the intersection between parametric surfaces and arbitrary rays. These are known as ray-tracing strategies and are widely used by the computer graphics community and the animation and videogames industries.

If the surface is defined by a tessellation, ray-tracing is performed on the resulting set of triangular surfaces, which is algebraically trivial. When using parametric surfaces, the curve-surface intersection is usually solved by a numerical method. Many algorithms for ray-tracing parametric surfaces are available in the literature[90, 91, 92, 93, 94].

In this Thesis we propose a robust algorithm for finding the intersections of a Cartesian grid and parametric surfaces. The algorithm includes the multivariate Newton's 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 only give details of how to find an accurate initial guess for the intersections.

The process will be illustrated by an example. Figure 3.3a shows 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. Figure 3.3b displays a set of axes defined along Z that intersect the parametric surface.



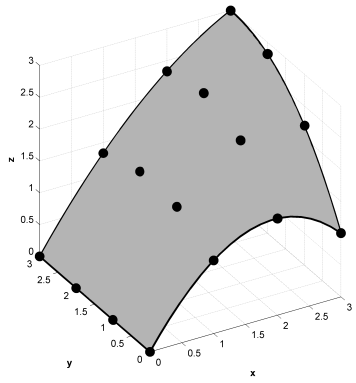
(a) Parametric surface. (b) Axes along Z direction. (c) Cartesian planes of an axis.

Figure 3.3: Example of a surface and Cartesian axes.

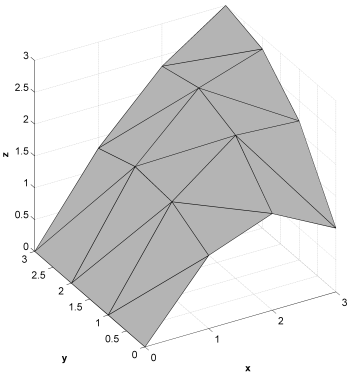
In addition, we know that each of the Cartesian axes will be defined by two Cartesian planes, as shown in Figure 3.3c, where we can see that the intersection between a YZ-plane, defined by the coordinate x, and a XZ-plane, defined only by y, yield a Z-axis.

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 from a set of points, as shown in Figures 3.4b and 3.4a, respectively. It is worth noting that 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.

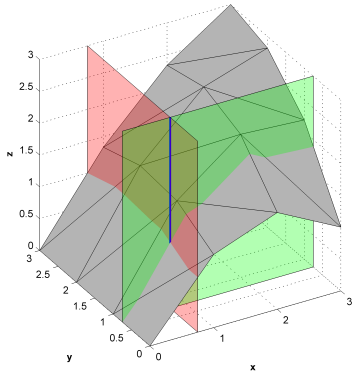
Once we have the auxiliary triangulation we will evaluate level sets of the intersection planes (Figure 3.4c) with respect to the points on the parametric surface. In this way we will identify the position of the points with respect to the two planes by evaluating the sign of the distances in the physical 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. A view of the level sets calculated can be seen in Figure 3.4d. The signs of the distances to the planes XZ and YZ are in red and green, respectively. Our strategy consists of find-



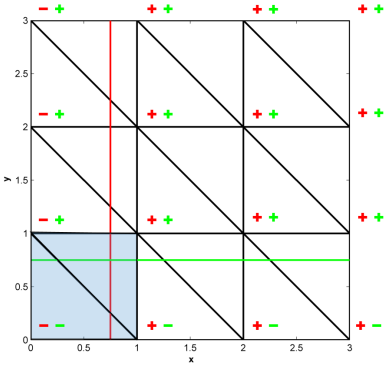
(a) Arbitrary discretization of the surface.



(b) Triangulation generated.



(c) Triangulation and intersection planes.



(d) 2D view of the level sets.

Figure 3.4: Procedure to find the Newton-Raphson initial guess.

ing 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 3.5a), 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 3.5b). Otherwise, the triangle is not intersected when the signs of all vertices are the same

(Figure 3.5c). As we 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. With these criteria we can easily identify the area where we have to choose the initial guess for the axis for the case in Figure 3.4 (see Figure 3.4d).

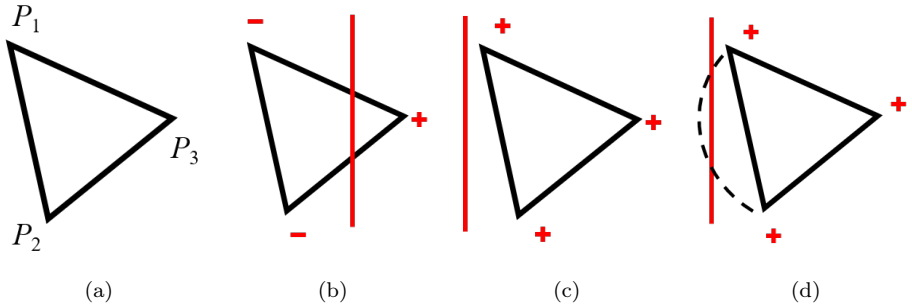


Figure 3.5: Intersecting the Cartesian planes and the triangulation. (a) Element of the triangulation. (b) Cutting plane. (c) Not-cutting plane. (d) Case of intersection not detected.

Although this strategy is very simple to implement we identified three safety procedures that are necessary to ensure its robustness:

Triangulation size. If the triangulation is too coarse the criteria described above are not strict enough and will be easily met. This could yield, for instance, a situation in which a Cartesian axis intersects the same triangle several times, thus the same initial guess would be considered for the computation of two different roots and would prevent the convergence of the Newton-Raphson algorithm in some cases (see Paper A - Section 3.2).

Approximate triangulation. When using 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 3.5d, where the signs 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. This criterion will flag the ambiguous triangles where there could be intersections. To eliminate this ambiguity we subdivide the triangle and recalculate the criteria. This subdivision is recursively applied until the ambiguity is eliminated (see Paper C - Section 3).

Coplanar surfaces. It is important to identify the surfaces coplanar to the Cartesian axes to avoid 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, it is easy to detect any axis contained in it (see Paper C - Section 3).

After following the previous guidelines, we have now 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 for the intersection. After obtaining these initial points we will compute the intersections using the Newton-Raphson method, as previously indicated. 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. Elements are automatically classified as internal, boundary or external by counting the number of internal and external nodes in each element.

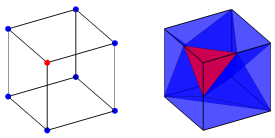
3.1.2. Integration over subdomains

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

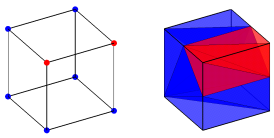
In IBM internal elements are treated as standard finite elements in which the integration is performed using a tensor product of one-dimensional Gauss quadratures with the desired number of points in each direction. However, the contribution of the boundary elements $\Omega_{\mathtt{B}}$ requires special 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 approach proposed in Paper A was inspired by the Marching Cubes (MC) algorithm [95], which uses a set of templates for the intersection between the surfaces and edges of cubes. The MC algorithm is widely used in computational graphics to represent approximations of surfaces, as it is very efficient in 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

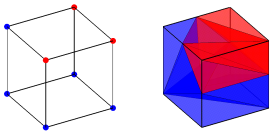
tetrahedralization of each one of these patterns is generated and stored. To facilitate the implementation, and without loss of generality, we assume that the Cartesian elements are intersected at most once by the boundary of the physical domain. From this premise, we need only seven out of fourteen templates of the original MC algorithm (1, 2, 5, 8, 9, 11 and 14, see [95]). 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 the geometry. The seven patterns considered are depicted in Figure 3.6. In the figures we can see the nodal topologies and the set of tetrahedra used for each pattern. The colors identify internal and external subdomains (or different materials in the case of multi-material problems).



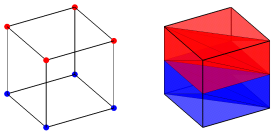
(a) Configuration 1.



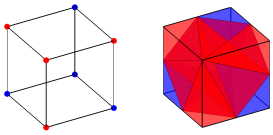
(b) Configuration 2.



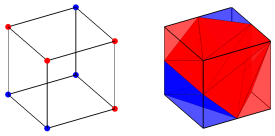
(c) Configuration 3.



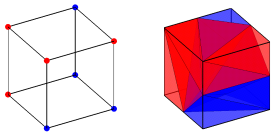
(d) Configuration 4.



(e) Configuration 5.



(f) Configuration 6.

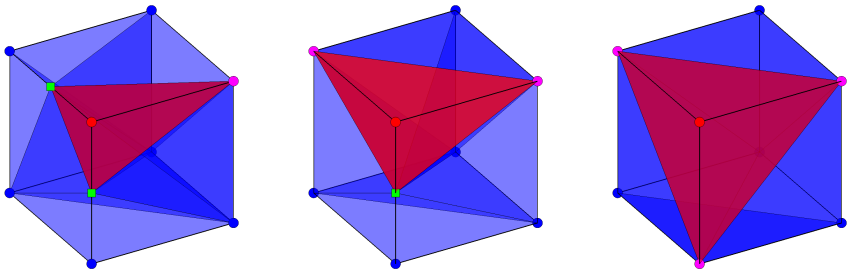


(g) Configuration 7.

Figure 3.6: Intersection patterns inspired by the MC algorithm. Nodal topology (left) and tetrahedralization (right).

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 cases are very likely to appear 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 3.6a, we can quickly see that we have as many options as intersections we can move to the nodes, as shown in Figure 3.7, the boundary nodes being the magenta dots.



(a) One boundary node. (b) Two boundary nodes. (c) Three boundary nodes.

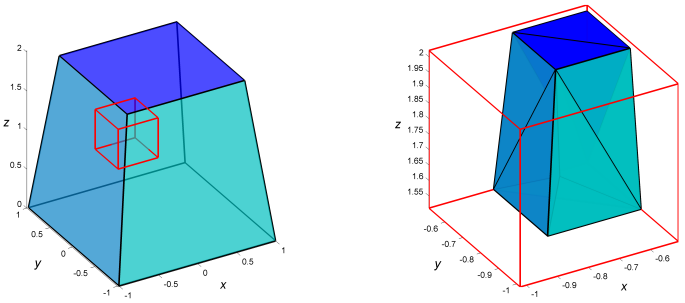
Figure 3.7: Configuration 1. Different options when considering surface intersections on boundary nodes.

In Paper C - Section 4 we propose a strategy to handle all these new possibilities using exactly the same number of stored patterns, i.e. the 7 original patterns. This is possible because of the relationship between boundary nodes and intersections, and the latter with the integration patterns.

The basic patterns presented are valid when the elements are intersected by only one surface, but in most problems there will be elements intersected by several surfaces at the same time. Sharp features are often found inside an element generated by the interfaces of connecting surfaces (see Figure 3.8a). The proposed method evaluates these elements individually and generates specific sets of tetrahedra, using for instance a Delaunay procedure, as in Figure 3.8b.

Following [96], the 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 these subdivisions are only applied to design a numerical quadrature for integration. Two examples are given to illustrate the proposed strategy. The first example considers a tetrahedral element Ω_T with two faces on different surfaces (face $\{P_1 - P_2 - P_4\}$ on Γ_A and face $\{P_2 - P_3 - P_4\}$ on Γ_B), see Figure 3.9a. 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 shown in Figure 3.9b. In this case, the subdomain is split into three tetrahedrons using the geometric center P_F of the only face not lying on a boundary (face $\{P_1 - P_2 - P_4\}$). New subdomains are then defined as a linear convex combination

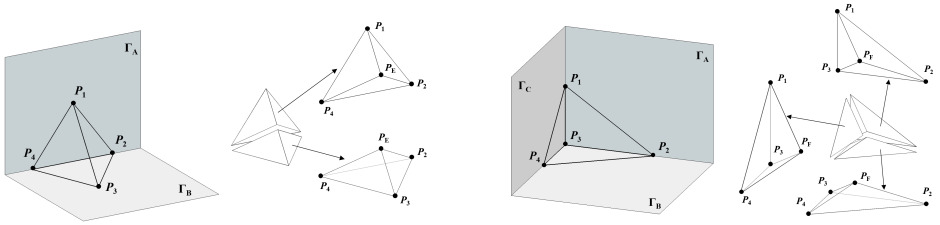


(a) Element intersected by several patches of a trapezoidal prism.

(b) Detail of the resulting tetrahedralization.

Figure 3.8: Exception to the intersection patterns.

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



(a) Two faces on different parametric boundaries.

(b) Three faces on different parametric boundaries.

Figure 3.9: Subdivision of tetrahedrons.

The evaluation of these elements has a higher computational cost than elements with standard patterns, however, the ratio of elements with configurations not represented by the standard patterns to the number of elements in the mesh is in general very low.

The numerical integration over the region $\Omega_B^{\texttt{Phys}}$ is then accomplished by integrating over each subdomain of the tetrahedralization by the strategy proposed in the NEFEM [97]. This methodology was designed to incorporate the exact boundary of the computational domain into body-fitted FE simulations. The advantages of this technique with respect to the classical FEM have been demonstrated for a variety of problems (see [98]).

A tetrahedral subdomain T_e^F with a face on the physical boundary is parametrized using the mapping

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

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

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

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 important property of the NEFEM mappings is the ability to decouple the directions of the surface definition, Λ_e and $[\xi_1,\xi_2]$ in mappings $\boldsymbol{\Psi}$ and $\boldsymbol{\Phi}$ respectively, with respect to the interior directions. In addition, the mappings are linear in the interior directions, guaranteeing that the number of integration points required is lower than other options, such as transfinite mappings [99].

Given these parametrizations, it is possible to perform the numerical integration over all the curved tetrahedral subdomains that form $\Omega_{\mathrm{B}}^{\mathtt{Phys}}$.

To this end, we consider tensor products of triangle quadratures [100] and one-dimensional Gaussian quadratures for the tetrahedrons with a face on the boundary of the physical domain (see Figure 3.10).

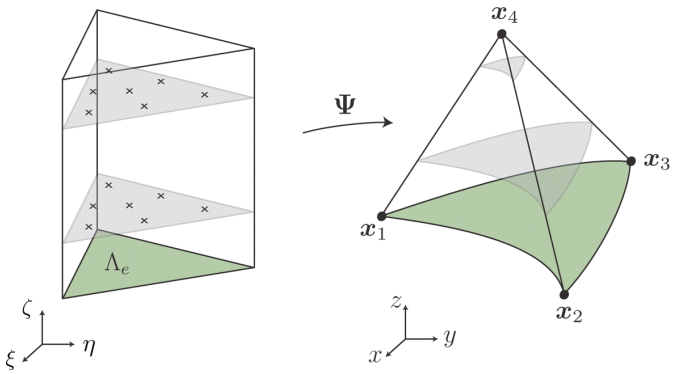


Figure 3.10: Integration over a curved tetrahedron with a face over the physical domain. Extracted from [96].

Tensor products of one-dimensional Gaussian quadratures are employed for the tetrahedra with an edge on the boundary of the physical domain.

It is important to note that NEFEM defines the tetrahedral faces of curved tetrahedra on the parametric space of the NURBS, usually as straight-sided triangles, whereas in the current approach the tetrahedral faces are defined as the intersection of a Cartesian plane and a NURBS in the physical space. This means that the tetrahedral faces in the parametric space of the NURBS are usually triangles with curved edges. It should be noted that the mapping depicted in Figure 3.10 is still valid for performing the numerical integration, even if the boundary face in the parametric space is a curved triangle.

Illustrative example. *The error associated to the proposed strategy to perform the numerical integration of polynomial functions over NURBS surfaces is studied. To do so, we 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.*

Let us consider a sphere of unit radius embedded in a coarse mesh with only eight Cartesian elements, similar to the one depicted in Figure 3.1. Let S be the surface integral of a polynomial function f defined as

$$S = \int_\Gamma f(x,y,z)\mathrm{d}\Gamma \tag{3.1}$$

where $\Gamma = \left\{(x,y,z) \mid x,y,z \geq 0,\ x^2+y^2+z^2 = 1\right\}$ represents the surface of the sphere's first octant. The numerical result computed with the strategy proposed, $S_h(f)$, is compared to the analytical result $S_e(f)$. The accuracy is evaluated by defining the relative error in percentage as $\eta_G = 100 \times \left(S_e(f) - S_h(f)\right)/S_e(f)$. To test the performance of the proposed approach we consider constant, linear and quadratic f functions. These types of functions are relevant because 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

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

$$S_e(f=x^2) = S_e(f=y^2) = S_e(f=z^2) = \frac{\pi}{6},$$

$$S_e(f=xy) = S_e(f=xz) = S_e(f=yz) = \frac{1}{3}.$$

Figure 3.11 shows the error percentage of the numerical integration of the constant, linear and quadratic functions. 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 24 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 3.12a. If we increase the number of integration points to 224 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 3.12b. 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 $f(x, y, z) = y$.

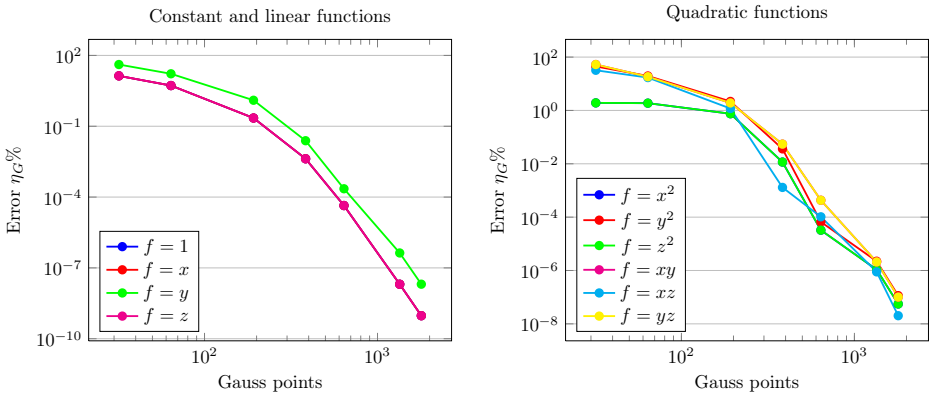
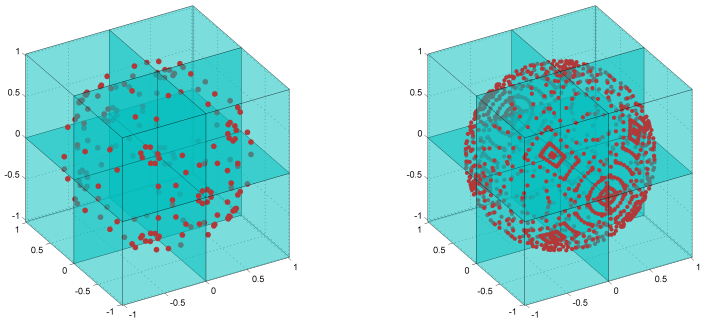


Figure 3.11: Integration error on the surface integral.

Regarding quadratic functions, it can be observed that with 24 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 224 in each element (i.e., 1792 integration points in total) the error in the numerical integration is of the order of 2×10^{-6}% or lower.

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. The number of integration points for interior elements is chosen a priori to be the minimum number required to exactly compute the integrals of the weak formulation.

In addition to this example, the reader can find in Paper A - Section 5.1. several numerical test demonstrating the accuracy of the methodology in the immersed boundary framework.



(a) 8 elements and 192 Gauss points.

(b) 8 elements and 1792 Gauss points.

Figure 3.12: Examples of the mesh used to evaluate the integration error with different number of quadrature points on the surface.

3.2. Essential boundary conditions

In practice, the problem with the Lagrange multipliers formulation (Equation (2.11)) is that most natural choices of the Lagrange multiplier field do not fulfill the *InfSup* condition because they introduce too many constraints. The idea behind stabilized methods is to impose additional conditions on the Lagrange multipliers without modifying the problem solution, at least in the limit, when the element size approaches zero, in order to have more freedom to choose the Lagrange multiplier field.

The Nitsche method can be derived from the following stabilized Lagrangian:

$$\mathscr{L}_N\left(\mathbf{v}^h, \boldsymbol{\mu}^h\right) = \mathscr{L}\left(\mathbf{v}^h, \boldsymbol{\mu}^h\right) - \frac{1}{2}\frac{h}{k}\int_{\Gamma_D} \left\|\boldsymbol{\mu}^h + \boldsymbol{\sigma}(\mathbf{v}^h)\mathbf{n}\right\|^2 d\Gamma \tag{3.2}$$

where $\mathscr{L}\left(\mathbf{v}, \boldsymbol{\mu}\right)$ was defined in Equation (2.8), h is the element size and k is a positive constant having the units of the Young's modulus. In order to use a dimensionless algorithmic constant we define $k = \kappa C_E$, using the constant defined in Equation (2.4) depending on the material properties.

The saddle point of the Lagrangian (3.2) is to find $[\mathbf{u}^h, \boldsymbol{\lambda}^h] \in \mathscr{U}^h \times \mathscr{M}^h$ such that:

$$\begin{aligned} &a\left(\mathbf{u}^h, \mathbf{v}^h\right) + b\left(\boldsymbol{\lambda}^h, \mathbf{v}^h\right) - \frac{h}{k}\int_{\Gamma_\mathrm{D}} \left(\boldsymbol{\lambda}^h + \boldsymbol{\sigma}(\mathbf{u}^h)\mathbf{n}\right)\cdot\boldsymbol{\sigma}(\mathbf{v}^h)\mathbf{n}\, d\Gamma = c\left(\mathbf{v}^h\right) \\ &b\left(\boldsymbol{\mu}^h, \mathbf{u}^h\right) - \frac{h}{k}\int_{\Gamma_\mathrm{D}} \boldsymbol{\mu}^h\cdot\left(\boldsymbol{\lambda}^h + \boldsymbol{\sigma}(\mathbf{u}^h)\mathbf{n}\right)\, d\Gamma = b\left(\boldsymbol{\mu}^h, \mathbf{g}\right) \end{aligned} \tag{3.3}$$

$\forall[\mathbf{v}^h, \boldsymbol{\mu}^h] \in \mathscr{U}^h \times \mathscr{M}^h$. Stenberg [101] shows that, for a suitable choice of the multiplier space in L^2, the Lagrange multiplier field can be eliminated element by element from Equation (3.3) to obtain the classical formulation of Nitsche's method, which has been widely used in the context of immersed boundary mesh to solve interface problems (see for example [51, 52, 58, 59, 102]). However, the original Nitsche's method has some limitations when imposing Dirichlet boundary conditions, as has been pointed out in the bibliography [46, 53, 54, 55]. The optimal convergence rate of the finite element solution can only be achieved if the norm of the tractions on the contour can be bounded by the energy norm, i.e.

$$\left\|\boldsymbol{\sigma}(\mathbf{v}^h)\mathbf{n}\right\|_{L^2,\Gamma_\mathrm{D}} \leq \frac{C_N}{h_e}\|\mathbf{v}^h\|_E \tag{3.4}$$

with a constant C_N independent of the element size. In the case of immersed boundary meshes, in general, C_N cannot be bounded as the mesh is refined. To illustrate this problem, Figure 3.13 shows an element of a 2D mesh cut by the boundary of the problem domain. The shaded part indicates the internal area of the element Ω^e. If the boundary comes closer to the element edge as the mesh is refined, the size of the area Ω^e is reduced faster than the size of the boundary Γ_D^e, and the expression (3.4) cannot be fulfilled with bounded values of C_N. High values of C_N increase the condition number of the system [55] and tend to overweight the boundary terms with respect to the domain energy, thus resulting in a finite element solution with a large error [54] (see numerical examples in Paper B - Section 7).

In this Thesis we propose the method derived from the following Lagrangian (see Paper B):

$$\mathscr{L}_S\left(\mathbf{v}^h, \boldsymbol{\mu}^h\right) = \mathscr{L}\left(\mathbf{v}^h, \boldsymbol{\mu}^h\right) - \frac{1}{2}\frac{h}{k}\int_{\Gamma_\mathrm{D}} \left\|\boldsymbol{\mu}^h + \boldsymbol{T}(\hat{\mathbf{u}}^h)\right\|^2 d\Gamma \tag{3.5}$$

where $\boldsymbol{T}(\hat{\mathbf{u}}^h)$ is the smoothed traction that depends on the finite element solution computed from a previous iteration (or mesh[103]), $\hat{\mathbf{u}}^h$. The penalty constant can be defined again as $k = \kappa C_E$.

In our formulation we use the recovered tractions on Γ_D evaluated from the recovered stress field $\boldsymbol{\sigma}^*$[104] to stabilize, solving the problem iteratively by updating the stress field value [103, 105], $\boldsymbol{\sigma}^*(\hat{\mathbf{u}}^h)$ the FE recovered stress field being calculated for an FE solution from a previous iteration (or mesh) $\hat{\mathbf{u}}^h$. The traction on the boundary is defined as $\mathbf{T}(\hat{\mathbf{u}}^h) = \boldsymbol{\sigma}^*(\hat{\mathbf{u}}^h)\cdot\mathbf{n}$ where $\mathbf{n}$ is the unit vector normal to the boundary.

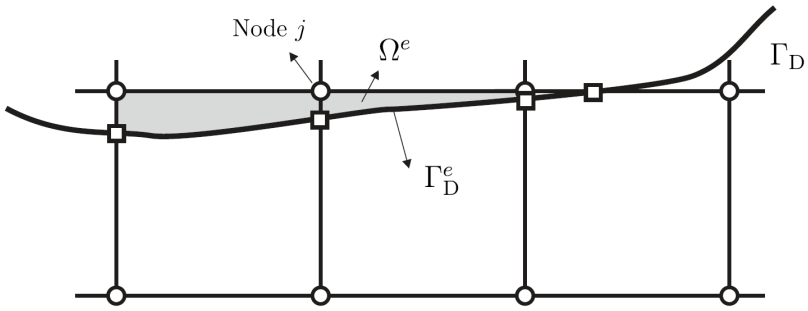


Figure 3.13: Cartesian element intersected by the geometry leading to a small area.

In the Thesis we use the traction computed from a recovered stress field as the stabilization term. The method follows the ideas of extending the solution from the internal elements to the boundary elements [53, 106], but using the stress field instead of the displacement field. The smoothed stress field is obtained from the stresses calculated by the finite element method in the boundary elements and adjacent elements using the concept of the Superconvergence Patch Recovery [107]. The aim of this choice is to avoid the problems arising from the condition of Equation (3.4). This method allows the optimal convergence rate to be obtained for predefined bounded values of the penalty constant k, regardless of the boundary cutting pattern.

The proposed formulation can be simplified by eliminating the Lagrange multipliers and obtaining a modified penalty method: Find the displacement field $\mathbf{u}^h \in \mathscr{U}^h$ such that

$$\begin{aligned} a\left(\mathbf{u}^h, \mathbf{v}^h\right) + \frac{k}{h}\int_{\Gamma_{\mathrm{D}}} \mathbf{u}^h \cdot \mathbf{v}^h \, d\Gamma = \\ c\left(\mathbf{v}^h\right) + \frac{k}{h}\int_{\Gamma_{\mathrm{D}}} \mathbf{g} \cdot \mathbf{v}^h + \int_{\Gamma_{\mathrm{D}}} \boldsymbol{T}(\hat{\mathbf{u}}^h) \cdot \mathbf{v}^h \, d\Gamma \qquad \forall \mathbf{v}^h \in \mathscr{U}^h \end{aligned} \tag{3.6}$$

The second term on the left hand side of (3.6) is a penalty term with a constant k/h. The last term on the right hand side is the virtual work of reaction forces. This term acts as a correction of the penalty term and is necessary for the finite element solution to converge to the exact solution of the problem when the mesh is refined.

As the traction field $\boldsymbol{T}$ depends on the finite element solution, an iterative procedure is proposed to solve the problem. In the first iteration, we solve the problem assuming that $\boldsymbol{T} = 0$. Then the smooth stress field is calculated. This stress field is used to update $\boldsymbol{T}$ in Equation (3.6) in order to solve the next iteration. This process runs until convergence is achieved.

The problem (3.6) can be written in matrix form as:

$$\bigcup_{e=1}^{n_e} \left(\mathbf{k}^e + \mathbf{k}_D^e\right) \{\mathbf{u}^e\} = \bigcup_{e=1}^{n_e} \left(\mathbf{f}_q^e + \mathbf{f}_g^e + \mathbf{f}_s^e\right) \tag{3.7}$$

The advantages of the proposed method are:

- As in Nitsche's method, the unknowns of the problem in the proposed formulation are the displacement degrees of freedom since the multipliers are condensed. Thus the size of the problem is not increased.
- Compared to Nitsche's method, fewer integral terms are needed to compute the system. The proposed method is stable and convergent, in spite of the absence of these terms (see Paper B).
- The method is stable for a mesh-independent bounded value of the penalty constant κ. In Paper B - Section 6 we obtain the value of this constant for 8-node tri-linear and 20-node tri-quadratic elements.
- The proposed method can be directly applied to solve problems with non-linear constitutive material behavior. In this case, Equation (3.6) is still valid if we replace the bilinear form $a\left(\mathbf{u}^h, \mathbf{v}^h\right)$ by the virtual work of internal forces.
- The proposed method provides the optimal convergence rate of the FE solution. The convergence of the finite element solution is analyzed in detail in Paper B.

The obvious drawback is that multiple iterations are needed to get the solution. However, this disadvantage can be minimized taking into account that the matrix to be solved for each iteration is always the same for linear problems (since it is only mesh dependent). Therefore it is only necessary to factorize the matrix once and perform the backward substitution at each iteration.

The global stiffness matrix is obtained by the contribution of the classical stiffness matrix of each element $\mathbf{k}^e$ and a stabilization term $\mathbf{k}_D^e$ for all the boundary elements containing the Dirichlet boundary.

The stabilization term is computed as:

$$\mathbf{k}_D^e = \int_{\Gamma_D^e} \frac{\kappa^*}{h} \mathbf{C}^T \mathbf{C} |\mathbf{J}| \mathrm{d}\Gamma \tag{3.8}$$

where

Γ_D^e is the portion of the Dirichlet boundary within the element,

κ^* is the penalty constant, being $\kappa^* = \kappa \cdot C_E$ and $\kappa > 0$,

h is the element size,

C is the matrix of finite element interpolation if Dirichlet conditions are applied on the three displacement components x, y and z.

$$\mathbf{C} = \mathbf{N} = \begin{bmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 & \dots & N_{nnod} & 0 & 0 \\ 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & \dots & 0 & N_{nnod} & 0 \\ 0 & 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & \dots & 0 & 0 & N_{nnod} \end{bmatrix}$$

with *nnod* as the number of nodes per element. Otherwise $\mathbf{C} = \mathbf{SN}$, where $\mathbf{S}_{ii} = \sum_d \delta_{id}$ would be a diagonal matrix, d is the direction in which Dirichlet boundary conditions are applied and δ is the Dirac delta function.

On the other side of the equation, the equivalent force vector $\mathbf{f}$ is evaluated by adding the contribution of the standard FE vector of equivalent forces on nodes $\mathbf{f}_q$, the stabilization term of the Dirichlet boundary $\mathbf{f}_g$ and the stabilizing stress component $\mathbf{f}_s$.

The vector $\mathbf{f}_g$ is due to the non-homogeneous Dirichlet condition $\mathbf{u}^h = \mathbf{g}$ on Γ_D and it is evaluated assembling the contribution of every element on the Dirichlet boundary:

$$\mathbf{f}_g^e = \int_{\Gamma_D^e} \frac{\kappa^*}{h} \mathbf{C}^T g |\mathbf{J}| \mathrm{d}\Gamma \tag{3.9}$$

Finally, $\mathbf{f}_s$ is the stabilizing term which depends on the stress field. As mentioned above, in our formulation we use the recovered tractions on Γ_D evaluated from the recovered stress field $\boldsymbol{\sigma}^*(\hat{\mathbf{u}}^h)$[104] to stabilize, $\hat{\mathbf{u}}^h$ being an FE solution from a previous iteration (or mesh). The traction on the boundary is defined as $\mathbf{T}(\hat{\mathbf{u}}^h) = \boldsymbol{\sigma}^*(\hat{\mathbf{u}}^h) \cdot \mathbf{n}$ where $\mathbf{n}$ is the unit vector normal to the boundary, then

$$\mathbf{f}_s^e = \int_{\Gamma_D^e} \mathbf{C}^T \mathbf{T}(\hat{\mathbf{u}}^h) |\mathbf{J}| \mathrm{d}\Gamma \tag{3.10}$$

Illustrative example. *In this section we illustrate the capabilities of the proposed formulation and explore the limitations of the different methods. A numerical example with exact solution is solved and used to check: a) the convergence rate of the finite element solution as the mesh is refined and, b) the effect of the stability constant k on convergence. It was also used to compare the proposed method with Nitsche's method showing that, in general, both methods have similar behavior. Linear ($\mathcal{L}_8$) and quadratic ($\mathcal{Q}_{20}$) elements are used in the examples.*

We consider a tilted hexahedron subjected to 4^{th} *order polynomial displacements and plain strain conditions. The Young's modulus and the Poisson's ratio are* $E = 1000$ *and* $\nu = 0.3$ *respectively. The exact displacement solution reads as:*

$$\begin{aligned} u_x &= -\frac{25}{192} + \frac{75}{64}x^2 - \frac{25}{24}x^4 - \frac{25}{4}y + \frac{25}{4}x^2y\frac{25}{8}y^2 + \frac{25}{8}x^2y^2 \\ u_y &= \frac{20}{3}x + \frac{65}{12}x^3 + \frac{65}{12}x^3y - 10xy^2 - \frac{10}{3}xy^3 \\ u_z &= 0 \end{aligned} \tag{3.11}$$

The exact expression of the stress tensor, obtained by using the constitutive relation can be found in Paper B - Section 7. The known values of the displacements were imposed as Dirichlet boundary conditions on the entire external surfaces of the domain.

Figure 3.14 shows the exact geometry of the problem embedded in the Level 3 Cartesian grid. The same figure also shows the surface triangulation used to numerically evaluate the contour integrals.

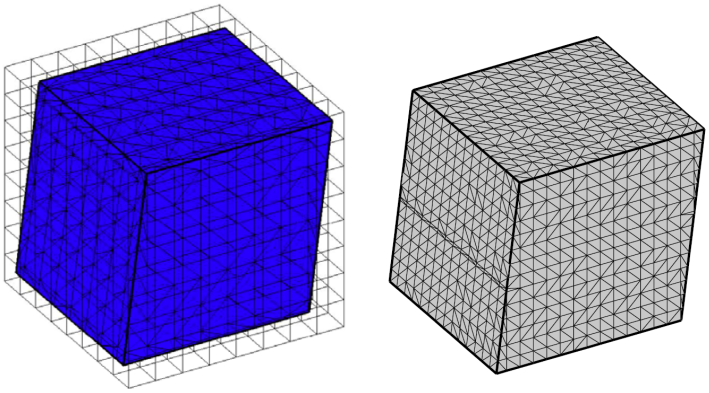


Figure 3.14: Tilted hexaedron. Left: Geometry embedded in a Level 3 Cartesian mesh. Right: Triangulation of the surface in a Level 4 Cartesian mesh.

The problem was solved using a sequence of Cartesian meshes obtained by element subdivision starting from a Level 2 mesh having 64 hexahedral elements (Level 0 has a single element and Level 1 has eight elements).

In order to test the influence of the stability constant on the convergence of the proposed iterative method and on the discretization error, the same mesh was analyzed with κ *ranging from* 0.04 *to* $4 \cdot 10^6$*. We also considered Nitsche's method in the convergence analysis for comparison purposes.*

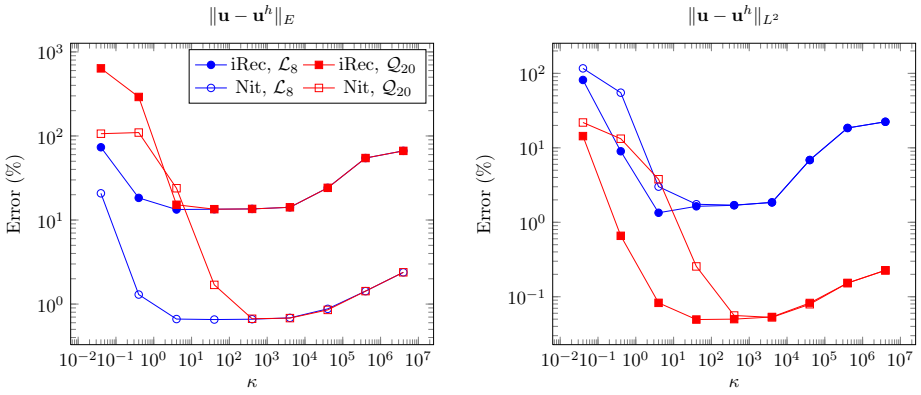


Figure 3.15: Tilted hexaedron. Discretization error in energy norm and L^2-norm (%) for different values of the penalty constant κ. Results for Level 2 Cartesian mesh using linear and quadratic elements. The proposed method is denoted as iRec and Nitsche's method as Nit.

The energy norm of the error is plotted in Figure 3.15 for different values of the penalty constant considering Level 2 meshes (similar behavior was obtained for the rest of the mesh levels), and compared with the results obtained with Nitche's method.

It can be seen that the proposed technique gave a wide range of κ values, from 4 to $4 \cdot 10^3$, for which the level of the error remains essentially unaffected for both $\mathcal{L}_8$ and $\mathcal{Q}_{20}$ elements. However, the values of κ for which the error level remains unaffected is narrower in Nitche's method, ranging from only $\kappa = 40$ to $\kappa = 4 \cdot 10^3$ for $\mathcal{L}_8$ elements and from $\kappa = 400$ to $\kappa = 4 \cdot 10^3$ for $\mathcal{Q}_{20}$ elements. It can also be observed that both techniques provide similar error levels for high levels of κ ($\kappa \geq 40$ for $\mathcal{L}_8$ elements and $\kappa \geq 400$ for $\mathcal{Q}_{20}$ elements).

In Figure 3.16 the energy and the L^2 norms of the discretization error are plotted as a function of the mesh size for $\mathcal{L}_8$ and $\mathcal{Q}_{20}$ elements and, for both, the proposed technique (iRec) and Nitsche's method (Nit). The triangles in this figure show the theoretical optimal convergence rate that can be achieved. Three different values of κ were considered, taking into account the theoretical value $\kappa > C_p C_r$ that provides optimal convergence ($C_p = 13$ for $\mathcal{L}_8$ and $C_p = 21$ for $\mathcal{Q}_{20}$ elements, assuming that $C_r = 1$), $\kappa = 4$ and $\kappa = 400$.

As $\kappa = 4$ is lower than $C_p C_r$, the optimal convergence rate is not ensured. Indeed it is only achieved using the proposed method for $\mathcal{L}_8$ elements. The expected behavior in terms of convergence rate is observed in the numerical results for $\kappa = 400$ using the proposed method both for linear and quadratic elements, whereas Nitsche's method was only able to recover the optimum convergence rate for linear elements. A reduction of the convergence rate for Nitsche's method can be seen in the last refinement step

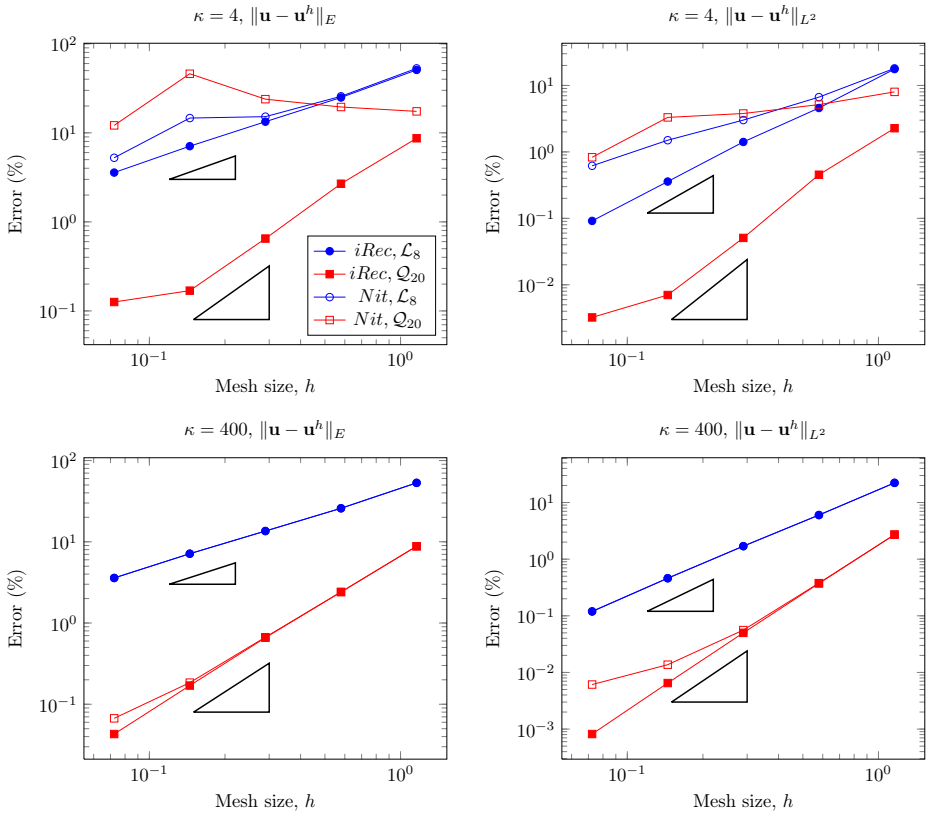


Figure 3.16: Discretization error in energy norm as a function of the mesh size for $\mathcal{L}_8$ and $\mathcal{Q}_{20}$ elements. The triangles show the optimal convergence rate.

for $\mathcal{Q}_{20}$ elements. Nitsche's method would therefore require a κ higher than 400 to be able to successfully recover the optimum convergence rate during the entire refinement process for $\mathcal{Q}_{20}$ elements.

This example showed that, if a sufficiently high value of κ is used, the proposed technique provides results similar to those obtained with Nitsche's method. However, it is able to provide accurate results, with the optimal convergence rate, for considerably lower values of the stabilization parameter κ than Nitsche's method.

In addition to this example, the reader can find in Paper B - Section 7 additional numerical test demonstrating the accuracy of the methodology. For instance, an example shows the behavior of the proposed method, without any modifications, when elasto-plastic behavior of the material is considered. This example is particularly

interesting as, to our knowledge, Nitsche's method has not been used to solve these type of problem because the formulation of the method required for plasticity has not been derived.

3.3. h-adapted meshing

This Thesis proposes 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 proposed h-adaptive finite element analysis 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 the maximum refinement difference between two elements adjacent by face or edge is limited to one level, and Multipoint constraints (MPCs)[108, 109] 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.

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 parameters 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.

3.3.1. Geometrical refinement

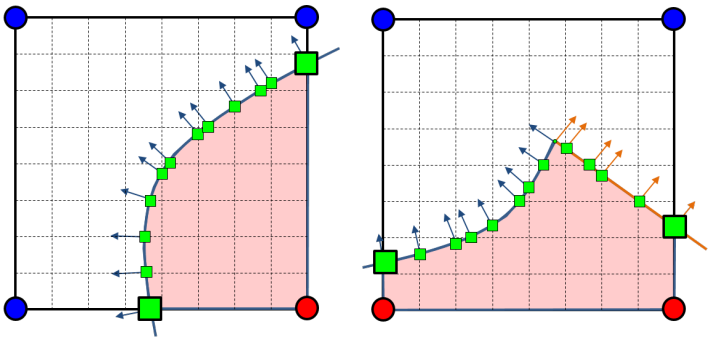
The refinement based on the features of the geometrical model is widely used in FEA, since the user can easily identify where the mesh should be finer to properly capture the boundaries of the models. For any kind of geometrical representation, from tesselations to advanced parametric surfaces, it is 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. This parameter is either set by the user or will be the initial level plus 3. 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 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 3.17 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 3.17a 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 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 3.17b 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



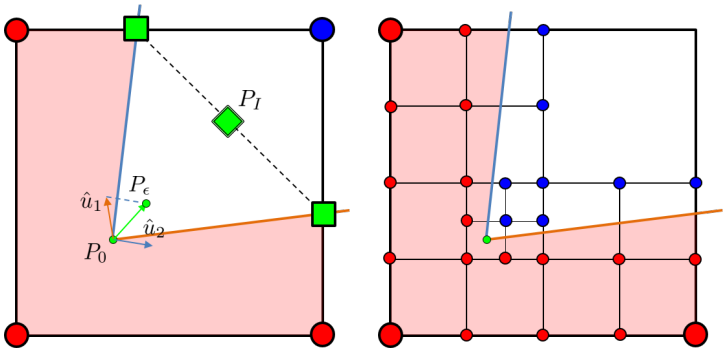
(a) Smooth boundary. (b) Sharp edge.

Figure 3.17: Geometrical features within elements. Variation of unit vectors.

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 it is key to obtaining good discretization of the mesh along the boundary.

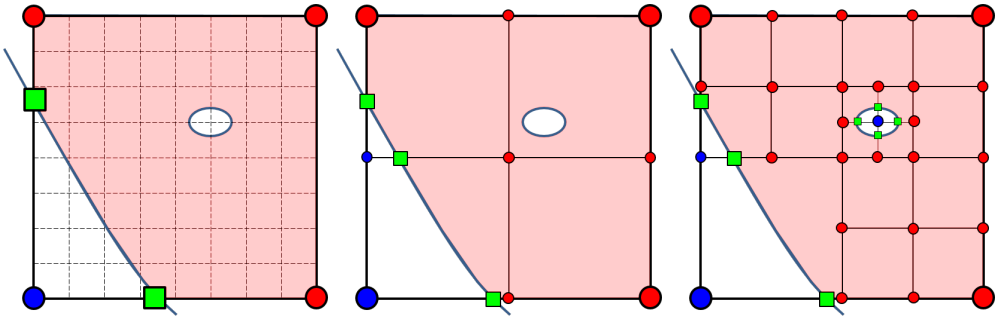
The next criterion completes the previous one and is related to the nature of the problem. FEAVox was first implemented to solve linear elasticity problems in which some internal corners could originate singularities and produce large gradients when evaluating displacements or stresses. These cases require an adequate discretization around the singularities to obtain a better representation of the singular solution. In Figure 3.18a 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 $P_\epsilon = P_0 + \epsilon$, ϵ being 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 3.18b). In 3D this evaluation will occur at several points along the intersections curve between surfaces.

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 3.19a 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 3.19b. To detect these small entities we have to remember that it is easy to find out if we are using the intersections of all the geometrical entities in the actual mesh and to locate points in the Cartesian elements by means of our hierarchical data structure.



(a) Measure of distances. (b) Example of refinement.

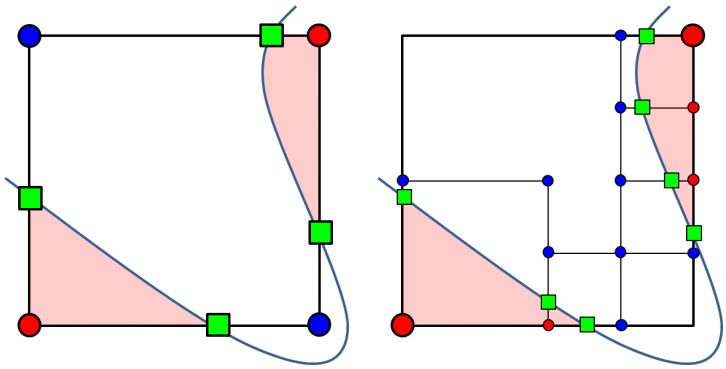
Figure 3.18: Identification of singularities.



(a) Coarse element. (b) First step of refinement. (c) Final step of refinement.

Figure 3.19: Detection of small features.

The last refinement criterion comes naturally with the mesh generation strategy implemented. As we explained in Section 3.1.2, only 7 of the 14 configurations of the original Marching Cubes algorithm were taken into account because they refer to non-ambiguous configurations. In [43] 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 3.20a). 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 3.20b.



(a) 2D projection of the pattern. (b) Solution through refinement.

Figure 3.20: Undefined pattern and elimination through refinement.

3.3.2. 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

$$\|\mathbf{e}\| = \sqrt{\int_{\Omega} (\boldsymbol{\sigma}_h - \boldsymbol{\sigma})^T \mathbf{D}^{-1} (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}) \mathrm{d}\Omega} \tag{3.12}$$

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

The error at each element can be evaluated by integrating (3.12) 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, in which M is the total number of elements in the mesh, relates the global and local errors

$$\|\mathbf{e}\|^2 = \sum_{i=1}^{M} \|\mathbf{e}^{(i)}\|^2 \tag{3.13}$$

Several types of error estimators have been proposed in the literature depending on the obtaining procedure: residual error estimators [110, 111, 112, 113] use the residuals of the approximate solution to evaluate the error, the Constitutive Relation Error (CRE) [114] consisting of the comparison of statically admissible stress fields with

kinematically admissible stress fields, the estimators based on dual analysis [115, 116] and making use of two solutions of the problems. Finishing the classification, we find the recovery based error estimators. Proposed by Zienckevick and Zhu [13], these estimators use a recovered solution, $\boldsymbol{\sigma}^*$, instead of the exact solution $\boldsymbol{\sigma}$ to measure the error [107, 117].

Assuming 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 (3.13), for the i^{th}-element, as

$$\|\mathbf{e}_{es}^{(i)}\| = \sqrt{\int_{\Omega^{(i)}} (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^*)^T \mathbf{D}^{-1} (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^*) \mathrm{d}\Omega} \tag{3.14}$$

where $\boldsymbol{\sigma}^*$ is a smoothed continuous stress field obtained using a 3D version of the recovery technique presented in [42, 104].

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 [13, 118] in which the optimality criterion is that of equidistributing the error on the elements of the new mesh. In [118] 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:

$$h_n^{(i),n-1} \approx h_{n-1}^{(i)} \left[\frac{1}{M_{n-1}}\right]^{1/2(p+1)} \left[\frac{\|\mathbf{e}\|_n}{\|\mathbf{e}\|_{n-1}}\right]^{\frac{d}{2p^2+pd}} \left[\frac{\|\mathbf{e}\|_n}{\|\mathbf{e}^{(i)}\|_{n-1}}\right]^{\frac{2}{2p+d}} \tag{3.15}$$

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 $n-1$,

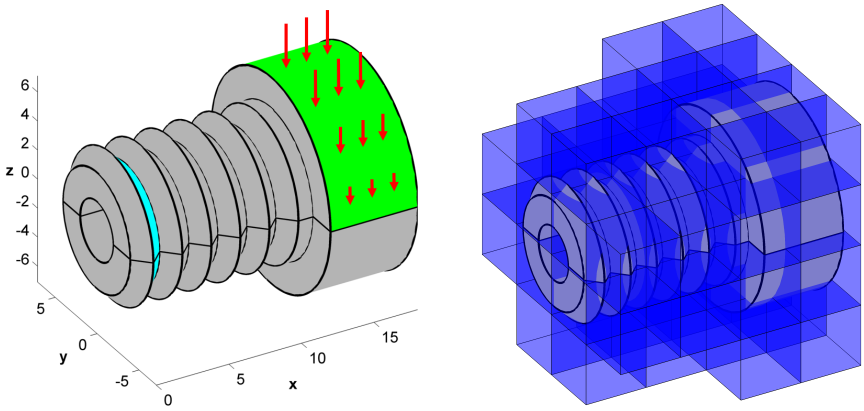
p is the polynomial degree of the shape functions used,

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

Replacing $\|\mathbf{e}^{(i)}\|$ in (3.15) by the estimation given in (3.14) 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 Paper C - Appendix.

Illustrative example. *With this problem our purpose is to show the performance of the h-adaptive geometrical refinement process in complex geometries. Naturally, in this type of problems 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.*

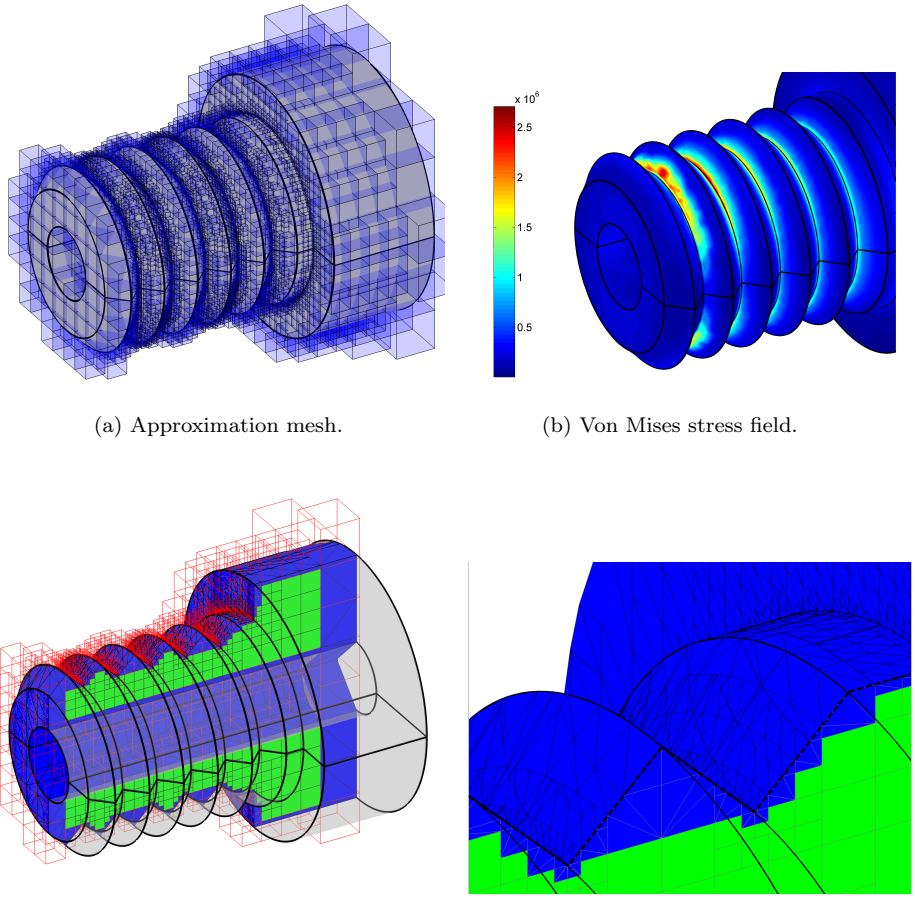
The model selected represents a perforated screw, as shown in Figure 3.21, with a topology as used in hydraulic applications. In this case, we restrained the displacements of the surfaces in blue and applied a variable vertical force per unit of area. The material was steel with Young's modulus $E = 2,1 \cdot 10^9 Pa$ *and Poisson's ratio* $\nu = 0,333$.



(a) CAD model and boundary conditions. (b) Initial coarse mesh.

Figure 3.21: Model of a hydraulic screw.

Figure 3.21b 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 3.22a 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 3.22b 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.



(a) Approximation mesh.

(b) Von Mises stress field.

(c) Section of the mesh.

(d) Detail of the integration subdomains.

Figure 3.22: Hydraulic screw. Geometrical h-refinement.

To make clear how boundary elements are treated, Figure 3.22c 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 3.22d 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 3.1 summarizes the topological features of the meshes used for this analysis. The first mesh is the geometrically refined mesh shown in Figure 3.22d and the second mesh corresponds to the one obtained from the error-based refinement, see Figure

3.23b. 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

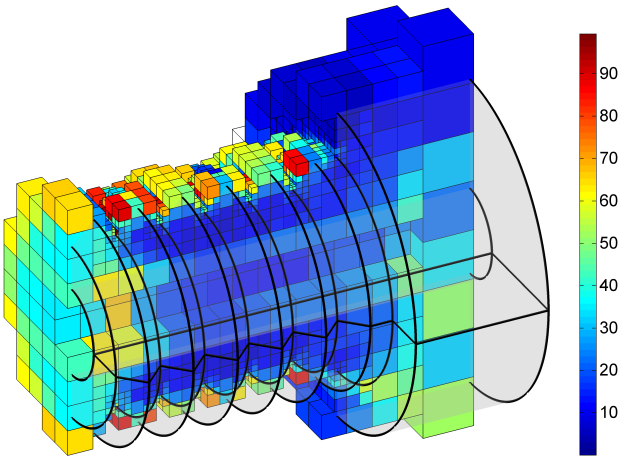
Table 3.1: Topology of the approximation meshes in terms of different types of elements and subdomains.

The global estimated error in energy norm for the first mesh is 20.86%*. Figure 3.23a 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 3.23b. We can observe higher density of elements in this mesh compared to the first one analyzed. Figure 3.23c 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.*

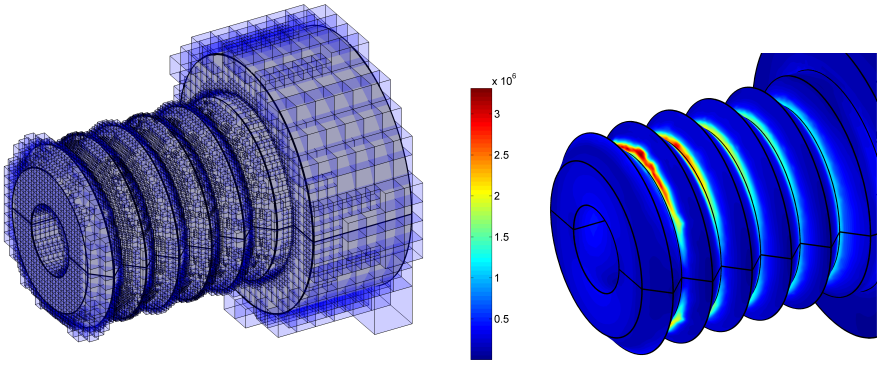
In Paper C - Section 6 the reader can find the convergence analysis with h-adapted meshes.

3.4. Shape sensitivities and optimization

The algorithms for the solution of (2.15) are, normally, iterative. We will mainly focus in this Thesis on the gradient-based optimization algorithms because of their fast convergence to the optimal solution. As indicated in Section 2.3, these methods require the computation of the objective function, the constraints and their derivatives (sensitivities) with respect to the design variables, $\mathbf{a}$, for each geometry considered during the process.



(a) Element-wise error estimation.



(b) Approximation mesh.

(c) Von Mises stress field.

Figure 3.23: Hydraulic screw. Error-based h-refinement.

We propose to adapt to cgFEM a shape sensitivity analysis methodology, exploiting the features of our embedded methodology, aiming for the efficient calculation of sensitivities to reduce the computational cost of the optimization process. We choose a discrete semi-analytical approach. This specification means that some of the discrete derivatives rely on analytical derivation and some on finite difference approximations, i.e. to differentiate the nodal locations with respect to design variables

(velocity fields), which is a challenging issue considering the immersed nature of the cgFEM.

3.4.1. Design velocity fields

Shape sensitivity analysis requires the evaluation of the so called design velocity field $\mathbf{V}_m$:

$$\mathbf{V}_m = \frac{\partial \mathbf{p}}{\partial a_m} \tag{3.16}$$

$\mathbf{p} = \mathbf{p}(\mathbf{a})$ being the position of an arbitrary point expressed as a function of the design variables $\mathbf{a}$. While the numerical solution and sensitivity analysis are defined on the whole domain $\Omega_{\texttt{Phys}}$, the velocity fields are defined only on the boundary of the domain and there is no closed conformation of this field in the interior.

Theoretically, the velocity field should have the same regularity as the displacements field and depend linearly on the alteration of the design variables. In practical terms, different applications can also impose certain practical additional requirements on the velocity field, such as the need to maintain the mesh topology, to provide FE nodes necessarily located on the boundary of the domain, to produce non-distorted meshes, to be naturally related with the design parameters of the CAD models or to be efficient and general.

Magnitudes like the sensitivity of the strain energy are not affected by the values of the velocity fields in the interior of the domain, provided the velocity fields meet the theoretical requirements and the exact structural response is used in the evaluation of this sensitivity. However, in practice, the FE approximation will be used instead of the exact structural response. As a consequence, the final sensitivity of the strain energy will be affected by the velocity fields considered in the interior of the domain [119].

The velocity field is usually defined at the nodes of the FE mesh and then interpolated by the shape functions used to interpolate the displacements, so that the velocity fields and the displacements field will have equivalent regularity. In the following subsections we describe the procedures used to define the design velocity fields along the boundary and inside the domain considering cgFEM's special characteristics for developing efficient procedures for shape sensitivity analysis.

As the algorithms used in standard FEM for velocity field definition (see comparative studies in[120, 121, 122, 123]) cannot be used directly in an IBM context, we will therefore consider some alternatives to generate adequate velocity fields for a Cartesian grid framework, in the context of embedded methods.

3.4.1.1. Generation of boundary velocity fields

NURBS (Non-Uniform Rational B-Spline) curves and surfaces were used in the present study to describe the boundary of 2D and 3D domains.

NURBS surfaces are obtained from a tensor product through two knot vectors $\Xi = \{\xi_1, \ldots, \xi_{n+p+1}\}$ and $\Gamma = \{\eta_1, \ldots, \eta_{m+q+1}\}$ which define the parametric space. The $n \times m$ control points $\mathbf{P}_{i,j}$ form a control net. 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

$$\mathbf{S}(\xi, \eta) = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{N_i^{(p)}(\xi) M_j^{(q)}(\eta) w_{i,j} \mathbf{P}_{i,j}}{\sum_{i=1}^{n} \sum_{j=1}^{m} N_i^{(p)}(\xi) M_j^{(q)}(\eta) w_{i,j}} \tag{3.17}$$

Therefore, for a specific design variable a_m the calculation of the velocity field on the parametrized boundary $\mathbf{S}(\xi, \eta, \mathbf{a})$, is simple and can be expressed as:

$$\mathbf{V}_{m,\Gamma}(x(\xi,\eta), y(\xi,\eta), z(\xi,\eta)) = \frac{\partial \mathbf{S}(\xi, \eta, \mathbf{a})}{\partial a_m} = f\left(\frac{\partial \mathbf{P}(\mathbf{a})}{\partial a_m}\right) \tag{3.18}$$

The boundary velocity field on the discrete model is achieved by the evaluation of (3.18) using the parametric coordinates (ξ, η) of each surface point. The analytical evaluation of the derivatives of the NURBS and trimmed NURBS (see Subsection 3.4.1.1.1) can be cumbersome, because of the lack of explicit expressions to evaluate the control points as a function of $\mathbf{a}$, and will depend on how each CAD system generates the surfaces as a function of the parameters defined by the user. Therefore, for the sake of generality, we propose to evaluate these derivatives by the finite differences approximation contained in the following equation:

$$\mathbf{V}_{m,\Gamma} \cong \frac{\Delta \mathbf{S}(\xi, \eta, \mathbf{a})}{\Delta a_m} = \frac{\mathbf{S}(\xi, \eta, \mathbf{a} + \Delta a_m) - \mathbf{S}(\xi, \eta, \mathbf{a})}{\Delta a_m} \tag{3.19}$$

where Δa_m is a perturbation of the design variable a_m.

Equation (3.19) will be used to obtain the velocity field at the parametric coordinates (ξ, η) of the points on the boundary required by the shape sensitivity analysis.

3.4.1.1.1. Velocity field on trimmed surfaces

NURBS surfaces are four-sided patches that do not allow for the presence of holes nor the direct creation of irregular shapes. Trimming is a valuable procedure to devise complex objects. A trimmed NURBS surface consists of: a) a tensor product NURBS surface and, b) a set of properly arranged trimming curves lying within the parametric rectangle of the surface. It is useful to represent the trimming curves in NURBS form.

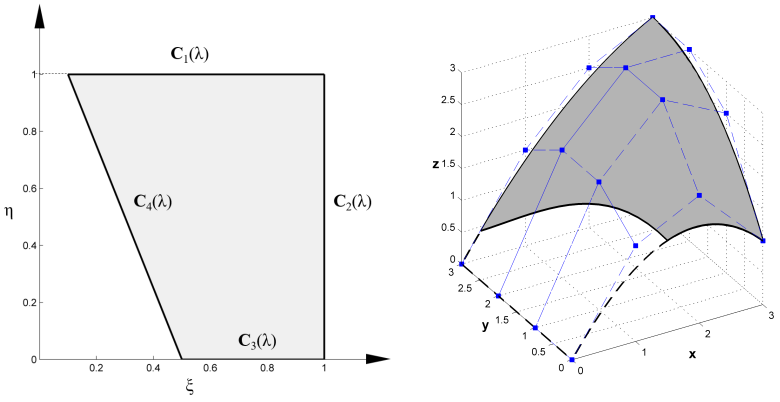
Assume that n_c NURBS curves are defined as:

$$\mathbf{C}_k(\lambda) = (\xi_k(\lambda), \eta_k(\lambda)) \qquad k = 1, 2, \ldots, n_c \tag{3.20}$$

The $\mathbf{C}_k(\lambda)$ curves are all properly oriented forming loops, which establish the boundary of the trimmed region such that, when advancing along the piecewise curve as indicated by its numbering, the real surface material is always on the same side, see Figure 3.24a. The trimmed surface boundaries are then retrieved by mapping the 2D trimming curves onto the surface. That is,

$$\mathbf{S}(\xi_k(\lambda), \eta_k(\lambda)) \qquad k = 1, 2, \ldots, n_c \tag{3.21}$$

are surface curves bounding the trimmed surface. Figure 3.24b shows the 3D mapping of the trimming loop.



(a) Trimmed parametric space. (b) Trimmed physical surface.

Figure 3.24: NURBS surface example.

The trimming procedure in general does not allow application of the previously explained procedure to evaluate the boundary velocity field. The reason for this is that, when dealing with trimmed entities, in some cases the generation of new geometries is obtained by modifying the trimming curve in the parametric space but not the parametric space itself, i.e. without modifying the control points of the surfaces, leading to $V_{m,\Gamma} = 0$ in (3.18), as in Figure 3.25.

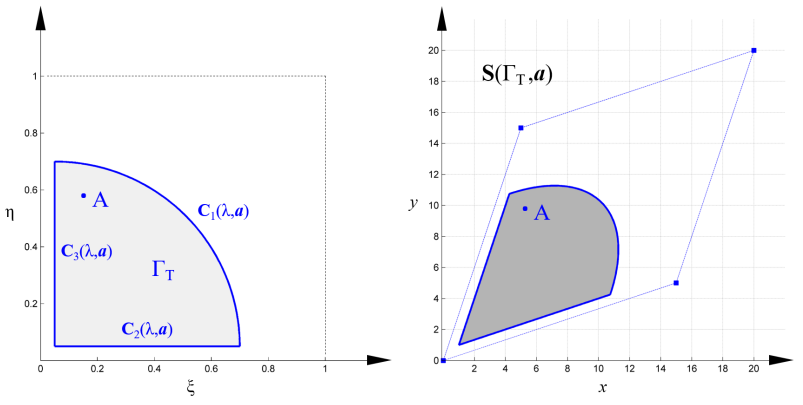
Let us consider a trimmed NURBS surface defined as:

$$\mathbf{S}(\mathbf{C}_k(\lambda, \mathbf{a}), \mathbf{a}) = \mathbf{S}(\xi_k(\lambda, \mathbf{a}), \eta_k(\lambda, \mathbf{a}), \mathbf{a}) \qquad k = 1, \ldots, n_c \tag{3.22}$$

where $\mathbf{C}$ are NURBS curves and $(\xi_k(\lambda, \mathbf{a}), \eta_k(\lambda, \mathbf{a}))$ are the surface parametric coordinates of the k trimming curve function of the parameter λ and the design variable

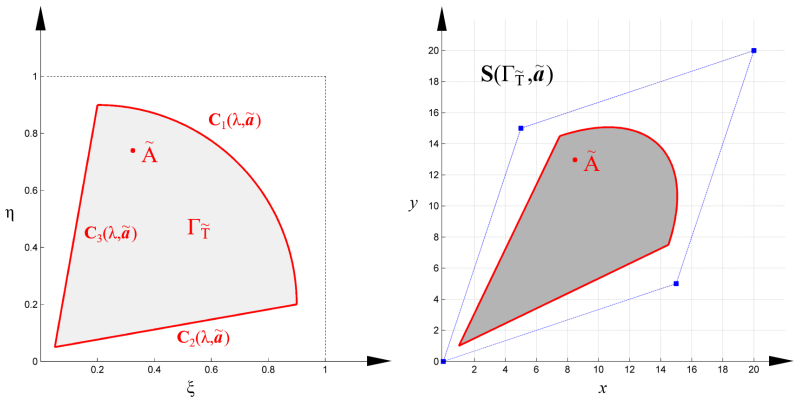
vector, **a**. In this definition, we assume that **a** can influence both the surface **S** and the trimming curves **C**.

Now let us assume a change in the design variables such that $\widetilde{\mathbf{a}} = \mathbf{a} + \Delta a_m$, where Δa_m is a small increment in a single design variable. Figure 3.25c shows an illustration of this change in the parametric space Γ_T, that leads to the new domain $\Gamma_{\widetilde{T}}$. Figures 3.25b and 3.25d show how a change in the trimming loop yields a different mapping of the subspace, while keeping the control polygon in place. However, it is still necessary to evaluate the value of the velocity field for the points in the domain represented in Figure 3.25a.



(a) Original parametric space.

(b) Original physical space.



(c) Perturbed parametric space.

(d) Perturbed physical space.

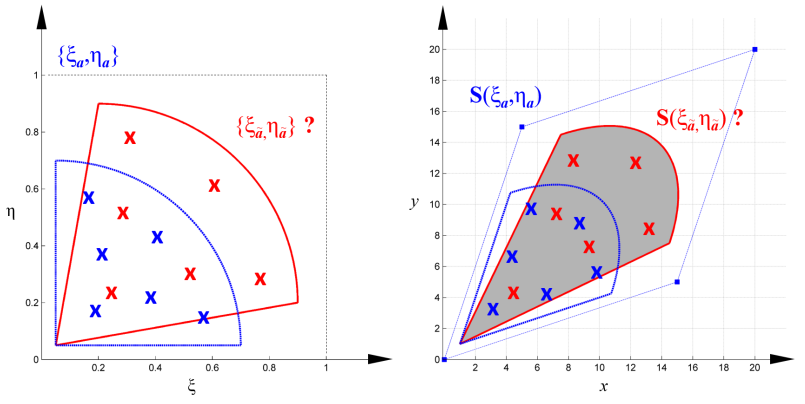
Figure 3.25: Trimmed NURBS surface example. Modifying trimming curves.

Let A in Figures 3.25a and 3.25b be a point of interest of coordinates $(\xi, \eta)_A$ in the parametric space and $(x, y)_A$ in the physical space. The perturbation of the design variable a_m will modify the trimmed surface in the parametric space and hence in the physical space. This will perturb the position of A to $\widetilde{A}$ (see Figures 3.25c and 3.25d) of coordinates $(\xi, \eta)_{\widetilde{A}}$ in the parametric space and $(x, y)_{\widetilde{A}}$ in the physical space. The evaluation of the design velocity field at A will require the evaluation of this perturbation in the physical space. We therefore need to find how $(\xi, \eta)_A$ is mapped to $(\xi, \eta)_{\widetilde{A}}$. This can be evaluated on the trimming curves $\mathbf{C}_k(\lambda, \mathbf{a})$ but we also need this information in the interior of Γ_T.

The velocity field on the trimming curve $\mathbf{C}_k$ boundary for this particular problem can be written as:

$$\mathbf{V}_{m,\mathbf{C}} = \frac{\partial \mathbf{S}\left(\mathbf{C}_k\left(\lambda, \mathbf{a}\right)\right)}{\partial a_m} \tag{3.23}$$

Equation (3.23) will evaluate the velocity field only on the trimming curves. This means that the parametric coordinates of the points (ξ, η) on Γ_T will have to be updated to consider the change of the parametric subspace that leads to $\Gamma_{\widetilde{T}}$. Figure 3.26 shows a general case in which a number of points of interest on the surface, defined by their original parametric coordinates $\{\xi_a, \eta_a\}$), should be updated to new coordinates $\{\xi_{\tilde{a}}, \eta_{\tilde{a}}\}$ to obtain a transformation consistent with the trimming curves.



(a) Parametric space. (b) Physical space.

Figure 3.26: Problem transforming points within a trimmed NURBS surface.

We adapted the idea of the physical approach [124] to obtain this update from $\{\xi_a, \eta_a\}$ to $\{\xi_{\tilde{a}}, \eta_{\tilde{a}}\}$. Hence, we propose solving an auxiliary elasticity problem with imposed displacements on the boundary. It is possible to create a 2D finite element

model in the original parametric space from information that is already at our disposal. In this case the intersections between the surface with the Cartesian axes would be the nodes and the elements would be defined by the faces of the integration subdomains on the surface (see Section 3.1). Figure 3.27a shows this proposal. The discretized system of equations of the auxiliary problem can be written as:

$$\mathbf{KP} = \mathbf{F} \tag{3.24}$$

where $\mathbf{K}$ is the stiffness matrix, $\mathbf{F}$ is the vector of equivalent nodal forces. In this case $\mathbf{F} = 0$ as Neumann boundary conditions are not applied, and $\mathbf{P}$ contains the prescribed displacements on the boundary and the unknown values of the field in the interior of the domain. These 'prescribed displacements' will correspond to the values of the change on the trimming curves coordinates such that:

$$\mathbf{P}_{m,k} = \left(\xi_k\left(\lambda, \mathbf{a} + \Delta a_m\right), \eta_k\left(\lambda, \mathbf{a} + \Delta a_m\right)\right) - \left(\xi_k\left(\lambda, \mathbf{a}\right), \eta_k\left(\lambda, \mathbf{a}\right)\right) \qquad k = 1, \ldots, n_c \tag{3.25}$$

In this way the displacements imposed are those that change the position of the trimming curves in the parametric space associated with the design variable under study.

Solving (3.24) after applying the Dirichlet boundary conditions provides the perturbation of the position of all the nodes of the mesh shown in Figure 3.27b, which can be interpolated into the elements. The result will be the position of the original points mapped into the new subspace defined by the perturbed boundary in the parametric space:

$$\left\{\widetilde{\xi}, \widetilde{\eta}\right\} = \left\{\xi_{\mathbf{a}+\Delta a_m}, \eta_{\mathbf{a}+\Delta a_m}\right\} = \left\{\xi_{\mathbf{a}}, \eta_{\mathbf{a}}\right\} + \mathbf{P}_m\left(\xi_{\mathbf{a}}, \eta_{\mathbf{a}}\right) \tag{3.26}$$

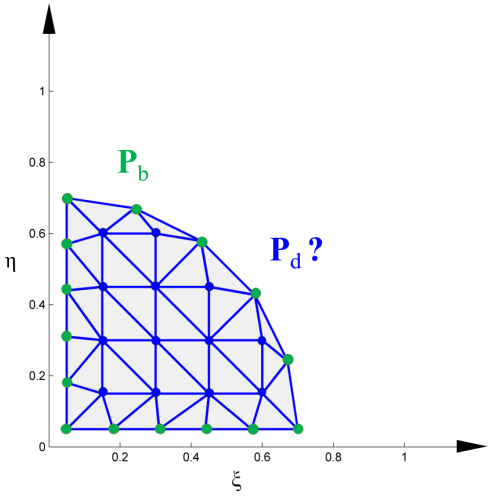
The velocity field on these surfaces will be calculated as:

$$\mathbf{V}_{m,\Gamma} \cong \frac{\Delta \mathbf{S}(\xi, \eta, \mathbf{a})}{\Delta a_m} = \frac{\mathbf{S}(\widetilde{\xi}, \widetilde{\eta}, \mathbf{a} + \Delta a_m) - \mathbf{S}(\xi, \eta, \mathbf{a})}{\Delta a_m} \tag{3.27}$$

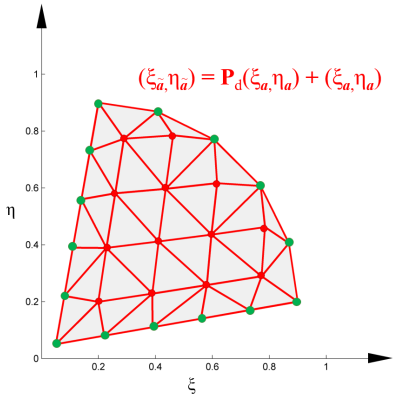
The procedure proposed to evaluate the design velocity field for these surfaces involves solving a 2D FE problem. However, the associated computational cost is low, as: a) the FE mesh used for the analysis is that of a previously evaluated triangulation of the trimmed surface in the parametric space required for intersecting the surface with the Cartesian mesh, b) the mesh is a relatively coarse 2D mesh, thus involving a low computational cost and c) the factorization of $\mathbf{K}$ obtained during the process is common to all the design variables.

3.4.1.2. Generation of domain velocity fields

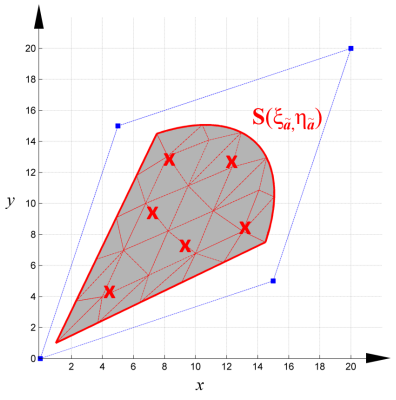
Figure 3.28a uses a 2D case to show that, using fixed Cartesian grids, it is possible to find nodes external to the domain (green dots) that will be involved in the



(a) FEM system using the parametric space of a trimmed NURBS surface.



(b) Solution in the parametric space.

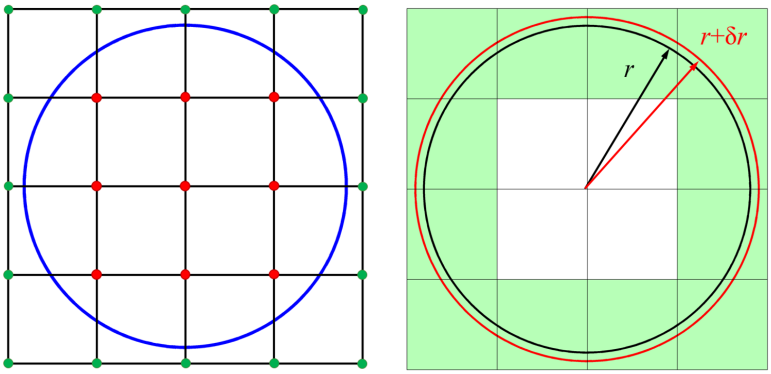


(c) Solution in the physical space.

Figure 3.27: NURBS surface example solved using the proposed strategy.

evaluation of the design velocity field. Strategies are needed to assign the velocity field both to internal and external nodes so that we can interpolate the velocity field at any point on the elements.

A perturbation of the boundary will not induce a perturbation of the Cartesian nodes internal to the surface. The velocity field will be zero in the internal elements,



(a) Existence of external nodes.

(b) Perturbation of geometry only in boundary elements.

Figure 3.28: Embedded methods and velocity fields.

thus reducing the computational cost associated with the evaluation of their volume integrals for shape sensitivity analysis, e.g. the non-shaded elements in Figure 3.28b.

We propose two different velocity field generators that will represent the geometry changes only in these elements. Both methods can be classified as boundary element methods [7], as the velocity fields will be non-zero only on the band of elements intersected by the parametrized boundary.

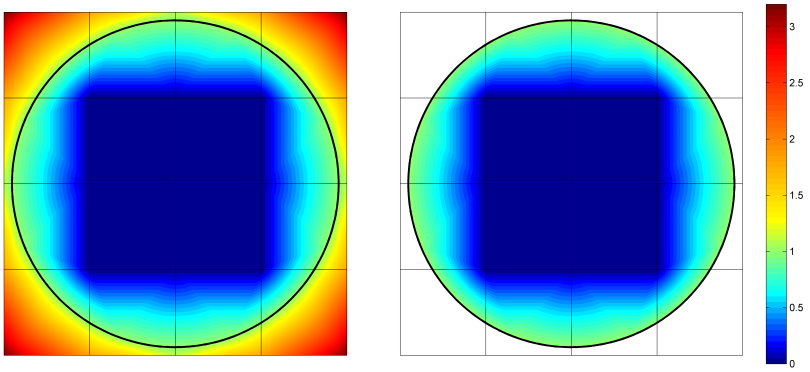
Figure 3.29 shows an example of the velocity field for the case shown in Figure 3.28 that would be obtained by the proposed methods using the radius as the design variable. Figure 3.29a shows the interpolation of the velocity field even on the external nodes, while Figure 3.29b represents the actual velocity field necessary to evaluate the integrals of the sensitivity analysis of the internal elements and the integration subdomains in the interior of the physical domain of the boundary elements.

3.4.1.2.1. Least squares approach

In this first method we use a least squares procedure to extrapolate the values to the external nodes imposing the velocity field on the boundary and the zero velocities on the internal nodes of the elements of the boundary of the layer.

An FE nodal interpolation for each component of the design velocity field is fitted into each element with the velocity field values at the surface integration points of the elements and to $\mathbf{V} = 0$ at the internal nodes of the boundary elements. By using a least square approach we obtain the linear system of equations:

$$\mathbf{M}\mathbf{V}_{m,q} = \mathbf{G}_{m,q} \qquad q = x, y \text{ and } z \tag{3.28}$$



(a) Velocity field including the values of external nodes.

(b) Effective velocity field.

Figure 3.29: Representation of a velocity field with the proposed strategies.

The system matrix $\mathbf{M}$ is obtained by the assembly of the mass matrix-type array of each element along the boundary. The global mass matrix is given by:

$$\mathbf{M} = \sum^{n_e} \int_{\Gamma_D^e} \mathbf{N}^T \mathbf{N} |\mathbf{J}| \mathrm{d}\Gamma \tag{3.29}$$

where

Γ_D^e is the portion of the boundary within the element,

$\mathbf{N}$ corresponds to the matrix of finite element interpolation functions.

On the other side of the equation, the vector $\mathbf{G}_{m,q}$ is evaluated by adding the contribution of elements:

$$\mathbf{G}_{m,q} = \sum^{n_e} \int_{\Gamma_D^e} \mathbf{N}^T \mathbf{V}_{m,q}^e |\mathbf{J}| \mathrm{d}\Gamma \qquad q = x, y, z \tag{3.30}$$

with $\mathbf{V}_{m,q}^e$ as the q^{th} component of the velocity field on the boundary related to the design variable m within each element. This is a low-cost procedure as it only involves the elements along the boundary, which is of interest for 3D domains.

3.4.1.2.2. Physical approach

This method consists of solving a linear elasticity problem in which the velocity field on the boundary is considered as the displacements applied on the boundary. This auxiliary problem will have, for example, the following characteristics:

- The body $\Omega_{\texttt{Phys}}$ is characterized with a linear elastic material with Young's modulus equal to one and zero Poisson's ratio;
- The discretization used to evaluate the design velocity field is the discretization used to evaluate the displacements;
- Every single shape design variable gives a non-zero velocity field on the elements cut by the boundary and zero velocity on the rest of the domain, which ensures the equilibrium of each auxiliary problem. The unknowns are the velocities for all external nodes of the actual FE mesh $\Omega_{\texttt{Approx}}$.

This method needs the resolution of a system of equations as large as the original problem, however the associated computational cost is reduced, as: a) the FE mesh used for the analysis is the same Cartesian mesh as that used for the sensitivity analysis, b) we can remove the internal nodes from the system since the velocity field is set to 0 on these nodes, leading to a problem only associated with the domain's boundary, which can be seen as a 2D problem, and c) the stiffness matrix $\mathbf{K}$ can be factorized during the process and used for all the design variables.

3.4.2. Calculating shape sensitivities with FEAVox

This section describes the adaptation of the discrete analytical method to evaluate shape sensitivities when using cgFEM. In order to do this we have to take into consideration that imposing Dirichlet boundary conditions in an immersed boundary environment requires different strategies from those used in standard FEM, as in the calculation of the shape sensitivities dealt with in this section.

The derivative of (3.7) with respect to any design variable a_m provides the sensitivity of the calculation

$$\left(\frac{\partial \mathbf{K}}{\partial a_m} + \frac{\partial \mathbf{K_D}}{\partial a_m}\right)\mathbf{u} + (\mathbf{K} + \mathbf{K}_D)\frac{\partial \mathbf{u}}{\partial a_m} = \frac{\partial \mathbf{f}_q}{\partial a_m} + \frac{\partial \mathbf{f}_g}{\partial a_m} + \frac{\partial \mathbf{f}_s}{\partial a_m} \tag{3.31}$$

then, rearranging, yields

$$(\mathbf{K} + \mathbf{K}_D)\frac{\partial \mathbf{u}}{\partial a_m} = \left(\frac{\partial \mathbf{f}_q}{\partial a_m} + \frac{\partial \mathbf{f}_g}{\partial a_m} + \frac{\partial \mathbf{f}_s}{\partial a_m}\right) - \frac{\partial \mathbf{K}}{\partial a_m}\mathbf{u} = \mathbf{f}_{ps_m} \tag{3.32}$$

The discrete analytical method consists of obtaining analytical expressions of the sensitivities of the external forces and stiffness matrix. In our case we used the finite differences approximation of Eqs. (3.19) and (3.27) in the evaluation of the velocity field that will be used to obtain the derivatives of the previous equation. The sensitivities of the displacements are obtained from (3.32) and from these sensitivities other response magnitudes are calculated.

3.4.2.1. Evaluation of derivatives

We derive the components of (3.32) to be able to evaluate the shape sensitivities in the Cartesian grid framework. First, starting with $\mathbf{k}^e$ and considering that the derivative of $\mathbf{D}$ with respect to design variables is zero

$$\frac{\partial \mathbf{k}^e}{\partial a_m} = \int_{\Omega^e} \left[\frac{\partial \mathbf{B}^T}{\partial a_m} \mathbf{D}\mathbf{B} + \mathbf{B}^T \mathbf{D} \frac{\partial \mathbf{B}}{\partial a_m} \right] |\mathbf{J}| \mathrm{d}\Omega + \int_{\Omega^e} \left[\mathbf{B}^T \mathbf{D}\mathbf{B} \frac{\partial |\mathbf{J}|}{\partial a_m} \right] \mathrm{d}\Omega \tag{3.33}$$

As shown in [125], this expression depends on known magnitudes and the factors $\frac{\partial \mathbf{B}}{\partial a_m}$ and $\frac{\partial |\mathbf{J}|}{\partial a_m}$, which are a function of the velocity field evaluated above.

To evaluate $\frac{\partial \mathbf{k}_D^e}{\partial a_m}$ it is necessary to take into account that, as the Cartesian grid will not be modified by the design variables, h and $\mathbf{C}$ do not depend on a_m. Therefore, the only non-zero partial derivative with respect to a_m is $\frac{\partial |\mathbf{J}|}{\partial a_m}$, which leads to:

$$\frac{\partial \mathbf{k}_D^e}{\partial a_m} = \int_{\Gamma_D^e} \frac{\kappa^*}{h} \mathbf{C}^T \mathbf{C} \frac{\partial |\mathbf{J}|}{\partial a_m} \mathrm{d}\Gamma \tag{3.34}$$

Considering constant acting forces, the derivatives of $\mathbf{f}$ are

$$\frac{\partial \mathbf{f}_g^e}{\partial a_m} = \int_{\Gamma_D^e} \frac{\kappa^*}{h} \mathbf{C}^T g \frac{\partial |\mathbf{J}|}{\partial a_m} \mathrm{d}\Gamma \tag{3.35}$$

where we have assumed that the Dirichlet boundary conditions are not a function of the design variables,

$$\frac{\partial \mathbf{f}_s^e}{\partial a_m} = \int_{\Gamma_D^e} \left[\mathbf{C}^T \frac{\partial \mathbf{T}(\hat{\mathbf{u}}^h)}{\partial a_m} |\mathbf{J}| + \mathbf{C}^T \mathbf{T}(\hat{\mathbf{u}}^h) \frac{\partial |\mathbf{J}|}{\partial a_m} \right] \mathrm{d}\Gamma \tag{3.36}$$

where

$$\frac{\partial \mathbf{T}(\hat{\mathbf{u}}^h)}{\partial a_m} = \frac{\partial \boldsymbol{\sigma}^*}{\partial a_m} \mathbf{n} + \boldsymbol{\sigma}^* \frac{\partial \mathbf{n}}{\partial a_m} \tag{3.37}$$

To evaluate the stresses in linear elasticity we consider the general expression for the calculation of the FE stresses $\boldsymbol{\sigma}_h$ in continuous isoparametric elements

$$\boldsymbol{\sigma}_h = \mathbf{D}\mathbf{B}\mathbf{u}_h^e \tag{3.38}$$

$\mathbf{u}_h^e$ being the vector of nodal displacements of element e. Taking the derivative with respect to the design variable a_m yields

$$\frac{\partial \boldsymbol{\sigma}}{\partial a_m} = \mathbf{D}\mathbf{B} \frac{\partial \mathbf{u}_h^e}{\partial a_m} + \mathbf{D} \frac{\partial \mathbf{B}}{\partial a_m} \mathbf{u}_h^e \tag{3.39}$$

where all terms on the right can be evaluated. Once we have evaluated both $\boldsymbol{\sigma}$ and $\frac{\partial \boldsymbol{\sigma}}{\partial a_m}$ we can apply the construction of the smoothing field based on a recovery technique shown in [104].

To simplify the evaluation of $\frac{\partial \boldsymbol{\sigma}^*}{\partial a_m}$ we considered $\frac{\partial \boldsymbol{\sigma}^*}{\partial a_m} = \left(\frac{\partial \boldsymbol{\sigma}}{\partial a_m}\right)^*$. The numerical results will show that this approximation, previously used in [126], does not influence the results.

In Paper D - Section 4 the reader can find the convergence analyses of this methodology using the proposed velocity fields.

3.4.3. Optimization using Cartesian grids

Structural shape optimization processes will benefit not only from the computational efficiency and accuracy of cgFEM but also from its data structure, which will allow information to be shared between the different geometries analyzed during the process, further improving the optimization process. An adapted mesh can be generated for each design without the need to perform a full adaptive remeshing procedure. This is based on using the sensitivity analysis of all magnitudes related with adaptive remeshing (location of nodes, error estimation, etc.) with respect to the design variables, and needs to be done only once on a geometry of reference and then the results can be projected onto other designs for analysis. This procedure is useful for moderate shape modifications during the whole optimization process, although the sensitivity analysis can be repeated if required. The projected information allows to generate an appropriate adapted mesh for each new design in one shot, with a much lower computational cost than the traditional adaptive remeshing operation over each design. This method was inspired by a similar strategy that was developed and used in the context of gradient-based[127] and evolutionary[128] optimization methods based on the standard, body-fitted, Finite Element Method.

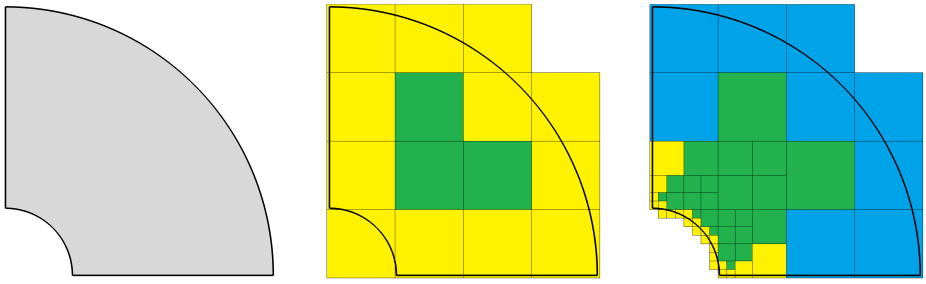
3.4.3.1. Data sharing

It should be noted that the hierarchical relationships considered in the data structure involves the automatic improvement of mesh refinement, thus positively affecting the efficiency of the FE implementation. Nodal coordinates, mesh topology, hierarchical relations, neighborhood patterns, and other geometric information are algorithmically evaluated when required.

As shown in [1, 42], cgFEM's hierarchical data structure provides the ability to re-use multiple calculations and considerably improve computational efficiency. We adapted the data sharing procedures presented in these references to the 3D case as summarized below.

The basic example of re-use of calculations is in linear elasticity problems with homogeneous materials, where all internal elements share the same stiffness matrix, which is only calculated once on a reference element. A scale factor related to the mesh level is then used to adapt the stiffness to the actual element size. Figure 3.30a shows a cross section of a model of a quarter of a cylinder, Figure 3.30b shows a coarse analysis mesh and Figure 3.30c shows the mesh obtained after its h-adaptive refinement. For both meshes we only have to evaluate one element for the domain colored in green, which represents all the internal elements.

H-adaptive analysis benefits from the hierarchical data structure by means of the so-called *vertical data sharing*. This technique consists of using the hierarchical data structure to keep the element matrices evaluated for each boundary element so that this information is already available if they appear again in other meshes of the h-adaptive analysis. Figure 3.30c represents the resulting mesh of an h-adaptive process where the blue colored elements have been evaluated in previous meshes. The only element matrices that need to be evaluated for the analysis of the mesh shown in this figure are the yellow elements.



(a) Model of a quarter of a cylinder.

(b) Coarse mesh i.

(c) h-adapted mesh $i+1$.

Figure 3.30: *Vertical data sharing* example.

The body-fitted meshes of the traditional FEM hinder the efficient exchange of information between different geometries. However, cgFEM's hierarchical data structure provides a framework that can easily transfer information between different geometries through the so-called *horizontal data sharing*, which only requires all the geometries to be defined within the same embedding domain to ensure that the Cartesian grid pile is the same for all geometries, making the inter-geometries data transfer possible.

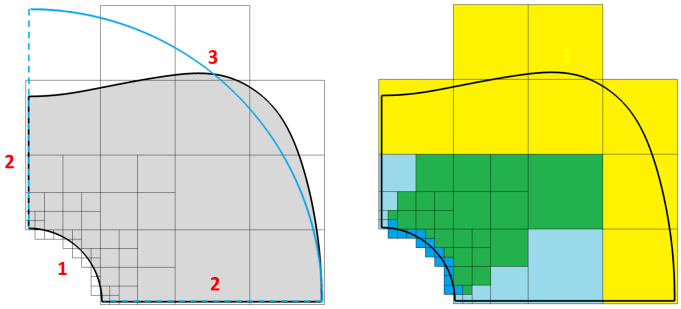
The boundary of the domain can be subdivided into three types:

Type 1. Fixed part: Part of the boundary that remains fixed in all the geometries (such as the internal curve of the cylinder represented in Figure 3.31a). The

matrices of the elements trimmed by a Type 1 boundary will be shared between different geometries (see dark blue elements in Figure 3.31).

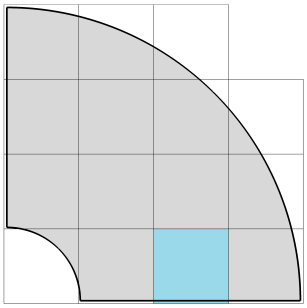
Type 2. Moving part with fixed intersection pattern: These surfaces or curves can be changed by the optimization algorithm, but this does not affect the intersection with the elements. In Figure 3.31a, we can see two of the models's Type 2 planes of symmetry curves. The matrices of the elements fully trimmed by a Type 2 boundary will also be shared between different geometries (see light blue elements in Figure 3.31).

Type 3. Free moving part: This part of the geometry can freely change during the optimization analysis with no predictable pattern, e.g. the outer curve in Figure 3.31a.

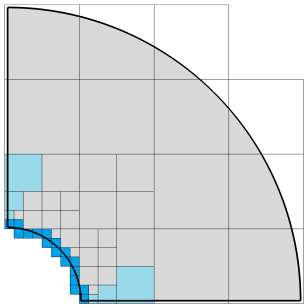


(a) Different types of entities.

(b) Individual $j+1$.



(c) Individual j, mesh i.



(d) Individual j, mesh $i+1$.

Figure 3.31: *Horizontal data sharing* example.

For the geometry shown in Figures 3.31a and 3.31b, the matrices for the green elements had already been evaluated in a reference element and *vertical data shar-*

ing reduced the calculations through the h-adaptive refinement of that particular geometry. Finally, after considering the *vertical data sharing*, the number of (yellow) elements that need integration are considerably fewer than the elements in the mesh.

3.4.3.2. Nested domain reordering

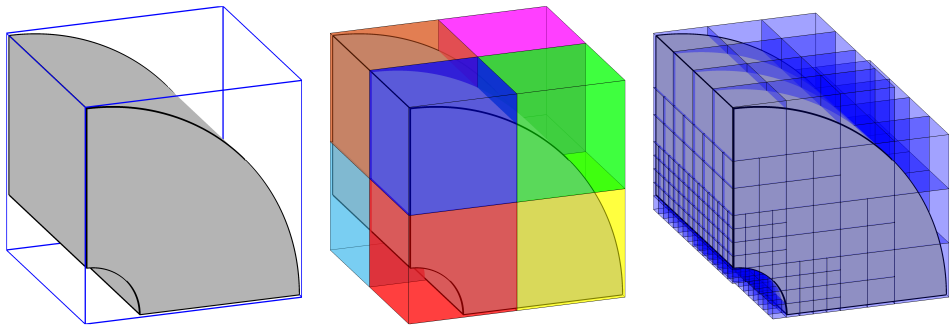
Solving large sparse linear systems is the most time-consuming computation in shape optimization using FEM.

Matrix reordering plays an important role in the performance of direct solvers. In fact, it is common to reorder the system matrix before proceeding to its factorization as it can increase the sparsity of the factorization, making the overall process faster and reducing the storage cost. Finding the optimal ordering is usually not possible although heuristic methods can be used to obtain good reorderings at a reasonable computational cost.

This section aims to show how the hierarchical data structure inherent to Cartesian grids, thus directly related to mesh topology, can be used to directly obtain a reordering of the system matrix that speeds up the Cholesky factorization process. The Nested Domain Decomposition (NDD) technique is a technique specially tailored to h-adaptive FE analysis codes with refinement based on element subdivision. The technique simply consists of recursively subdividing the problem domain using the hierarchical structure of the mesh. This technique was first described in [88] and applied in an implementation of a FEM that used geometry-conforming meshes and later adapted to a Cartesian grid environment in 2D[42]. In this Thesis we use an NDD 3D generalization. The technique consist of subdividing the problem domain considering each element of a uniform grid of the lowest levels of the Cartesian grid pile (normally the Level-1 grid, with 2x2x2 elements). The degrees of freedom of the nodes of the mesh to be analyzed falling into a subdomain will then be allocated together in the stiffness matrix. The nodes falling on the interface of the subdomains will not be reordered and will simply be moved to the end of the matrix, producing the typical arrowhead-type structure the domain decomposition techniques. This idea is then recursively applied to each original subdomain, producing a nested arrowhead-type structure. This reordering considerably reduces the computational cost associated with the resolution of the system of equations.

Figures 3.32, 3.33 and 3.34 graphically show the process. The embedding domain (Figure 3.32a) is subdivided into 8 subdomains or regions as shown in Figure 3.32b. Each subdomain is in a different color. We can easily identify the subdomains with the elements of the first refinement level, so that nested reordering in cgFEM is done by grouping the nodes according to the corresponding element in the hierarchical structure. Figure 3.32c shows an example of an analysis mesh ready for nested reordering.

For the sake of clarity we use a 2D representation of the process. Figure 3.33a shows the domain subdivision with a Level-1 grid and the nodes subdivided into 9



(a) Body within the embedding domain $\Omega_{\text{Phys}} \subset \Omega$.

(b) Level 1 subdivision.

(c) Example of 3D mesh to be reordered.

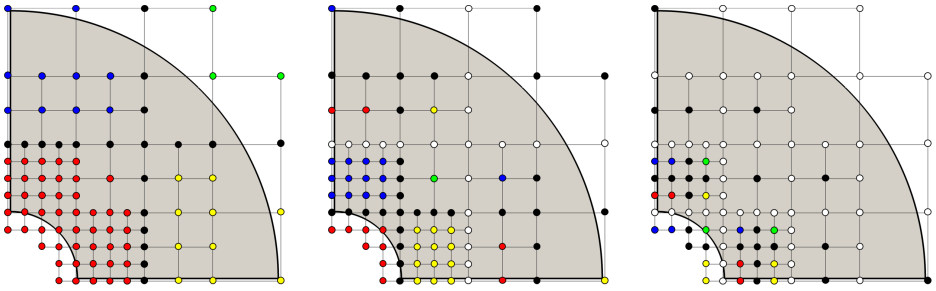
Figure 3.32: Nested Domain Decomposition environment.

different categories, 5 of which are shown in the 2D representation. The colored categories indicate the nodes falling into each of the elements in the Level-1 grid and the black fall on the interface between the Level-1 elements. The stiffness matrix will be reordered, grouping all the nodes by color, as shown in Figure 3.34b, into an arrowhead-type structure made up of blocks. The blocks on the diagonal (two shadowed in blue and red) have a structure similar to that of the original non-reordered stiffness matrix in Figure 3.34a.

Level-2 reordering (Figure 3.33b) indicates that each of the Level-1 subdomains is again reordered in the same way. For instance, the red subdomain in Figure 3.33a is subdivided into 8 subdomains (only 4 are shown in 2D) separated by their interface, represented in black, as shown in Figure 3.33b.The interfaces of previous levels are represented by white nodes. The same process is followed for the next levels, using the elements of the corresponding level of the hierarchical structure.

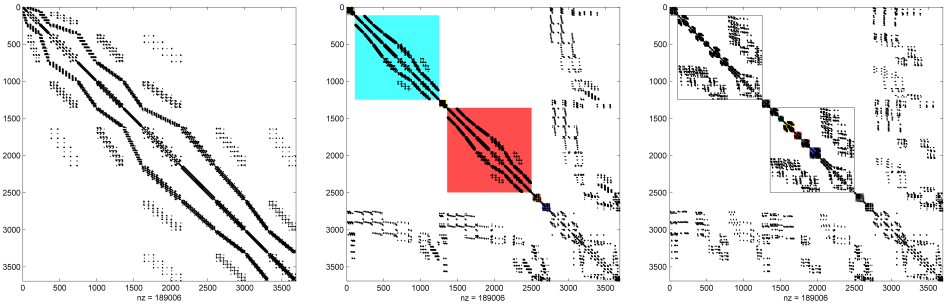
In the process, each node of the mesh is given a code with as many digits as levels of the Cartesian grid pile used. The i-th digit of the code contains the subdomain number (1 to 8) of the node considering the Level-i grid, or 9 if the node is on the interface between the Level-i subdomains, as in Figures 3.33a to 3.33c for levels 1 to 3. Once the code of each node has been obtained a simple 'alphabetical' reordering of the codes provides the NDD reordering of the nodes. The degrees of freedom of the matrix will then be reordered considering nodal reordering.

The result of the NDD reordering generates the nested arrowhead type structure of the stiffness matrix represented in Figure 3.34c, which could also be used to define efficient nested domain decomposition solvers or iterative solvers, as in [129] which describes their initial implementations. However, we use this technique here to reorder the system of equations to improve the performance of the Cholesky factorization.



(a) Level-1 decomposition. (b) Level-2 decomposition. (c) Level-3 decomposition.

Figure 3.33: Nested Domain Decomposition scheme.



(a) Original stiffness matrix. (b) Level 1 decomposition reordering. (c) Last reordering (Level 4).

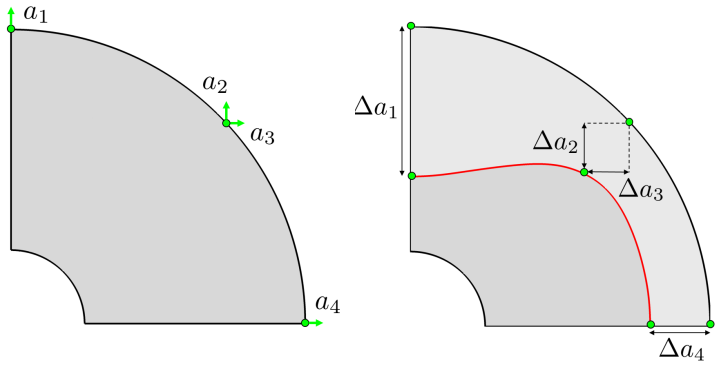
Figure 3.34: Nested Domain Decomposition output.

3.4.3.3. Automatic *h*-adaptive mesh projection

We use a gradient-based algorithm [130] which uses first-order sensitivities of the objective functions and constraints to evaluate the solution of (2.15). Using this information and the values of the design variables for the j-th geometry obtained during the iterative process ($\mathbf{a}^j$), see Figure 3.35a, the algorithm generates the modified values of $\mathbf{a}^j$ defining an improved design ($\mathbf{a}^{j+1}$) using

$$\mathbf{a}^{j+1} = \mathbf{a}^j + \alpha \mathbf{S}(\mathbf{a})^{j+1} \tag{3.40}$$

where $\mathbf{S}(\mathbf{a})^j$ is the search direction vector and α is a parameter related to the step size.



(a) Reference design (j). (b) Perturbed design ($j+1$).

Figure 3.35: Design evolution during optimization.

After the definition of the $(j+1)$-th geometry to be analyzed (see Figure 3.35b) it is necessary to construct the new analysis mesh. Previous developments using standard body-fitted FE meshes [127, 128], in which the information required to define a new mesh was projected from one geometry to another, made use of the following expression:

$$M_{j+1} \approx M_j + \sum \left(\frac{\partial M_j}{\partial a_m} \right) \cdot \Delta a_m \tag{3.41}$$

where M represents any magnitude that has to be projected from geometry j to geometry $j+1$. The h-adapted mesh used in these references generated from a mesh optimality criterion that minimized the number of elements in the mesh to be created that would produce the prescribed estimated error in energy norm. In the following, we use our 3D generalization of this criterion (see Section 3.3.2).

Let us assume that $\Omega_{j,def}$ is mesh n of an h-adaptive analysis that corresponds to the geometry $j+1$ and we want to evaluate mesh $n+1$ (the new mesh) of the h-adaptive sequence, then the size h of each element of the mesh $n+1$ is given by Equation (3.15), reproduced here for convenience:

$$h_{e,n}^{n+1} \approx h_e^n \left[\frac{1}{M^n} \right]^{1/2(p+1)} \left[\frac{\|\mathbf{e}^{n+1}\|}{\|\mathbf{e}^n\|} \right]^{\frac{d}{2p^2+pd}} \left[\frac{\|\mathbf{e}^{n+1}\|}{\|\mathbf{e}^n\|_e} \right]^{\frac{2}{2p+d}}$$

To use this expression we have to replace $\|\mathbf{e}^n\|_e$ by a projection from a previous geometry j, evaluate $\|\mathbf{e}\|_n$ as the summation of all the projected errors in elements and evaluate $\|\mathbf{e}^{n+1}\|$ as

$$\|\mathbf{e}^{n+1}\| = \frac{\gamma}{100} \|\mathbf{u}_{es}^{j+1}\| \tag{3.42}$$

where γ is the prescribed percentage of relative error in energy norm and $\|\mathbf{u}_{es}^{j+1}\|$ is the global projected energy norm.

Once a new design has been defined, the projection thus starts with the previous analysis mesh, defined as $\Omega_{j,\square}$ in Figure 3.36a, using the previously computed coordinate sensitivities. The projected position $\mathbf{r}^{j+1}$ for each node of the mesh is given by:

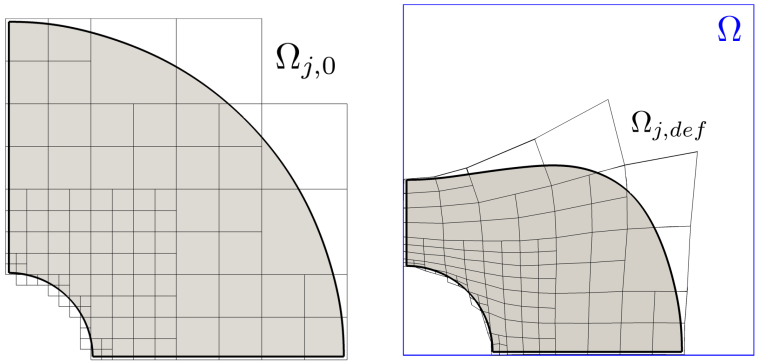
$$\mathbf{r}^{j+1} = \mathbf{r}^{j} + \sum_{i}^{m} \left(a_i^{j+1} - a_i^{j}\right) \left(\frac{\partial \mathbf{r}^{j}}{\partial a_i^{j}}\right) \tag{3.43}$$

Likewise, the estimated error in energy norm and the estimated energy norm at each element required in (3.15) can also be estimated by projection using the expressions

$$\|\mathbf{e}_{es}\|_{e,j+1}^2 \approx \|\mathbf{e}_{es}\|_{e,j}^2 + \sum_{i}^{m} \left(a_i^{j+1} - a_i^{j}\right) \frac{\partial \|\mathbf{e}_{es}\|_{e}^2}{\partial a_i} \tag{3.44}$$

$$\|\mathbf{u}_{es}\|_{e,j+1}^2 \approx \|\mathbf{u}_{es}\|_{e,j}^2 + \sum_{i}^{m} \left(a_i^{j+1} - a_i^{j}\right) \frac{\partial \|\mathbf{u}_{es}\|_{e}^2}{\partial a_i} \tag{3.45}$$

These projections give an approximation of the values of the estimated error in energy norm and the energy norm that would be obtained if the next design were computed with the previous Cartesian mesh $\Omega_{j,\square}$ projected onto the new geometry, represented as $\Omega_{j+1,def}$ in Figure 3.36b.



(a) Cartesian reference analysis mesh, $\Omega_{j,\square}$.

(b) Projected (non-Cartesian) mesh on geometry $j+1$.

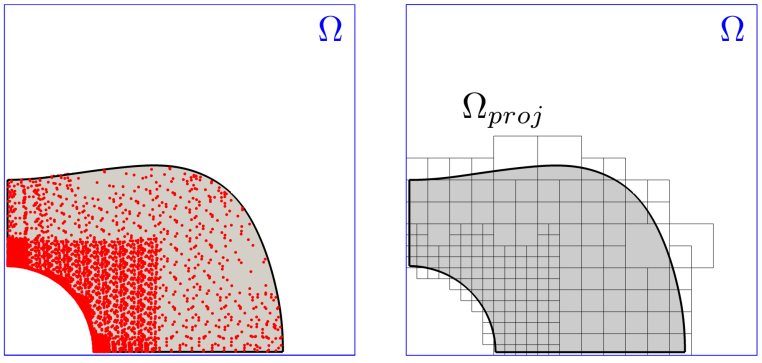
Figure 3.36: Mesh projection procedure.

Without any further computation on geometry $j+1$, the projected estimated error and energy norm allow us to estimate the quality of the results that would be obtained through the FE analysis of geometry $j+1$ with a mesh (Figure 3.36b) equivalent to the one used in the previous design j (Figure 3.36a). If the target error prescribed for the FE analysis is lower than the projected error of the $(j+1)$-th geometry, the mesh must be h-refined using (3.15).

Up to this point, the mesh projection described is comparable to the strategies used for standard body-fitted meshes [127, 128]. As we can easily observe in Figure 3.36b, this kind of projection yields a discretization that is not compatible with cgFEM's hierarchical Cartesian structure, thus wasting most of its advantages.

We therefore propose a projection strategy to generate an h-adapted analysis mesh of the new design $j+1$, keeping the Cartesian structure intact.

This strategy simply requires projecting the element size evaluated using (3.15) for the elements of $\Omega_{j+1,def}$ (Figure 3.36b) onto the embedding domain Ω. To do this we assign this element size to the Gauss points of each element and project all the integration points of $\Omega_{j+1,def}$ to Ω. These projected integration points containing element size information can be trivially located in the elements of a uniform Cartesian grid of the prescribed level. These Cartesian elements are then recursively refined until the size of each element is smaller than the minimum element sizes defined by the Gauss points contained in the element, leading to an h-adapted Cartesian grid (see 3.37b)



(a) Perturbed integration points.

(b) Projected Cartesian mesh, $\Omega_{j+1,\square}$

Figure 3.37: Mesh projection procedure.

From this projection perspective, sensitivity analysis can transform a posteriori error estimation into a preprocess tool able to generate an h-adapted mesh for the new design, recycling the calculations obtained in previous stages of the optimization process.

Illustrative example. *The objective of this problem is to minimize the volume of a connecting rod without violating the given maximum Von Mises stress. Because of the symmetry, only a fourth of the component is modeled. The geometry of the initial design and the boundary conditions are shown in Figure 3.38. The geometry parameters are* $AB = 11$, $C = 4$, $AD = 20$, $DE = 4$, $F = 1.5$, $DG = 7$, $HG = 5.5$. *The Young's modulus is* $E = 10^5$, *and Poisson's ratio* $\nu = 0.333$. *The pressure is* $P = 100$ *in the normal direction of the half arc as shown in Figure 3.38.*

The design boundary is the surface HG. *The end point* H *is fixed while eight points are used to interpolate* HG. *The vertical positions of the eight interpolation points on the design surface are set as design variables (see Figure 3.39). The allowable von Mises stress is* $\sigma_{VM} = 900$.

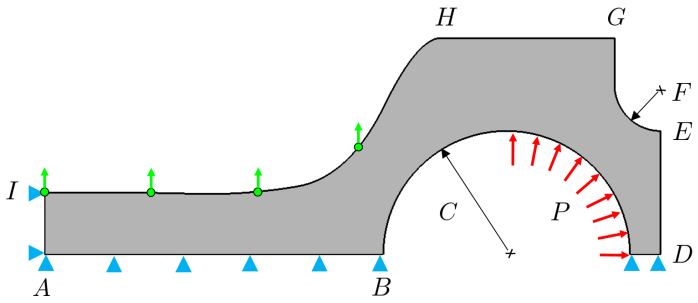


Figure 3.38: Front view of the connecting rod problem with boundary conditions.

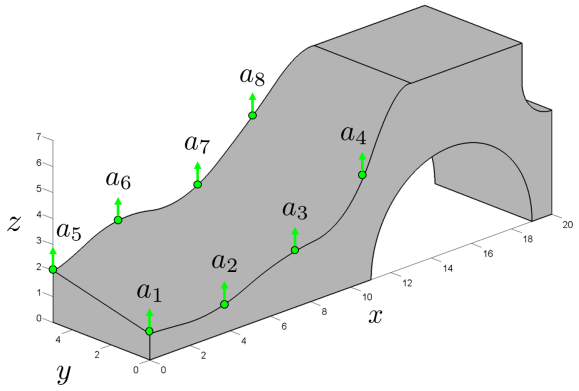


Figure 3.39: 3D model representation showing the 8 design variables.

The initial values of the design variables and their allowed data range are shown in Table 3.2.

Design variable	Initial value	Data range
a_1, a_5	7	$[1-7]$
a_2, a_6	7	$[1-7]$
a_3, a_7	7	$[1.2-7]$
a_4, a_8	7	$[2-7]$

Table 3.2: Connecting rod defined by 8 design variable. Design variables data.

Table 3.3 shows the average discretization estimated error in energy norm per individual and the computational cost per individual. We observe for this problem how the optimization procedure based in the mesh projection presented above saves slightly more of the 20% of the time per individual.

Type of mesh	Computational cost (s)	Estimated discretization error
hAdapMeshing	607.84	2.83%
ProjMeshing	471.02	2.62%

Table 3.3: Connecting rod defined by 8 design variable. Computational results for h-adapted and projected meshes.

Figure 3.40 shows the Von Mises stress fields for the initial configuration of the model opposed to the field obtained for the optimal solution provided by the shape optimization algorithm.

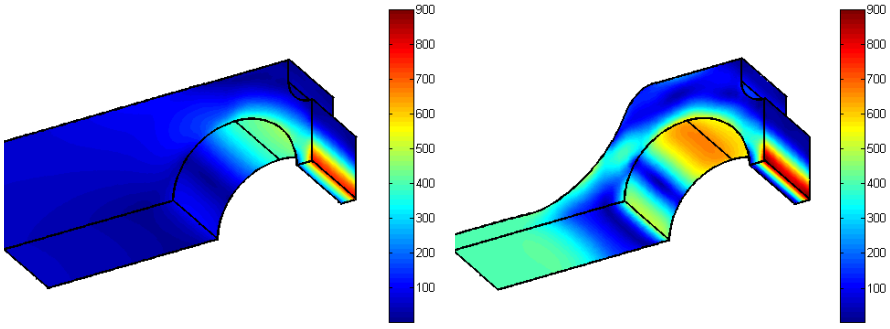


Figure 3.40: Von Mises stress fields: (left) initial configuration results and (right) configuration obtained using projected meshes.

Chapter 4

Closure

The main contributions of the Thesis and the open lines of research are summarized in the following subsections.

4.1. Summary

In this Thesis we introduce a cgFEM-based 3D methodology for solving linear elasticity and optimization problems and demonstrate its performance in a variety of numerical examples. The key points of the improved cgFEM operations can be summarized as follows:

- The use of Cartesian grids, which make the mesh independent of the geometry and embed the problem domain in a simpler domain to facilitate the process of creating FE meshes. This independence requires a robust intersection process that defines the relative position of the elements with respect to the geometry. The intersected elements also require the creation of integration subdomains to capture the region of the cut elements internal to the physical domain.
- Novel NURBS-Enhanced integration techniques to remove the geometric errors associated with standard Immersed Boundary Methods due to inexact approximations of the embedded geometry. Even though our methodology can consider piece-wise polynomial approximations, i.e. linear or quadratic facets, the

proposed integration scheme is based on the latest CAD technologies, namely NURBS or T-spline.

- Since IBM mesh nodes do not lie on the boundary of the physical domain it is not possible to strongly impose essential boundary conditions, so the technique adopted here consists of using stabilized Lagrange multipliers to impose Dirichlet boundary conditions. Our approach takes advantage of recovered stress fields to improve the quality of the solution along the boundary and to define the stabilization term.
- Two strategies are proposed to generate h-adapted meshes: the first is in the pre-analysis step and allows meshes to be geometrically adapted. The second strategy allows the refinement to be driven by the level of error present in the FE solution.
- A shape sensitivity calculation module has been adapted to the non-conforming Cartesian grids. Due to the immersed boundary features, standard FEM sensitivity calculations need to be adapted to provide the parameters necessary for gradient-based optimization algorithms.
- The hierarchical data structure, easily built on the Cartesian grid mesh structure, means easy-to-implement instruments can be used to save computational cost during shape optimization procedures. In addition, the solver benefits from the Cartesian structure thanks to Nested Domain Decomposition reordering.

4.2. Open research lines

This work has opened up several lines of research on cgFEM technology as follows:

- **Improvement of integration along the boundary**

  Despite the fact that the special treatment of intersected elements during integration is limited to a small proportion of the mesh, creating subdomains in elements in which several surfaces intersect arbitrarily could be further improved. In the present implementation we use pre-calculated patterns only in the elements intersected by one surface when the other elements need specific tetrahedralization, although a few topological situations may appear. One possible solution would be to create patterns for the most frequent intersection situations. Another possibility is to eliminate the tetrahedrons as integration subdomains and substitute them by specifically designed integration quadratures for each of the intersection patterns.

- **To enrich the FE basis functions**

  It would be interesting to represent weak discontinuities, such as internal material interfaces or even strong discontinuities such as cracks. This task could be addressed by implementing XFEM in the cgFEM code, which would facilitate dealing with crack propagation and provide FEAVox with interesting capabilities.

- **Implementation of high-order basis functions for nodal interpolation**.

  Many physical problems need a high-order polynomial interpolation to properly capture the numerical solution. In this regard, there are several options, including the degree elevation of standard FE, but since we are using structured grids, it would be more reliable to use the Cartesian mesh as a control net for high-order B-spline embedding volume.

- **Fully implementation of recovery-based techniques including goal-oriented error estimation**

  In this work error estimators are a fundamental part in the development of the contributions. Several extensions for recovery-based error estimators are described in [1], including error estimation in quantities of interest. These strategies have been shown to be a reliable option in estimating errors in 2D cgFEM. A natural solution would be to extrapolate the knowledge acquired in the 2D developments to the 3D implementation described here.

- **Stabilization methods**

  Work is in progress on improving the ill-conditioning of the system of equations when using non-conforming meshes. The main reason for this behavior is the fact that the energy contribution of external nodes can be very small and the global energy of the problem is hardly affected by their solution. In addition to the Dirichlet formulation terms, additional terms could be designed to stabilize the solution along the Neumann boundaries. These improvements in the condition of the matrices could also affect the behavior of the iterative solvers used for large problems and at the same time preserve the accuracy of the results.

- **Developing new strategies for shape optimization**

  Although the developments described here are oriented towards gradient-based optimization algorithms, gradientless algorithms could also be used. However, the main drawback of gradientless optimization is its slow convergence due to the lack of a proper search direction. Most of the strategies described here could be applied to such cases and would reduce the computational cost, e.g. the sensitivity analysis for mesh projection. Also, recent techniques based on the creation of surrogate models, with the appropriate training, can be used to obtain good approximations to FE results at a fraction of the cost.

- **Testing the methodology with different problems and formulations**

  Contact problems or Finite Element simulations of medical images are some of the current Cartesian grid study areas in the Department of Mechanical and Materials Engineering. These problems are the subject of a project on the design of patient-specific implants by means of FEM models obtained from 3D images. This Thesis provides the basis to include contact algorithms to properly model movements between the bone and the implant and between the different implant parts, and the shape optimization tools described here will allow to obtain optimal implants, specially tailored to the patients' needs extracted from their medical data.

Bibliography

[1] Nadal E. *Cartesian Grid FEM (cgFEM): High Performance h-adaptive FE Analysis with Efficient Error Control. Application to Structural Shape Optimization. PhD Thesis*. Universitat Politècnica de València, 2014. 1.1, 3.4.3.1, 4.2

[2] Bendsoe MP, Sigmund O. *Topology Optimization: Theory, Methods, and Applications*. Springer Verlag, Berlin, 2003. i

[3] Allaire G, Jouve F, Toader AM. Structural optimization using sensitivity analysis and a level-set method. *Journal of Computational Physics* 2004; **194**(1):363–393. i

[4] Braibant V, Fleury C. Shape optimal design using b-splines. *Computer Methods in Applied Mechanics and Engineering* 1984; **44**(3):247–267. 1.2.1

[5] Haftka RT, Grandhi RV. Structural shape optimization: A survey. *Computer Methods in Applied Mechanics and Engineering* 1986; **57**(1):91–106. 1.2.1

[6] Ródenas JJ, Bugeda G, Albelda J, Oñate E. On the need for the use of error-controlled finite element analyses in structural shape optimization processes. *International Journal for Numerical Methods in Engineering* 2011; **87**(11):1105–1126. 1.2.1

[7] Bennett JA, Botkin ME. Structural Shape Optimization with Geometric Problem Description and Adaptive Mesh Refinement. *AIAA Journal* 1985; **23**(3):459–464. 1.2.1, 3.4.1.2

[8] Chang K, Choi KK. A geometry-based parameterization method for shape design of elastic solids. *Mechanics of Structures and Machines* 1992; **20**(2):215–252. 1.2.1

[9] Kikuchi N, Chung KY, Torigaki T, Taylor JE. Adaptive finite element methods for shape optimization of linearly elastic structures. *Computer Methods in Applied Mechanics and Engineering* 1986; **57**(1):67–89. 1.2.1

[10] Yao T, Choi KK. 3-d shape optimal design and automatic finite element regridding. *International Journal for Numerical Methods in Engineering* 1989; **28**(2):369–384. 1.2.1

[11] Riehl S, Steinman P. An integrated approach to shape optimization and mesh adaptivity based on material residual forces. *Computer Methods in Applied Mechanics and Engineering* 2014; **278**:640–663. 1.2.1

[12] Chiandussi G, Bugeda G, Oñate E. A simple method for automatic update of finite element meshes. *Communications in Numerical Methods in Engineering* 2000; **16**(1):1–19. 1.2.1

[13] Zienkiewicz OC, Zhu JZ. A Simple Error Estimator and Adaptive Procedure for Practical Engineering Analysis. *International Journal for Numerical Methods in Engineering* 1987; **24**(2):337–357. 1.2.1, 3.3.2, 3.3.2

[14] González-Estrada OA, Nadal E, Ródenas JJ, Kerfriden P, Bordas SPA, Fuenmayor FJ. Mesh adaptivity driven by goal-oriented locally equilibrated superconvergent patch recovery. *Computational Mechanics* 2014; **53**(5):957–976. 1.2.1

[15] Hughes TJR, Cottrell JA, Bazilevs Y. Isogeometric Analysis: CAD, Finite Elements, NURBS, Exact Geometry, and Mesh Refinement. *Computer Methods in Applied Mechanics and Engineering* 2005; **194**:4135–4195. 1.2.1

[16] Nguyen VP, Anitescu C, Bordas SPA, Rabczuk T. Isogeometric analysis: An overview and computer implementation aspects. *Mathematics and Computers in Simulation* 2015; **117**:89–116. 1.2.1

[17] Zhang Y, Wang W, Hughes TJR. Conformal Solid T-spline Construction from Boundary T-spline Representations. *Computational Mechanics* 2013; **6**(51):1051–1059. 1.2.1

[18] Escobar JM, Montenegro R, Rodríguez E, Cascón JM. The meccano method for isogeometric solid modeling and applications. *Engineering with Computers* 2014; **30**(3):331–343. 1.2.1

[19] Liu L, Zhang Y, Hughes TJR, Scott MA, Sederberg TW. Volumetric T-spline Construction using Boolean Operations. *Engineering with Computers* 2014; **30**(4):425–439. 1.2.1

[20] Cho S, Ha SH. Isogeometric shape design optimization: exact geometry and enhanced sensitivity. *Structural and Multidisciplinary Optimization* 2009; **38**(1):53–70. 1.2.1

[21] Ha SH, Choi K, Cho S. Numerical method for shape optimization using T-spline based isogeometric method. *Structural and Multidisciplinary Optimization* 2010; **42**(3):417–428. 1.2.1

[22] Qian X. Full analytical sensitivities in NURBS based isogeometric shape optimization. *Computer Methods in Applied Mechanics and Engineering* 2010; **199**(29–32):2059–2071. 1.2.1

[23] Li K, Qian X. Isogeometric analysis and shape optimization via boundary integral. *Computer-Aided Design* 2011; **43**(11):1427–1437. 1.2.1

[24] Lian H, Kerfriden P, Bordas SPA. Implementation of regularized isogeometric boundary element methods for gradient-based shape optimization in two-dimensional linear elasticity. *International Journal for Numerical Methods in Engineering* 2016; **106**(12):972–1017. 1.2.1

[25] Peskin CS. Numerical Analysis of Blood Flow in the Heart. *Journal of Computational Physics* 1977; **25**:220–252. 1.2.2

[26] Zhang L, Gerstenberger A, Wang X, Liu WK. Immersed Finite Element Method. *Computer Methods in Applied Mechanics and Engineering* 2004; **293**(21):2051–2067. 1.2.2

[27] Haslinger J, Jedelsky D. Genetic algorithms and fictitious domain based approaches in shape optimization. *Struc. Optim.* 1996; **12**:257–264. 1.2.2

[28] Kunisch K, Peichl G. Numerical gradients for shape optimization based on embedding domain techniques. *Comput. Optim.* 1996; **18**:95–114. 1.2.2

[29] Kim NH, Chang Y. Eulerian shape design sensitivity analysis and optimization with a fixed grid. *Computer Methods in Applied Mechanics and Engineering* 2005; **194**(30–33):3291–3314. 1.2.2

[30] Najafi AR, Safdari M, Tortorelli DA, Geubelle PH. A gradient-based shape optimization scheme using an interface-enriched generalized FEM. *Computer Methods in Applied Mechanics and Engineering* 2015; **296**:1–17. 1.2.2

[31] Riehl S, Steinmann P. On structural shape optimization using an embedding domain discretization technique. *International Journal for Numerical Methods in Engineering* 2016; . 1.2.2

[32] García-Ruíz MJ, Steven GP. Fixed grid finite elements in elasticity problems. *Engineering Computations* 1999; **16**(2):145–164. 1.2.2

[33] Dunning PD, Kim HA, Mullineux G. Investigation and improvement of sensitivity computation using the area-fraction weighted fixed grid FEM and structural optimization. *Finite Elements in Analysis and Design* 2011; **47**(8):933–941. 1.2.2

[34] Parvizian J, Düster A, Rank E. Finite Cell Method: h- and p- Extension for Embedded Domain Methods in Solid Mechanics. *Computational Mechanics* 2007; **41**(1):121–133. 1.2.2

[35] Düster A, Parvizian J, Yang Z, Rank E. The finite cell method for three-dimensional problems of solid mechanics. *Computer Methods in Applied Mechanics and Engineering* 2008; **197**(45-48):3768–3782. 1.2.2

[36] Schillinger D, Ruess M. 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* 2015; **22**(3):391–455. 1.2.2

[37] Meagher D. Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer. *Technical Report IPL-TR-80-11 I*, Rensselaer Polytechnic Institute 1980. 1.2.2

[38] Jackins CL, Tanimoto SL. Oct-tree and their use in representing three-dimensional objects. *Computer Graphics and Image Processing* 1980; **14**(3):249–270. 1.2.2

[39] Doctor LJ, Torborg JG. Display techniques for octree-encoded objects. *IEEE Comput. Graph. Appl.* 1981; **1**(3):29–38. 1.2.2

[40] Kudela L, Zander N, Kollmannsberger S, Rank E. Smart octrees: Accurately integrating discontinuous functions in 3d. *Computer Methods in Applied Mechanics and Engineering* 2016; **306**(1):406–426. 1.2.2

[41] Fries TP, Omerović S. Higher-order accurate integration of implicit geometries. *International Journal for Numerical Methods in Engineering* 2016; **106**(5):323–371. 1.2.2

[42] Nadal E, Ródenas JJ, Albelda J, Tur M, Tarancón JE, Fuenmayor FJ. Efficient Finite Element Methodology based on Cartesian Grids: Application to Structural Shape Optimization. *Abstract and Applied Analysis* 2013; **2013**. 1.2.2, 3.3.2, 3.4.3.1, 3.4.3.2

[43] Marco O, Sevilla R, Zhang Y, Ródenas JJ, Tur M. Exact 3D boundary representation in finite element analysis based on Cartesian grids independent of the geometry. *International Journal for Numerical Methods in Engineering* 2015; **103**:445–468. 1.2.2, 3.3.1

[44] Brezzi F, Fortin M. *Mixed and hybrid finite element methods*. Springer-Verlag New York, Inc., 1991. 1.2.3, 1

[45] Béchet E, Moës N, Wohlmuth B. A stable lagrange multiplier space for stiff interface conditions within the extended finite element method. *International Journal for Numerical Methods in Engineering* 2009; **78**(8):931–954. 1.2.3

[46] Hautefeuille M, Annavarapu C, Dolbow JE. Robust imposition of dirichlet boundary conditions on embedded surfaces. *International Journal for Numerical Methods in Engineering* 2012; **90**(1):40–64. 1.2.3, 3.2

[47] Barbosa HJC, Hughes TJR. The finite element method with lagrange multipliers on the boundary: circumventing the babuska-brezzi condition. *Computer Methods in Applied Mechanics and Engineering* 1991; **85**(1):109–128. 1.2.3

[48] Barbosa HJC, Hughes TJR. Boundary lagrange multipliers in finite element methods: Error analysis in natural norms. *Numerische Mathematik* 1992; **62**(1):1–15. 1.2.3

[49] Hansbo A, Hansbo P. An unfitted finite element method based on nitsche method for elliptic interface problems. *Computer Methods in Applied Mechanics and Engineering* 2002; **191**(47–48):5537–5552. 1.2.3

[50] Hansbo P, Lovadina C, Perugia I, Sangalli G. A lagrange multiplier method for the finite element solution of elliptic interface problems using non-matching meshes. *Numerische Mathematik* 2005; **100**(1):91–115. 1.2.3

[51] Dolbow J, Harari I. An efficient finite element method for embedded interface problems. *International Journal for Numerical Methods in Engineering* 2009; **78**(2):229–252. 1.2.3, 3.2

[52] Sanders JD, Dolbow JE, Laursen TA. On methods for stabilizing constraints over enriched interfaces in elasticity. *International Journal for Numerical Methods in Engineering* 2009; **78**(9):1009–1036. 1.2.3, 3.2

[53] Haslinger J, Renard Y. A new fictitious domain approach inspired by the extended finite element method. *SIAM J. Numer. Anal.* 2009; **47**(2):1474–1499. 1.2.3, 3.2, 3.2

[54] Schott B, Wall W. A new face-oriented stabilized xfem approach for 2d and 3d incompressible navier-stokes equations. *Computer Methods in Applied Mechanics and Engineering* 2014; **276**:233–265. 1.2.3, 3.2, 3.2

[55] Burman E, Hansbo P. Fictitious domain finite element methods using cut elements: Ii. a stabilized nitsche method. *Applied Numerical Mathematics* 2012; **62**(4):328–341. 1.2.3, 3.2, 3.2

[56] Amdouni S, Hild P, Lleras V, Moakher M, Renard Y. A stabilized lagrange multiplier method for the enriched finite-element approximation of contact problems of cracked elastic bodies. *ESAIM: Mathematical Modelling and Numerical Analysis* 2012; **46**(4):813–839. 1.2.3

[57] Annavarapu C, Hautefeuille M, Dolbow JE. A robust nitche's formulation for interface problems. *Computer Methods in Applied Mechanics and Engineering* 2012; **225–228**:44–54. 1.2.3

[58] Sanders JD, Laursen TA, Dolbow JE. A nitsche embedded mesh method. *Computational Mechanics* 2012; **49**(2):243–257. 1.2.3, 3.2

[59] Annavarapu C, Hautefeuille M, Dolbow JE. A nitsche stabilized finite element method for frictional sliding on embedded interfaces. part i: single interface. *Computer Methods in Applied Mechanics and Engineering* 2014; **268**:417–436. 1.2.3, 3.2

[60] Yoon BG, Belegundu AD. Iterative methods for design sensitivity analysis. *AIAA Journal* 1988; **26**(11):1413–1415. 1.2.4

[61] Haftka RT. Semi-analytical static nonlinear structural sensitivity analysis. *AIAA Journal* 1993; **31**(7):1307–1312. 1.2.4

[62] Akgün MA, Garcelon GH, Haftka RT. Fast exact linear and nonlinear structural reanalysis and the sherman-morrison-woodbury formulas. *International Journal for Numerical Methods in Engineering* 2001; **50**(7):1587–1606. 1.2.4

[63] Kirsch U. *Design-oriented Analysis: a Unified Approach*. Springer Netherlands, 2002. 1.2.4

[64] Phelan DG, Haber RB. Sensitivity analysis of linear elastic systems using domain parametrization and a mixed mutual energy principle. *Computer Methods in Applied Mechanics and Engineering* 1989; **77**(1–2):31–59. 1.2.4

[65] Arora JS, Lee TH, Cardoso JB. Structural shape sensitivity analysis: relationship between material derivative and control volume approaches. *AIAA Journal* 1992; **30**(6):1638–1648. 1.2.4

[66] Arora JS. An exposition of the material derivative approach for structural shape sensitivity analysis. *Computer Methods in Applied Mechanics and Engineering* 1993; **105**(1):41–62. 1.2.4

[67] Lee BY. Consideration of body forces in axisymmetric design sensitivity analysis using the bem. *Computers & Structures* 1996; **61**(4):587–596. 1.2.4

[68] Lee BY. Direct differentiation formulation for boundary element shape sensitivity analysis of axisymmetric elastic solids. *International Journal of Solids and Structures* 1997; **34**(1):99–112. 1.2.4

[69] Navarrina F, López-Fontán S, Colominas I, Bendito E, Casteleiro M. High order shape design sensitivities: a unified approach. *Computer Methods in Applied Mechanics and Engineering* 2000; **188**(4):681–696. 1.2.4

[70] Choi KK, Duan W. Design sensitivity analysis and shape optimization of structural components with hyperelastic material. *Computer Methods in Applied Mechanics and Engineering* 2000; **187**(1–2):219–243. 1.2.4

[71] Kibsgaard S. Sensitivity analysis-the basis for optimization. *International Journal for Numerical Methods in Engineering* 1992; **34**(3):901–932. 1.2.4

[72] Poldneff MJ, Rai IS, Arora JS. Implementation of design sensitivity analysis for nonlinear structures. *AIAA Journal* 1993; **31**(11):2137–2142. 1.2.4

[73] Pandey PC, Bakshi P. Analytical response sensitivity computation using hybrid finite elements. *Computers & Structures* 1999; **71**(5):525–534. 1.2.4

[74] Moita JS, Infanta J, Mota CM. Sensitivity analysis and optimal design of geometrically non-linear laminated plates and shells. *Computers & Structures* 2000; **76**(1–3):407–420. 1.2.4

[75] Ozaki I, Kimura F, Berz M. Higher-order sensitivity analysis of finite element method by automatic differentiation. *Computational Mechanics* 1995; **16**(4):223–234. 1.2.4

[76] Wujek BA, Renaud JE. Automatic differentiation for more efficient system analysis and optimization. *Engineerign Optimization* 1998; **31**(1):101–139. 1.2.4

[77] Bischof C, Carle A, Khademi P, Mauer A. The adifor 2.0 system for the automatic differentiation of fortran 77 programs. *IEEE Computational Science and Engineering* 1996; **3**(3):18–32. 1.2.4

[78] Griewank A, Juedes D, Utke J. Adol-c, a package for the automatic differentiation of algorithms written in c/c++. *ACM Transactions on Mathematical Software (TOMS)* 1996; **22**(2):131–167. 1.2.4

[79] Shiriaev D, Griewank A. Adol-f: Automatic differentiation of fortran codes. *Computational Differentiation: Techniques, Applications, and Tools (SIAM)* 1996; **1**:375–384. 1.2.4

[80] Choi KK, Twu S. Equivalence of continuum and discrete methods of shape design sensitivity analysis. *AIAA Journal* 1989; **27**(10):1419–1424. 1.2.4

[81] Haftka RT, Barthelemy B. *Discretization Methods and Structural Optimization — Procedures and Applications: Proceedings of a GAMM-Seminar October 5–7, 1988, Siegen, FRG*, chap. On the Accuracy of Shape Sensitivity Derivatives. Springer: Berlin, Heidelberg, 1989; 136–144. 1.2.4

[82] Haftka R, Adelman H. Recent developments in structural sensitivity analysis. *Structural Optimization* 1989; **1**(3):137–151. 1.2.4

[83] Salmenjoki K, Neittaanmäki P. *Computer Aided Optimum Design of Structures: Recent Advances*, chap. Comparison of Various Techniques for Shape Design Sensitivity Analysis. Springer-Verlag: Berlin, Heidelberg, 1989; 367–377. 1.2.4

[84] van Keulen F, Haftka R, Kim N. Review of options for structural design sensitivity analysis. Part I: linear systems. *Computer Methods in Applied Mechanics and Engineering* 2005; **194**(30-33):3213–3243. 1.2.4

[85] Zienkiewicz OC, Taylor RL, Zhu JZ ((eds.)). *The Finite Element Method: Its Basis and Fundamentals*. Butterworth-Heinemann: Oxford, 2013. 2.2

[86] Hassan O, Probert EJ. *Grid Control and Adaptation*, chap. 35. CRC Press, Inc, 1999; 35.1–35.29. 3.1

[87] Johnson A, Tezduyar TE. Advanced Mesh Generation and Update Methods for 3D Flow Simulations. *Computational Mechanics* 1999; **23**:130–143. 3.1

[88] Ródenas JJ, Tarancón JE, Albelda J, Roda A, Fuenmayor FJ. Hierarchical Properties in Elements Obtained by Subdivision: a Hierarquical h-adaptivity Program. *Adaptive Modeling and Simulation 2005*, Díez P, Wiberg NE (eds.), 2005. 3.1, 3.4.3.2

[89] Sevilla R, Hassan O, Morgan K. The Use of Hybrid Meshes to Improve the Efficiency of a Discontinuous Galerkin Method for the Solution of Maxwell's Equations. *Computers & Structures* 2014; **137**:2–13. 3.1

[90] Kajiya JT. Ray Tracing Parametric Patches. *SIGGRAPH Comput. Graph.* 1982; **16**(3):245–254. 3.1.1

[91] Toth DL. On Ray Tracing Parametric Surfaces. *SIGGRAPH Comput. Graph.* 1985; **19**(3):171–179. 3.1.1

[92] Sweeney M, Bartels R. Ray tracing free-form b-spline surfaces. *IEEE Computer Graphics and Applications* 1986; **6**(2):41–49. 3.1.1

[93] Nishita T, Sederberg TW, Kakimoto M. Ray Tracing Trimmed Rational Surface Patches. *SIGGRAPH Comput. Graph.* 1990; **24**(4):337–345. 3.1.1

[94] Barth W, Stürzlinger W. Efficient ray tracing for Bezier and B-spline surfaces. *Computers & Graphics* 1993; **17**(4):423–430. 3.1.1

[95] Lorensen WE, Cline HE. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *ACM SIGGRAPH Computer Graphics* 1987; **21**(4):163–169. 3.1.2

[96] Sevilla R, Fernández-Méndez S, Huerta A. NURBS-enhanced Finite Element Method (NEFEM): A Seamless Bridge Between CAD and FEM. *Archives of Computational Methods in Engineering* 2011; **18**(4):441–484. 3.1.2, 3.10

[97] Sevilla R, Fernández-Méndez S, Huerta A. 3D-NURBS-enhanced Finite Element Method (NEFEM). *International Journal for Numerical Methods in Engineering* 2011; **88**(2):103–125. 3.1.2

[98] Sevilla R, Fernández-Méndez S, Huerta A. Comparison of High-order Curved Finite Elements. *International Journal for Numerical Methods in Engineering* 2011; **87**(8):719–734. 3.1.2

[99] Gordon WJ, Hall CA. Transfinite Element Methods: Blending-function Interpolation over Arbitrary Curved Element Domains. *Numer. Math.* 1973; **21**(2):109–129. 3.1.2

[100] Wandzura S, Xiao H. Symmetric Quadrature Rules on a Triangle. *Comput. Math. Appl.* 2003; **45**(12):1829–1840. 3.1.2

[101] Stenberg R. On some techniques for approximating boundary conditions in the finite element method. *Journal of Computational and Applied Mathematics* 1995; **63**(1–3):139–148. 3.2

[102] Jiang W, Annavarapu C, Dolbow J, Harari I. A robust nitsche's formulation for interface problems with spline-based finite elements. *International Journal for Numerical Methods in Engineering* 2015; **104**(7):676–696. 3.2

[103] Tur M, Albelda J, Nadal E, Ródenas JJ. Imposing dirichlet boundary conditions in hierarchical cartesian meshes by means of stabilized lagrange multipliers. *International Journal for Numerical Methods in Engineering* 2014; **98**(6):399–417. 3.2

[104] Ródenas JJ, Tur M, Fuenmayor FJ, Vercher A. Improvement of the superconvergent patch recovery technique by the use of constraint equations: the SPR-C technique. *International Journal for Numerical Methods in Engineering* 2007; **70**(6):705–727. 3.2, 3.2, 3.3.2, 3.4.2.1

[105] Tur M, Albelda J, Marco O, Ródenas JJ. Stabilized Method to Impose Dirichlet Boundary Conditions using a Smooth Stress Field. *Computer Methods in Applied Mechanics and Engineering* 2015; **296**:352–375. 3.2

[106] Sanders JD, Laursen TA, Dolbow JE. Approximate imposition of boundary conditions in immersed boundary methods. *International journal for numerical methods in engineering* 2009; **80**(11):1379–1405. 3.2

[107] Zienkiewicz OC, Zhu JZ. The superconvergent patch recovery and a posteriori error estimates. Part 1: The recovery technique. *International Journal for Numerical Methods in Engineering* 1992; **33**(7):1331–1364. 3.2, 3.3.2

[108] Abel JF, Shephard MS. An algorithm for multipoint constraints in finite element analysis. *International Journal for Numerical Methods in Engineering* 1979; **14**(3):464–467. 2

[109] Farhat C, Lacour C, Rixen D. Incorporation of linear multipoint constraints in substructure based iterative solvers. Part 1: a numerically scalable algorithm. *International Journal for Numerical Methods in Engineering* 1998; **43**(6):997–1016. 2

[110] Babuška I, Rheinboldt C. A-posteriori error estimates for the finite element method. *International Journal for Numerical Methods in Engineering* 1978; **12**:1597–1615. 3.3.2

[111] Ainsworth M, Oden JT. *A posteriori Error Estimation in Finite Element Analysis*. John Wiley & Sons, 2000. 3.3.2

[112] Díez P, Parés N, Huerta A. Recovering lower bounds of the error by postprocessing implicit residual a posteriori error estimates. *International Journal for Numerical Methods in Engineering* 2003; **56**(10):1465–1488. 3.3.2

[113] Gerasimov T, Rüter M, Stein E. 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* 2012; **90**(April):1118–1155. 3.3.2

[114] Ladevèze P, Leguillon D. Error estimate procedure in the finite element method and applications. *SIAM Journal on Numerical Analysis* 1983; **20**(3):485–509. 3.3.2

[115] Almeida Pereirea OJB, Moitinho de Almeida JP, Maunder EAW. Adaptive methods for hybrid equilibrium finite element models. *Computational Methods in Applied Mechanics and Engineering* 1999; **176**:19–39. 3.3.2

[116] Almeida Pereira OJB, Moitinho de Almeida JP. A posteriori error estimation for equilibrium finite elements in elastostatic problems. *Computer Assisted Mechanics and Engineering Sciences* 2001; **8**(2-3):439–453. 3.3.2

[117] Zienkiewicz OC, Zhu JZ. The superconvergent patch recovery and a posteriori error estimates. Part 2: Error estimates and adaptivity. *International Journal for Numerical Methods in Engineering* 1992; **33**(7):1365–1382. 3.3.2

[118] Fuenmayor FJ, Oliver JL. Criteria to achieve nearly optimal meshes in the h-adaptive finite element mehod. *International Journal for Numerical Methods in Engineering* 1996; **39**(23):4039–4061. 3.3.2

[119] Ródenas JJ, Fuenmayor FJ, Tarancón J. A Numerical Methodology to Assess the Quality of the Design Velocity Field Computation Methods in Shape Sensitivity Analysis. *International Journal for Numerical Methods in Engineering* 2003; **59**(13):1725–1747. 3.4.1

[120] Zhang WH, Beckers P. *Computer Aided Optimum Design of Structures: Recent Advances*, chap. Comparison of Different Sensitivity Analysis Approaches for Structural Shape Optimization. Springer-Verlag: Berlin, Heidelberg, 1989; 347–356. 3.4.1

[121] Choi KK, Chang KH. A Study of Design Velocity Field Computation for Shape Optimal Design. *Finite Elements in Analysis and Design* 1994; **15**(4):317–341. 3.4.1

[122] Ródenas JJ. Error de Discretización en el Cálculo de Sensibilidades mediante el Método de los Elementos Finitos. *PhD Thesis* 2001; . 3.4.1

[123] Choi K, Kim N. *Structural sensitivity analysis and optimization, mechanical engineering series, vol. 1*. Springer-Verlag: Berlin, Heidelberg, 2005. 3.4.1

[124] D Belegundu YM S Zhang, Salagame R. The Natural Approach for Shape Optimization with Mesh Distortion Control. *Technical Report*, Penn State University 1991. 3.4.1.1.1

[125] El-Sayed MEM, Zumwalt KW. Efficient design sensitivity derivatives for multiload case structures as an integrated part of finite element analysis. *Computers & Structures* 1991; **40**(6):1461–1467. 3.4.2.1

[126] Fuenmayor FJ, Oliver JL, Ródenas JJ. Extension of the Zienkiewicz-Zhu error estimator to shape sensitivity analysis. *International Journal for Numerical Methods in Engineering* 1997; **40**(8):1413–1433. 3.4.2.1

[127] Bugeda G, Oliver J. A General Methodology for Structural Shape Optimization Problems Using Automatic Adaptive Remeshing. *International Journal for Numerical Methods in Engineering* 1993; **36**(18):3161–3185. 3.4.3, 3.4.3.3, 3.4.3.3

[128] Bugeda G, Ródenas JJ, Oñate E. An integration of a low cost adaptive remeshing strategy in the solution of structural shape optimization problems using evolutionary methods. *Computers & Structures* 2008; **86**(13–14):1563–1578. 3.4.3, 3.4.3.3, 3.4.3.3

[129] Ródenas JJ, Corral C, Albelda J, Mas J, Adam C. Nested domain decomposition direct and iterative solvers based on a hierarchical h-adaptive finite element code. *Adaptive Modeling and Simulation 2007*, Runesson K, Díez P (eds.), Internacional Center for Numerical Methods in Engineering (CIMNE), 2007; 206–209. 3.4.3.2

[130] Nocedal J, Wright SJ. *Numerical Optimization, Second Edition*. Springer-Verlag New York, 2006. 3.4.3.3

Part II

Articles

PAPER A

Exact 3D boundary representation in finite element analysis based on Cartesian grids independent of the geometry

O. Marco, R. Sevilla, Y. Zhang, J. J. Ródenas and M. Tur

International Journal for Numerical Methods in Engineering
Volume 103, Issue 6, Pages 445–468, 2015
DOI: 10.1002/nme.4914

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.

Key words

Immersed Boundary Methods; Cartesian grids; NURBS; T-spline; Bézier extraction; NEFEM

Contents

1 Introduction **95**

2 Geometrical representation: from NURBS to T-spline **98**
2.1 NURBS fundamentals 98
2.2 T-spline fundamentals 101

3 Cartesian grids with exact representation of the immersed geometry **102**
3.1 Generation of the analysis mesh 103
3.2 Element classification and geometry-mesh intersection 104
3.3 Integration over subdomains 108

4 Problem formulation and numerical solution **113**
4.1 Boundary conditions 113

5 Numerical examples **115**
5.1 Numerical integration 115
5.2 Discretization error 118
5.2.1 Sphere defined by NURBS 120
5.2.2 Torus defined by NURBS 122
5.2.3 Torus defined by T-spline 123

6 Conclusions **125**

Bibliography **126**

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 the 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 appropriate meshing of that geometry 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) [1] 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 only one geometric model, which can be utilized directly as the analysis model. A newer CAD representation tool, the T-splines [2], 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 [3, 4, 5, 6, 7, 8].

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) [9] and the Generalized FEM (GFEM) [10] 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) [11], this approach enriches the numerical solution to represent singular stress fields near the

crack tip and discontinuities on the crack faces. The GFEM uses a similar rationale also considering the PUM to incorporate enrichment functions that characterize 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 term of Finite Elements in Ambient Space in [12]. Examples of these analysis techniques are the Immersed Boundary Method (IBM) and the Immersed Finite Element Method (IFEM). The IBM was introduced by Peskin [13] to alleviate the cost associated with remeshing in body-fitted techniques when simulating the flow around heart valves. Later developments including the IFEM [14] 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 [15] for a review. These methods have become very popular in the last decade within the computational bio-mechanics community, see for instance [16, 17, 18]. 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, $\Omega_{\texttt{Phys}}$. 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 [19]. The scientific community has realized about the need to integrate CAD systems with the numerical analysis tools and any attempt towards this integration will require from 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 [20, 21] or the NURBS-Enhanced Finite Element Method (NEFEM) [22, 23, 24] 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 [22] and references therein, but it has been rarely accounted for in immersed techniques until very recently [25, 26]. 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 [27] 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) [28] 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.* [29, 30] 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 [31] that drives the h-adaptive refinement process. Dirichlet boundary conditions are imposed using a stabilized Lagrange multipliers approach [32], 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 and 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 [33], NURBS [34, 35], subdivision surfaces [36] or T-splines [2]. 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 defined in parametric space on a so-called knot vector, Ξ. This is a set of non-decreasing real numbers in the parametric space representing coordinates (knots):

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

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 known as uniform if its knots are uniformly spaced and non-uniform otherwise. A knot vector is 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 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 $q = 1, \ldots, p$ and with $i = 1, \ldots, n + p + 1$. They are $\mathcal{C}^{p-1}$-continuous when the internal knots are not repeated. If a knot has multiplicity k, the basis is $\mathcal{C}^{p-k}$-continuous at that knot. Other properties of the basis functions are

- B-spline basis functions formed from open knot vectors are 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 $w_i \geq 0 \; \forall \; i$. This projection allows the exact representation of all common shapes, in particular conic sections like ellipses.

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 revealed 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 union of two patches separately created may require the insertion of many knots and nonlinear reparameterization of one or both patches. Furthermore, all NURBS refinement operations are global, so the knot lines extend throughout the entire domain when refining by inserting knots into the knot vectors of a surface, . 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

A tight integration of design and analysis requires a technology built on the smooth B-spline basis functions which can be locally refined and are 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-spline. 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 net allow T-junctions which are introduced during local refinement. T-splines does not have the superfluous control points present in NURBS models 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 $\eta_i = [\eta_{i_0}, \eta_{i_1}, \eta_{i_2}, \eta_{i_3}, \eta_{i_4}]$ [2, 4]. The algorithm used to infer knot vectors from a T-mesh is introduced in [37].

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 to decompose the domain into Bézier elements. The Bézier extraction operator maps a piecewise Bernstein polynominal basis onto a B-spline basis [38]. A Bézier extraction operator $\mathbf{E}$ is a linear operator such that

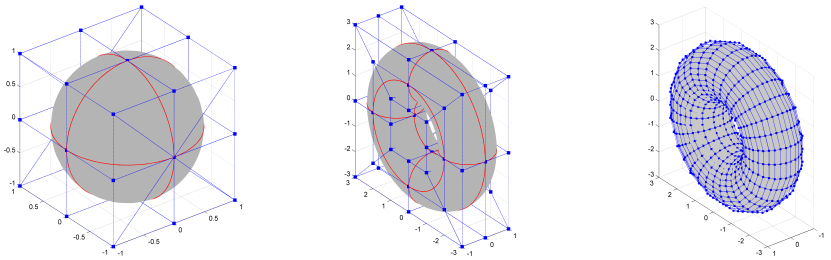
$$N(s) = \mathbf{E} B(s) \tag{11}$$

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 [39]. 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 vector B_t^e is generated all the T-spline basis functions with nonzero function values, and B_b^e is the vector of the Bézier basis functions. For each Bézier element, $\mathbf{M}^e$ can be calculated using the Oslo knot insertion algorithm [40]. 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.



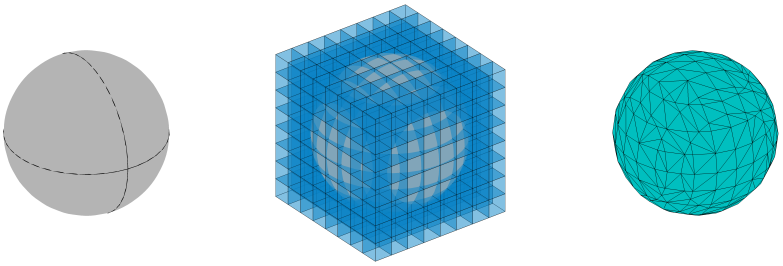
(a) NURBS sphere model. (b) NURBS torus model. (c) T-spline torus model.

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 costly and require a substantial amount of human hours [41, 42]. Know-how 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 $\Omega_{\texttt{Phys}} \subset \mathbb{R}^3$, see Figure 2a, with boundary $\Gamma_{\texttt{IB}} = \partial\Omega_{\texttt{Phys}}$, the key principle of FEAVox, or any other IBM, consists in 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



(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.

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 conversion of arbitrary geometrical CAD models into valid conforming FE discretizations is computationally expensive and difficult to fully automate depending on most cases of the user meshing skills. Immersed boundary methods do not need boundary-fitted meshes, but embed the domain into a Cartesian grid, which is generated independently of the geometric features 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. The different level of the Cartesian meshes are connected by predefined hierarchical relations. Regarding the initial mesh level, the user should choose that level based on the geometrical features of the model to avoid losing any information, such as small features or holes.

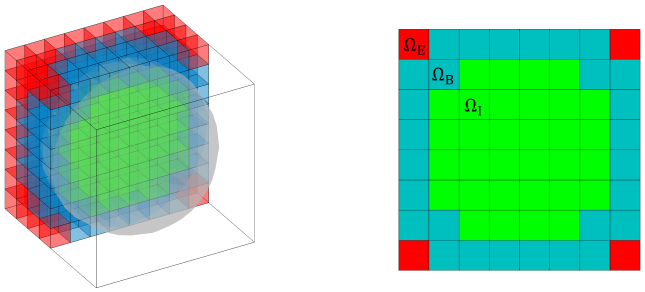
The term Cartesian grid pile, denoted by $\{\mathcal{Q}_h^i\}_{i=1,\dots,m}$, is used to define the sequence of m meshes utilized to discretize the embedding 3D domain Ω. For each level of refinement, the embedding domain Ω is partitioned in $\mathtt{n}_{\mathtt{el}}^i$ disjoint cubes of uniform size, where $\mathtt{n}_{\mathtt{el}}^{i+1} = 8\mathtt{n}_{\mathtt{el}}^i$. The present implementation is an extrapolation of the 2D data structure based on element subdivision shown in [43]. The data structure considers the hierarchical relations between the elements of different refinement levels, obtained during the element subdivision process, to accelerate FE computations. In FEAVox, the element used in the coarsest level of the Cartesian grid pile is called reference element and the data structure has been modified to the particular case of the Cartesian grid pile, where all elements are geometrically identical to the reference element. One important benefit of this data structure 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 evaluation of the elemental matrices. For instance, the analysis presented in [44] shows that the number of operations required to compute the elemental matrices can be reduced by a factor of 10 when a mapping with constant Jacobian is considered with low-order hexahedral elements. The hierarchical relationships considered in the data structure simplify the mesh refinement and the precomputation of most of the data used during the analyses, outstandingly improving the efficiency of the FE implementation. The code uses subroutines whose outputs are nodal coordinates, mesh topology, hierarchical relations, neighborship patterns, and other geometric information when required. Therefore, this information is not stored in memory, making the proposed algorithm more efficient, not only in terms of computational expense but also in terms of memory requirements.

3.2. Element classification and geometry-mesh intersection

The first step of the proposed strategy is the creation of the FE analysis mesh used to solve 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, Ω_E, not considered in the analysis, (2) in blue, boundary elements, Ω_B, intersected by the embedded domain and (3) in green interior elements, Ω_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 execute 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

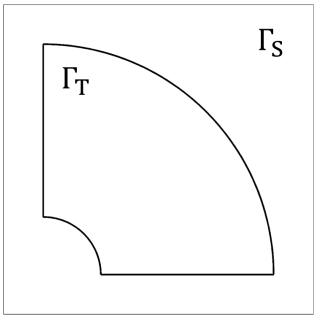
3. Classify the elements as internal, boundary or external.

We employ a Newton-Raphson algorithm to find the intersections between the edges of the elements of the analysis mesh and the parametric surfaces describing the boundary of the physical domain. Efficiency is guaranteed by using, as 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

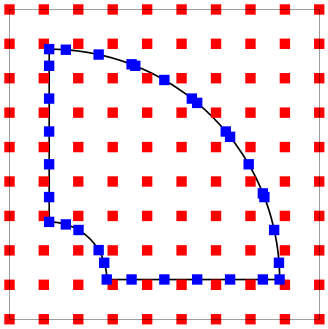
to remark that this triangular mesh is only an approximation of the exact CAD description of the physical boundary. However, this approximation is only used in our case to compute a good initial guess for the Newton-Raphson algorithm.

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 $\Gamma_{\mathtt{T}}$, (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 in $\Gamma_{\mathtt{T}}$ or outside $\Gamma_{\mathtt{T}}$. Obviously, for an appropriate representation, the points located over the boundary curves of $\Gamma_{\mathtt{T}}$ must include the extremes of these curves, but additional points must also be included to properly define the boundary. 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 $\Gamma_{\mathtt{T}}$ 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 $\Gamma_{\mathtt{T}}$.

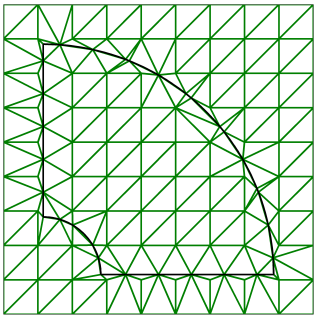
The above procedure requires the definition of an auxiliary triangulation in the parametric space of the NURBS surfaces defining the embedded domain. In order to guarantee convergence of the Newton-Raphson algorithm to the desired intersection point, it is advantageous to define a triangulation with 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



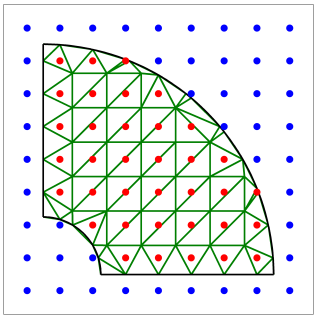
(a) Parametric space of a NURBS surface, Γ_S, and subspace to define a trimmed NURBS, Γ_T.



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



(c) Triangulation over the parametric space.

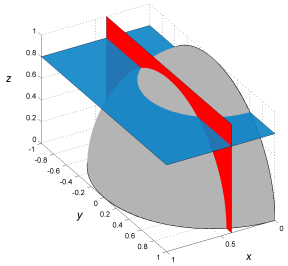


(d) Final auxiliary triangulation that ensures the correct classification of all intersection points.

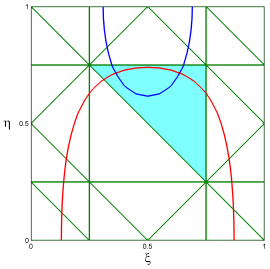
Figure 4: Identification of intersections between a trimmed NURBS surface and the edges of the Cartesian grid.

triangle highlighted in light blue. Figure 5c shows a finer triangulation that fixes this issue. To avoid the problem the size of the triangles has to be selected depending on the refinement of the Cartesian grid. In addition, we have to take into consideration special situations where the refinement of the elements will not be enough to choose the triangle size, for instance when some axes of the mesh are almost tangent to the surface. In the general case, we will subdivide the triangles where we can find more than one intersection until we reach the simple case with only one intersection per

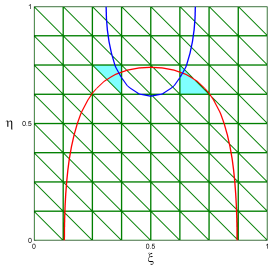
triangle. If we need to keep subdividing, the stopping criterion will be to reach a triangle size small enough to assume the axis is tangent to the geometry.



(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.

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.

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 internal domain 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.

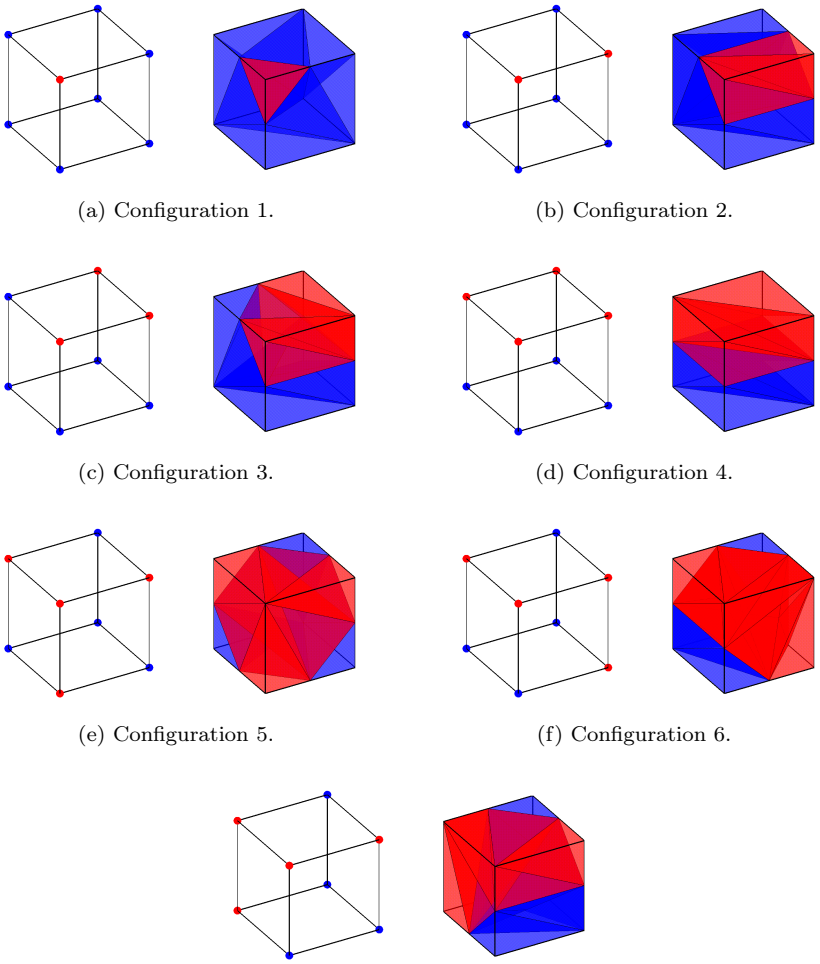
Internal elements are treated as standard finite elements and the integration is performed using a tensor product of one-dimensional Gauss quadratures with the desired number of points in each direction. However, the contribution from the boundary element Ω_B requires special attention as the integral 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 [45], 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 [45]). 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 multi-material problems).

Since the mesh is independent of the geometry, in most problems there will be elements intersected by several surfaces at the same time. To integrate properly the weak form in these elements, we have to ensure that the tetrahedralization will be consistent with the different parametric spaces of the NURBS. For these cases, the precomputed patterns based shown previously are not enough. A clear and common example of this situation is the existence of sharp features inside an element generated by the interfaces of connecting surfaces, see Figure 7a. The proposed method is to evaluate individually these elements generating specific sets of tetrahedra using a Delaunay procedure as in Figure 7b. We have to note that the ratio between the amount of elements with configurations not represented by the standard patterns and the number of elements in the mesh is very low, in general.

As we explained in the previous section, devoted to intersection, handling trimmed surfaces is not an issue because we can retrieve all the trimming information (surface and trimming curves) from the CAD models. Then, when we have elements with several surfaces whose interfaces are given by trimming curves we can always perform an specific tetrahedralization using a Delaunay procedure creating subdomains only in the domain delimited by the trimmed surfaces.



(a) Configuration 1.

(b) Configuration 2.

(c) Configuration 3.

(d) Configuration 4.

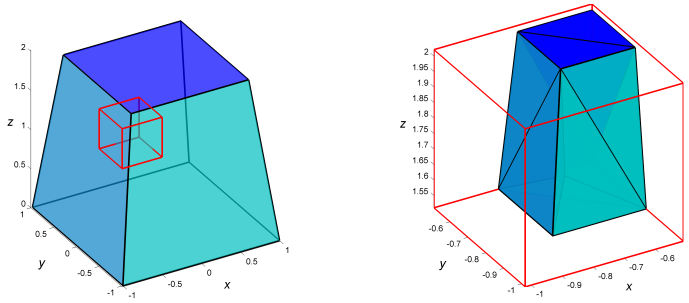
(e) Configuration 5.

(f) Configuration 6.

(g) Configuration 7.

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

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 [23] is adopted. This methodology was designed to incorporate the exact boundary of the computational domain into body-fitted FE simulations and the advantages with respect to the classical



(a) Element intersected by several patches of the model.

(b) Detail of the resulting tetrahedralization.

Figure 7: Tetrahedralization of an element intersecting a model of a trapezoidal prism.

FEM were demonstrated for a variety of problems, see [46]. A tetrahedral subdomain T_e^F with a face on the physical boundary is parametrized using the mapping

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

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

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

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 $\boldsymbol{\Psi}$ and $\boldsymbol{\Phi}$ 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 [47].

Given these parametrizations, it is possible to perform the numerical integration over all the curved tetrahedral subdomains that form $\Omega_{\mathtt{B}}^{\mathtt{phys}}$. To this end, we consider tensor products of triangle quadratures [48] and one-dimensional Gaussian quadratures for the tetrahedrons with a face on the boundary of the physical domain, see Figure 8.

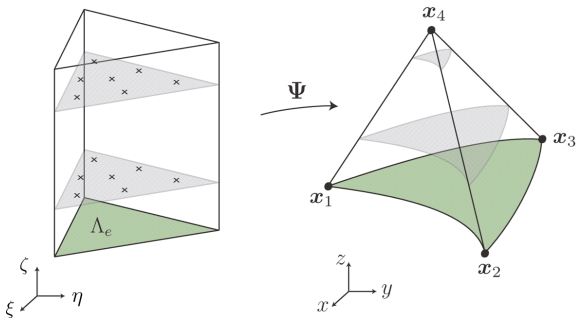


Figure 8: Integration over a curved tetrahedron with a face over the physical domain.

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 [49], 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 NEFEM defines the tetrahedral faces of curved tetrahedra on the parametric space of the NURBS, usually as straight-sided triangles, whereas in the current approach the tetrahedral faces are defined as the intersection of a Cartesian plane and a NURBS in the physical space. This means that, in the parametric space of the NURBS, tetrahedral faces are usually triangles with curved edges. It is worth noting that the mapping depicted in Figure 8 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 forced displacement on Γ_D.

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

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

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 IBM with Cartesian grids is the fact that, in general, the mesh nodes are not placed on the boundary of the domain. This increases the difficulty to apply Dirichlet boundary conditions in strong form.

In this paper, Dirichlet boundary conditions are imposed using stabilized Lagrange multipliers. More precisely, the procedure chosen to impose the constraints (i.e., Dirichlet boundary conditions) follows the technique proposed by [32]. 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. The stabilization term uses the FE stress field from a previous mesh [50] or a recovered stress field from a previous mesh or the current one [32]. In the second case, an iterative method is defined to solve the problem. In both cases, the definition of the Lagrange multipliers field allows for a direct condensation of the degrees of freedom of the multipliers at an element level. For the model problem of Equation (13), the weak formulation with Lagrange multipliers reads: find $\left(\mathbf{u}, \boldsymbol{\lambda}\right) \in \left[\mathcal{H}^1(\Omega_{\texttt{Phys}})\right]^3 \times \left[\mathcal{H}^{-1/2}(\Gamma_D)\right]^3$ such that

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

where

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

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

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

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 [50], i.e. $\mathbf{T}^* = -\boldsymbol{\sigma}^*\left(\mathbf{u}^h_{i-1}\right) \cdot \mathbf{n}$, where $\mathbf{n}$ is the vector normal to the Dirichlet boundary, and $\mathbf{u}^h_{i-1}$ 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 [32]. In both cases, the excessive oscillations of the Lagrange multipliers solution are prevented by the stabilization form 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 the geometrical aspects, 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 $\Gamma = \left\{(x, y, z) \mid x, y, z \geq 0,\ x^2 + y^2 + z^2 = 1\right\}$ 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 $100 \times \left(S_e(f) - S_h(f)\right) / S_e(f)$. 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

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

$$S_e(f = x^2) = S_e(f = y^2) = S_e(f = z^2) = \frac{\pi}{6},$$

$$S_e(f = xy) = S_e(f = xz) = S_e(f = yz) = \frac{1}{3}.$$

Tables 1 and 2 show the result of the numerical integration of 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 24 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 9a. If we increase the number of integration points to 224 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 9b. 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 $f(x, y, z) = y$.

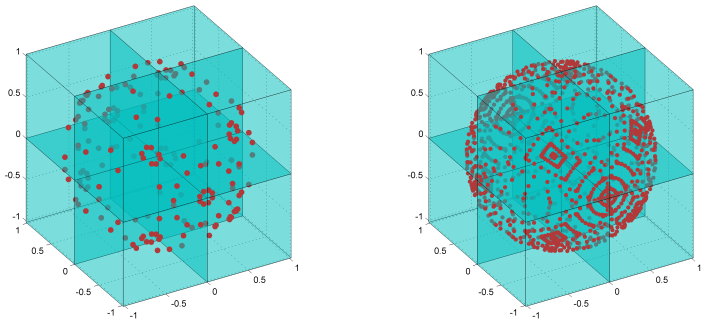
Gauss points	$f = 1$	Error (%)	$f = x$	Error (%)
32	$1,7847357662$	$13,6198$	$0,8923678831$	$13,6198$
64	$1,6538967700$	$5,2903$	$0,8269483850$	$5,2903$
192	$1,5672909334$	$0,2231$	$0,7836454667$	$0,2231$
384	$1,5708636716$	$0,0042$	$0,7854318358$	$0,0042$
640	$1,5707956384$	$4,38 \cdot 10^{-5}$	$0,7853978192$	$4,38 \cdot 10^{-5}$
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, 4 and 5show the result of the numerical integration of quadratic functions and the associated percentage error. Again, it can be observed that with 24 integration



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

Figure 9: Sphere defined by NURBS: Examples of the mesh used to evaluate the integration error with different number of quadrature points on the surface.

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 224 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.

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

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

Gauss points	$f = z^2$	Error (%)	$f = xy$	Error (%)
32	$0,5135503309$	$1,9191$	$0,5086633254$	$52,5989$
64	$0,5137816653$	$1,8749$	$0,3946299371$	$18,3889$
192	$0,5275430006$	$0,7532$	$0,3269680495$	$1,9095$
384	$0,5235375836$	$0,0116$	$0,3335154896$	$0,0546$
640	$0,5235986049$	$3,25 \cdot 10^{-5}$	$0,3333318950$	$4,31 \cdot 10^{-4}$
1344	$0,5235987699$	$1,08 \cdot 10^{-6}$	$0,3333333403$	$2,09 \cdot 10^{-6}$
1792	$0,5235987759$	$5,56 \cdot 10^{-8}$	$0,3333333330$	$1,01 \cdot 10^{-7}$

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

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

Table 5: Sphere defined by NURBS: Integration error on the surface integral for quadratic functions.

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 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-spline. 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

$$\begin{aligned} u_x &= x + x^2 - 2xy + x^3 - 3xy^2 + x^2 y \\ u_y &= -y - 2xy + y^2 - 3x^2 y + y^3 - xy^2 \\ u_z &= 0 \end{aligned} \tag{20}$$

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

$$\begin{aligned} \sigma_x &= \frac{E}{1+\nu}\left(1 + 2x - 2y + 3x^2 - 3y^2 + 2xy\right) \\ \sigma_y &= \frac{-E}{1+\nu}\left(1 + 2x - 2y + 3x^2 - 3y^2 + 2xy\right) \\ \sigma_z &= \nu\left(\sigma_x + \sigma_y\right) \\ \tau_{xy} &= \frac{E}{1+\nu}(-x - y + \frac{x^2}{2} - \frac{y^2}{2} - 6xy) \\ \tau_{xz} &= \tau_{yz} = 0 \end{aligned}$$

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

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

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

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

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 6 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.

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

Table 6: Sphere defined by NURBS: Information about the calculation meshes.

Table 7 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 6. 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	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 7: Sphere defined by NURBS: discretization errors and convergence rates using linear (L8) and quadratic (Q20) elements.

The results shown in Table 7 can be seen in Figure 10 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 [51]. In particular, the comparison in Figure 10 shows that the error attained with linear

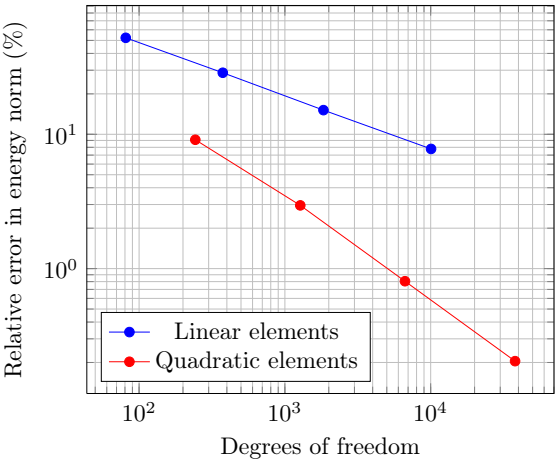
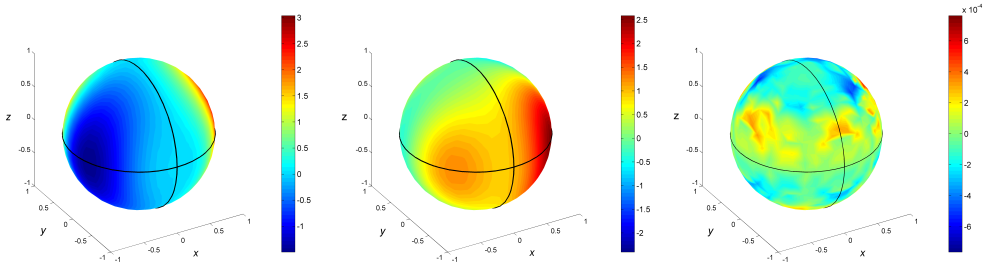


Figure 10: Sphere defined by NURBS: discretization error vs. degrees of freedom for linear and quadratic elements.

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 11. 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.



(a) x displacements. (b) y displacements. (c) z displacements.

Figure 11: Sphere defined by NURBS: Computed displacement field.

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.

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 8 summarizes the main features of these four computational meshes.

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

Table 8: Torus defined by NURBS: Information about the calculation meshes.

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 8. 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 9: Torus defined by NURBS: discretization errors and convergence rates using linear (L8) and quadratic (Q20) elements.

Figure 12 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 13. 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

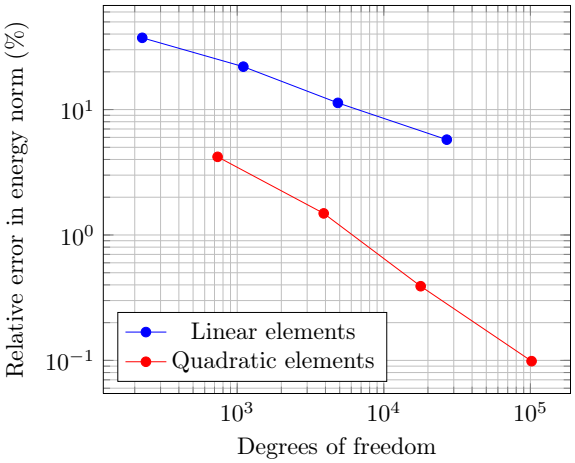
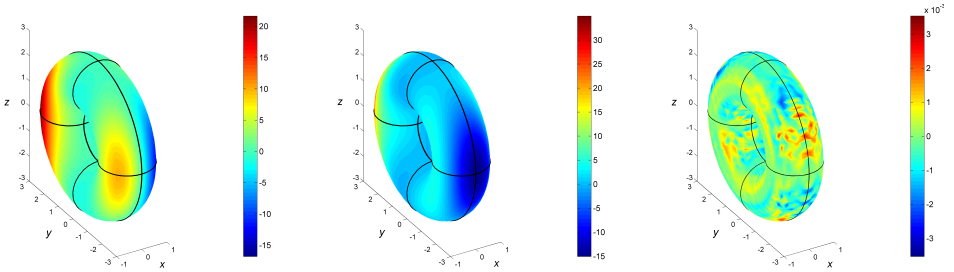


Figure 12: Torus defined by NURBS: discretization error vs degrees of freedom.

the error due to the imposition of the Dirichlet boundary condition in weak form by using the technique described in Section 4.



(a) x displacements. (b) y displacements. (c) z displacements.

Figure 13: Torus defined by NURBS: computed displacement field.

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 8. Table 10 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 8. The convergence rate of the

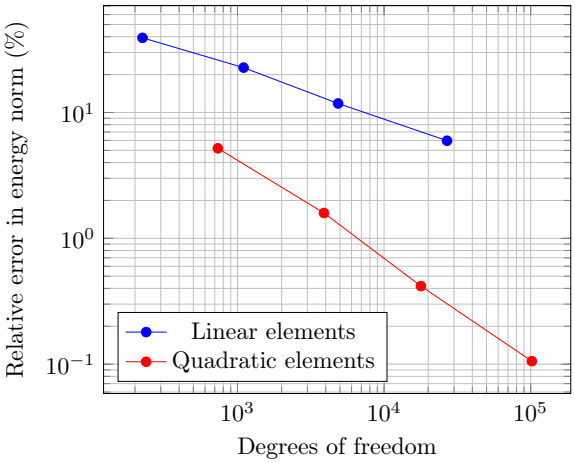


Figure 14: Torus defined by T-spline: discretization error vs degrees of freedom.

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

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

Figure 14 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 the CAD technology employed to represent the geometry of the embedded domain.

6. Conclusions

This paper proposes a novel methodology to consider the exact 3D boundary representation of the domain in a immerse boundary framework where a Cartesian grid is used to mesh the embedding domain. 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 IBM due to the approximation of the embedded geometry by a faceted representation.

The strategy to compute the intersections between the Cartesian grid and the exact geometry of the boundary of the embedded domain is detailed. 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 in defining, for integration purposes only, a tetrahedral submesh in each of the elements cut by the boundary where the tetrahedral have at most one face or one edge in contact with the boundary of the embedded domain. Then, specifically designed numerical quadratures are defined in the tetrahedra 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 has been analyzed using numerical examples.

One crucial aspect in IBM 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 in using stabilized Lagrange multipliers to impose Dirichlet 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 in terms of error convergence rate, 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 has been presented using linear elasticity problems but it could be adapted to solve any problem where immersed boundaries show advantages with respect to standard methods. In addition, a refinement strategy to generate an optimal h-adapted mesh from the geometrical point of view just taking into account the data contained in the CAD file is under development. This technique will be fully integrated with the techniques proposed in this paper.

Acknowledgments

With the support of the European Union Framework Program (FP7) under grant No. 289361 INSIST, Ministerio de Economía y Competitividad of Spain (DPI2010-20542)(DPI2013-46317-R), FPI program (BES-2011-044080) and Generalitat Valenciana (PROMETEO/2012/023). R. Sevilla gratefully acknowledges the financial support provided by the Sêr Cymru National Research Network in Advanced Engineering and Materials. Y. Zhang was supported in part by the PECASE Award N00014-14-1-0234 and NSF CAREER Award OCI-1149591.

References

[1] Hughes TJR, Cottrell JA, Bazilevs Y. Isogeometric Analysis: CAD, Finite Elements, NURBS, Exact Geometry, and Mesh Refinement. *Computer Methods in Applied Mechanics and Engineering* 2005; **194**:4135–4195. 1

[2] Sederberg TW, Zheng J, Bakenov A, Nasri A. T-splines and T-NURCCs. *ACM Transactions on Graphics (TOG)* 2003; **22**(3):477–484. 1, 2, 2.2

[3] Escobar JM, Cascon JM, Rodriguez E, Montenegro R. A New Approach to Solid Modeling with Trivariate T-splines Based on Mesh Optimization. *Computer Methods In Applied Mechanics And Engineering* 2011; **200**:3210–3222. 1

[4] Wang W, Zhang Y, Scott MA, Hughes TJR. Converting an Unstructured Quadrilateral Mesh to a Standard T-spline Surface. *Computational Mechanics* 2011; **4**(48):477–498. 1, 2.2

[5] Zhang Y, Wang W, Hughes TJR. Solid T-spline Construction from Boundary Representations for Genus-Zero Geometry. *Computer Methods in Applied Mechanics and Engineering* 2012; **249–252**:185–197. 1

[6] Zhang Y, Wang W, Hughes TJR. Conformal Solid T-spline Construction from Boundary T-spline Representations. *Computational Mechanics* 2013; **6**(51):1051–1059. 1

[7] Liu L, Zhang Y, Hughes TJR, Scott MA, Sederberg TW. Volumetric T-spline Construction using Boolean Operations. *Engineering with Computers* 2014; **30**(4):425–439. 1

[8] Liu L, Zhang Y, Liu Y, Wang W. Feature-Preserving T-mesh Construction using Skeleton-Based Polycubes. *Computer-Aided Design* 2014; **58**:162–172. 1

[9] Dolbow J, Moës N, Belytschko T. Discontinuous Enrichment in Finite Elements with a Partition of Unity Method. *Finite Elements in Analysis and Design* 2000; **36**(3-4):235–260. 1

[10] Strouboulis T, Babuška I, Copps K. The Design and Analysis of the Generalized Finite Element Method. *Computer Methods in Applied Mechanics and Engineering* 2000; **181**(1-3):43–69. 1

[11] Melenk JM, Babuška I. The Partition of Unity Finite Element Method: Basic Theory and Applications. *Computer Methods in Applied Mechanics and Engineering* 1996; **139**(1-4):289–314. 1

[12] Bordas SPA, Rabczuk T, Ródenas JJ, Kerfriden P, Moumnassi M, Belouettar S. Recent Advances Towards Reducing the Meshing and Re-meshing Burden in Computational Sciences. *Computational Technology Reviews* 2010; **2**:51–82. 1

[13] Peskin CS. Numerical Analysis of Blood Flow in the Heart. *Journal of Computational Physics* 1977; **25**:220–252. 1

[14] Zhang L, Gerstenberger A, Wang X, Liu WK. Immersed Finite Element Method. *Computer Methods in Applied Mechanics and Engineering* 2004; **293**(21):2051–2067. 1

[15] Mittal R, Iaccarino G. Immersed Boundary Methods. *Annual Review of Fluid Mechanics* 2005; **37**:239–261. 1

[16] Liu WK, Liu Y, Darell D, Zhang L, Wang XS, Fukui Y, Patankar N, Zhang Y, Bajaj C, Lee J, *et al.*. Immersed Finite Element Method and its Applications to Biological Systems. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**(13):1722–1749. 1

[17] Liu WK, Tang S. Mathematical Foundations of the Immersed Finite Element Method. *Computational Mechanics* 2007; **39**(3):211–222. 1

[18] Gil AJ, Arranz-Carreño A, Bonet J, Hassan O. The Immersed Structural Potential Method for Haemodynamic Applications. *Journal of Computational Physics* 2010; **229**(22):8613–8641. 1

[19] Cirak F, Ortiz M, Schröder P. Subdivision Surfaces: a New Paradigm for Thinshell Finite Element Analysis. *International Journal for Numerical Methods in Engineering* 2000; **47**(12):2039–2072. 1

[20] Inoue K, Kikuchi Y, Masuyama T. A NURBS Finite Element Method for Product Shape Design. *J. Engrg. Design* 2005; **16**(2):157–174. 1

[21] Cottrell JA, Hughes TJR, Bazilevs Y. *Isogeometric Analysis: Toward Integration of CAD and FEA*. Wiley, 2009. 1

[22] Sevilla R, Fernández-Méndez S, Huerta A. NURBS-enhanced Finite Element Method (NEFEM): A Seamless Bridge Between CAD and FEM. *Archives of Computational Methods in Engineering* 2011; **18**(4):441–484. 1

[23] Sevilla R, Fernández-Méndez S, Huerta A. 3D-NURBS-enhanced Finite Element Method (NEFEM). *International Journal for Numerical Methods in Engineering* 2011; **88**(2):103–125. 1, 3.3

[24] Legrain G. A NURBS-enhanced Extended Finite Element Approach for Unfitted CAD Analysis. *Computational Mechanics* 2013; **52**(4):913–929. 1

[25] Rüberg T, Cirak F. Subdivision-stabilised Immersed B-spline Finite Elements for Moving Boundary Flows. *Computer Methods in Applied Mechanics and Engineering* 2012; **209-212**:266–283. 1

[26] Legrain G. A NURBS-enhanced Extended Finite Element Method. *Computational Mechanics* 2013; **52**(4):913–929. 1

[27] Kim HJ, Seo YD, Youn SK. Isogeometric Analysis with Trimming Technique for Problems of Arbitrary Complex Topology. *Computer Methods in Applied Mechanics and Engineering* 2010; **199**(45–48):2796–2812. 1

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

[29] Nadal E, Ródenas JJ, Albelda J, Tur M, Tarancón JE, Fuenmayor FJ. Efficient Finite Element Methodology based on Cartesian Grids: Application to Structural Shape Optimization. *Abstract and Applied Analysis* 2013; **2013**. 1

[30] Nadal E. *Cartesian Grid FEM (cgFEM): High Performance h-adaptive FE Analysis with Efficient Error Control. Application to Structural Shape Optimization. PhD Thesis*. Universitat Politècnica de València, 2014. 1

[31] Zienkiewicz OC, Zhu JZ. A Simple Error Estimator and Adaptive Procedure for Practical Engineering Analysis. *International Journal for Numerical Methods in Engineering* 1987; **24**(2):337–357. 1

[32] Tur M, Albelda J, Marco O, Ródenas JJ. Stabilized Method to Impose Dirichlet Boundary Conditions using a Smooth Stress Field. *Computer Methods in Applied Mechanics and Engineering* 2015; **296**:352–375. 1, 4.1, 4.1

[33] de Boor C. *A Practical Guide to Splines*. Springer-Verlag, 1978. 2

[34] Piegl L, Tiller W. *The NURBS Book*. Springer-Verlag, 1995. 2

[35] Rogers DF. *An Introduction to NURBS: with Historical Perspective*. Elsevier, 2001. 2

[36] Jörg P, Ulrich R. *Subdivision Surfaces*, vol. 3. Springer-Verlag, 2008. 2

[37] Sederberg TW, Cardon DL, Finnigan GT, North NS, Zheng J, Lyche T. T-spline Simplification and Local Refinement. *ACM SIGGRAPH*, 2004; 276–283. 2.2

[38] Borden MJ, Scott MA, Evans JA, Hughes TJR. Isogeometric Finite Element Data Structures Based on Bézier Extraction of NURBS. *International Journal for Numerical Methods in Engineering* 2011; **87**:15–47. 2.2

[39] Wang W, Zhang Y, Xu G, Hughes TJR. Converting an Unstructured Quadrilateral/hexahedral Mesh to a Rational T-spline. *Computational Mechanics* 2012; **50**(1):65–84. 2.2

[40] Goldman R, Lyche T. *Knot Insertion and Deletion Algorithms for B-spline Curves and Surfaces*. Society for Industrial and Applied Mathematics, Philadelphia, 1993. 2.2

[41] Hassan O, Probert EJ. *Grid Control and Adaptation*, chap. 35. CRC Press, Inc, 1999; 35.1–35.29. 3

[42] Johnson A, Tezduyar TE. Advanced Mesh Generation and Update Methods for 3D Flow Simulations. *Computational Mechanics* 1999; **23**:130–143. 3

[43] Ródenas JJ, Tarancón JE, Albelda J, Roda A, Fuenmayor FJ. Hierarchical Properties in Elements Obtained by Subdivision: a Hierarquical h-adaptivity Program. *Adaptive Modeling and Simulation 2005*, Díez P, Wiberg NE (eds.), 2005. 3.1

[44] Sevilla R, Hassan O, Morgan K. The Use of Hybrid Meshes to Improve the Efficiency of a Discontinuous Galerkin Method for the Solution of Maxwell's Equations. *Computers & Structures* 2014; **137**:2–13. 3.1

[45] Lorensen WE, Cline HE. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *ACM SIGGRAPH Computer Graphics* 1987; **21**(4):163–169. 3.3

[46] Sevilla R, Fernández-Méndez S, Huerta A. Comparison of High-order Curved Finite Elements. *International Journal for Numerical Methods in Engineering* 2011; **87**(8):719–734. 3.3

[47] Gordon WJ, Hall CA. Transfinite Element Methods: Blending-function Interpolation over Arbitrary Curved Element Domains. *Numer. Math.* 1973; **21**(2):109–129. 3.3

[48] Wandzura S, Xiao H. Symmetric Quadrature Rules on a Triangle. *Comput. Math. Appl.* 2003; **45**(12):1829–1840. 3.3

[49] Sevilla R, Fernández-Méndez S. Numerical Integration over 2D NURBS Shaped Domains with Applications to NURBS-enhanced FEM. *Finite Elements in Analysis and Design* 2011; **47**(10):1209–1220. 3.3

[50] Tur M, Albelda J, Nadal E, Ródenas JJ. Imposing dirichlet boundary conditions in hierarchical cartesian meshes by means of stabilized lagrange multipliers. *International Journal for Numerical Methods in Engineering* 2014; **98**(6):399–417. 4.1, 4.1

[51] Szabó B, Babuška I. *Finite Element Analysis*. John Wiley & Sons, 1991. 5.2.1

PAPER B

Stabilized method of imposing Dirichlet boundary conditions using a recovered stress field

M. Tur, J. Albelda, O. Marco and J. J. Ródenas

Computer Methods in Applied Mechanics and Engineering
Volume 296, Pages 352-375, 2015
DOI: 10.1016/j.cma.2015.08.001

Abstract

This paper proposes a new formulation to impose Dirichlet boundary conditions on immersed boundary Cartesian Finite Element meshes. The method uses a recovered stress field calculated by Superconvergent Patch Recovery to stabilize the Lagrange multiplier formulation of the problem. The optimal convergence of the method and the convergence of the proposed iterative procedure are demonstrated. The proposed method is also suitable for problems with non-linear material behavior. Some numerical examples are included to confirm the theoretical results.

Key words

Dirichlet boundary conditions; Lagrange multipliers; Stabilization; Immersed boundary method; Cartesian grid

Contents

1 Introduction **135**

2 Statement of the problem **137**
2.1 Finite element formulation 139

3 Stabilized methods **140**
3.1 Nitsche's method 140
3.2 Proposed method 142
3.3 Iterative Nitsche's method 144
3.4 Plasticity 144

4 Interpolation and numerical integration **145**

5 Recovered stress field **146**

6 Convergence of the finite element solution **151**
6.1 Coercivity 151
6.2 Convergence of the iterative method 152
6.3 Stability of the formulation 153
6.4 Optimal convergence 154

7 Numerical examples **155**
7.1 Example 1: Tilted hexahedron 156
7.2 Example 2: Spherical domain 159
7.3 Example 3: Cubic domain parallel to the Cartesian grid 161
7.4 Example 4: Plasticity 164

8 Conclusions **166**

Bibliography **167**

1. Introduction

The Finite Element Method (FEM) makes use of a mesh of elements to perform the analysis that will provide the numerical solution to the problem under consideration. In the standard version of the FEM the mesh must conform to the geometry of the domain to be analyzed. At the same time, the distortion of each of the elements with respect to the reference elements in the normalized space must be kept sufficiently low, as high element distortion leads to inaccurate results. As a result, it is no simple task to create an appropriate mesh for Finite Element Analysis (FEA). According to [1], a study at Sandia National Laboratories (USA) revealed that the generation of the Finite Element (FE) numerical model, including the process of creating a geometry suitable for analysis by the FEM and the subsequent geometry meshing, takes 80% of the total analysis time, whereas only 20% is devoted to the numerical analysis itself, which provides the solution to the problem. Under the umbrella term of *finite elements in ambient space* [2] we can classify a number of variants of the standard FEM which have recently gained in popularity because they reduce the computational cost to generate the FE model by making the mesh independent of the geometry of the problem. These techniques have been given several names in the literature, such as *Fictitious Domain*, *Implicit Meshing*, *Immersed FEM*, *Immersed Boundary Method*, *Fixed grid FEM* and *Cartesian grid FEM* (cgFEM). In these techniques an auxiliary domain Ω_E with a simple geometry that embeds the problem domain Ω is used for the FE discretization. Because of the simple geometry used to define Ω_E (normally a square in 2D or a cube for the 3D case) its subdivision into elements is very simple, thus reducing the meshing burden. There are two main issues that clearly distinguish these methods from the standard FEM: integration and imposition of boundary conditions, in particular the Dirichlet boundary conditions.

Integration: As the mesh does not conform to the geometry of the domain it is necessary to use special procedures to evaluate integrals. Different approaches have been considered in the bibliography to ensure that the integration in each element is only extended to the exact part of the volume (area in 2D), see for example references [3, 4, 5]. In general terms the solution to this problem consists of using two different meshes, one for interpolation and another for integration. In order to maintain the optimal convergence rate of the FE solution, the degree of approximation to the boundary must be at least of the same order as the degree of the FE interpolation [6]. Transfinite mapping techniques commonly used in the p-version of the FEM, or the integration techniques described in [7] to consider the exact geometry given by a NURBS representation of the boundary, can be used in the elements cut by the boundary to obtain an exact representation of the domain.

Boundary conditions: as the FE nodes do not generally lie on the boundary, the procedures used in the standard FEM to apply the boundary conditions cannot be

used. The case of the Neumann boundary conditions can be easily tackled by simply taking into account that the integration surface can cut the element and does not necessarily have to coincide with the element faces. However, the case of the Dirichlet boundary conditions is much more complex. To solve the problem, a common alternative is to use the Lagrange multiplier technique. It can be difficult to find compatible discretizations of displacements and multipliers that satisfy the *InfSup* condition [8] but 2D [9] and 3D [10] methods of doing so can be found in the bibliography. However, the naive choice for the multiplier interpolation based on the element edge intersection does not satisfy the *InfSup* condition. The main problem appears because the number of Lagrange multipliers is too high, which can cause undesired oscillations in the Lagrange multiplier field. One of the alternatives is thus to stabilize the solution [11, 12].

One of the most popular methods to stabilize the solution is by Nitche's method, which can be derived from the Barbosa-Hughes stabilization [11]. For early implementations of the method in immersed boundaries see for example [13, 14, 15], or for a comparison with other methods see [10, 16]. The stabilizing term in Nitsche's method has an algorithmic constant to be defined that affects the stability of the method. As pointed out in the bibliography [10, 17, 18], this constant depends on how the boundary intersects the underlying mesh and it can become unbounded for certain configurations. At the same time very large values of the penalty constant result in an ill-conditioned system [19]. This issue has been treated for interface and X-FEM problems by choosing appropriate values for each intersected element [20, 21, 22, 23]. In these works the stabilizing term is the finite element stress field, like in the Barbosa-Hughes stabilization, but computed in a different way. In [20] the stress is computed in the element partition with larger size. In the weighted Nitsche's method [21, 23] the stress field of both partitions are weighted using the partition sizes.

Some variations of Nitsche's method have been proposed in recent years to overcome this problem for imposing Dirichlet constraints. In [17] the solution of the internal elements is extended to the boundary elements with a very small volume/area ratio. The same idea was exploited in [24]. In [18] and [25] the flux jump across the boundary element edges is used to modify the stabilized problem. In [26] the solution of a coarser mesh was used as stabilizing stress. Other stabilizing techniques not directly derived from the Nitsche's method have also been proposed. In [27] the discontinuous-Galerkin method was used. In [28] a polynomial stabilization first proposed in [29] was applied for solving contact problems. Similar ideas where used in [30] where the stabilizing term is a suitable projection of the Lagrange multiplier field. In the method proposed in [31] the Lagrange multipliers field is defined in all the domain of the boundary elements and an optimal value of the penalty parameter is proposed regardless of mesh size.

The aim of this work is to propose a stabilized formulation using a recovered stress field (SPR-type recovered solution, Superconvergent Patch Recovery [32]) iter-

atively obtained from the finite element solution. We present theoretical values for the stabilizing constant, for linear and quadratic discretizations, that ensure optimal convergence rate. The theoretical analyses are done for the linear elastic case, although it will be also shown that the proposed method can be used to impose Dirichlet boundary conditions in problems with non-linear material behavior, such as plasticity. The paper is organized as follows. Section 2 presents the linear elasticity problem and its mixed finite element implementation. Section 3 presents the stabilized formulation in the context of Cartesian grids. Section 4 describes the finite element interpolation. Section 5 describes the adaptation of the SPR technique required by the proposed method and the evaluation of the stabilization term. Section 6 demonstrates the convergence and stability of the method. The numerical results are given in Section 7, followed by the main conclusions of this work.

2. Statement of the problem

Let us consider the linear elastic problem. Let $\Omega \in \mathbb{R}^d$, with $d = 2$ or $d = 3$ be a bounded domain with a sufficiently smooth boundary Γ. The contour can be divided into two non-overlapping parts, Γ_{D} and Γ_{N}, where Dirichlet and Neumann conditions are respectively imposed. The aim is to find the displacement field $\mathbf{u} \in \mathscr{U}$ that fulfills internal equilibrium equations in the domain and the Dirichlet and Neumann boundary conditions on the boundary, which can be written as follows:

$$\begin{aligned} &\nabla \boldsymbol{\sigma}(\mathbf{u}) + \mathbf{t}_v = 0 \qquad \text{in} \quad \Omega \\ &\boldsymbol{\sigma}(\mathbf{u})\,\mathbf{n} = \mathbf{t}_s \qquad \text{on} \quad \Gamma_{\mathrm{N}} \\ &\mathbf{u} = \mathbf{g} \qquad \text{on} \quad \Gamma_{\mathrm{D}} \\ &\boldsymbol{\epsilon}(\mathbf{u}) = \mathscr{D}\,\boldsymbol{\sigma}(\mathbf{u}) \end{aligned} \tag{1}$$

In the above expression $\mathbf{t}_v \in [L^2(\Omega)]^d$ are the volume forces, $\mathbf{t}_s \in [L^2(\Omega)]^d$ the tractions imposed on the Neumann boundary, $\mathbf{g}$ the displacements imposed on the Dirichlet boundary and $\mathbf{n}$ the unit normal vector. $\mathscr{U} \equiv [H^1(\Omega)]^d$ is the Hilbert space of functions whose integral of the first derivative over the domain is bounded. In linear elasticity, strain tensor is defined from displacement field by

$$\boldsymbol{\epsilon}(\mathbf{u}) = \left(\nabla \mathbf{u} + \nabla^{\mathrm{T}} \mathbf{u}\right)/2 \tag{2}$$

The constitutive equation, which relates the strains with the stresses by means of the tensor $\mathscr{D}$, can be expressed using two material dependent constants, the Young's

modulus E and the Poisson ratio ν, in the case of isotropic behavior. This relationship can be written as

$$\boldsymbol{\epsilon} = \left(\boldsymbol{\sigma} - \nu \left(\operatorname{tr}\left(\boldsymbol{\sigma}\right)\mathbf{I} - \boldsymbol{\sigma}\right)\right)/E \tag{3}$$

It is straightforward to show the following property concerning the constitutive equation, which will be used below.

Property 1. *The scalar product of the tractions can be bounded by the energy per unit volume with a constant C_E, which depends on the material properties, as*

$$\|\boldsymbol{\sigma}(\mathbf{u})\|^2 \leq C_E \left(\boldsymbol{\sigma}(\mathbf{u}) : \boldsymbol{\epsilon}(\mathbf{u})\right) \qquad \textit{with} \qquad C_E = \frac{E}{1-2\nu} \tag{4}$$

The weak variational formulation of the elastic problem allows two approaches to imposing the Dirichlet boundary conditions. The most common procedure is to impose a constraint in the space of virtual displacement $\mathscr{V}$, i.e. the virtual displacement is zero on the Dirichlet boundary. The virtual work of the elastic forces is in equilibrium with the virtual work of the external forces applied, as follow

$$\begin{aligned} & a\left(\mathbf{u},\mathbf{v}\right) = c\left(\mathbf{v}\right) \qquad \forall \mathbf{v} \in \mathscr{V} \\ \text{where} \quad & a\left(\mathbf{u},\mathbf{v}\right) = \int_{\Omega} \boldsymbol{\sigma}\left(\mathbf{u}\right) : \boldsymbol{\epsilon}\left(\mathbf{v}\right)\, d\Omega \\ & c\left(\mathbf{v}\right) = \int_{\Omega} \mathbf{v}\cdot\mathbf{t}_v\, d\Omega + \int_{\Gamma_{\mathrm{N}}} \mathbf{v}\cdot\mathbf{t}_s\, d\Gamma \end{aligned} \tag{5}$$

This method is simple to implement and effective in the context of the standard finite element method in which the geometry boundary is properly represented by the mesh. However, for Cartesian meshes this method is not valid, since it is very difficult to get a null field on the Dirichlet boundary if the contour of the geometry does not match with the edge of the elements.

For this reason it seems more appropriate to seek another formulation, which involves raising the elastic problem as a minimization with constraints. This means finding the displacement field $\mathbf{u}$ that minimizes the total potential energy, subject to the constraints imposed by Dirichlet conditions. The problem can be expressed as

$$\begin{aligned} & \min \frac{1}{2} a\left(\mathbf{v},\mathbf{v}\right) - c\left(\mathbf{v}\right) \\ & \text{with} \quad \mathbf{v} = \mathbf{g} \text{ in } \Gamma_{\mathrm{D}} \end{aligned} \tag{6}$$

One approach to solving this minimization problem is to use the Lagrange multiplier method. In addition to the displacement field, a new field of Lagrange multipliers

$\boldsymbol{\lambda}$ associated with the reaction forces is added. Formally, the problem is to find the saddle point $[\mathbf{u}, \boldsymbol{\lambda}] \in \mathscr{U} \times \mathscr{M}$ of the following Lagrangian

$$\mathscr{L}(\mathbf{v}, \boldsymbol{\mu}) = \frac{1}{2} a(\mathbf{v}, \mathbf{v}) + b(\boldsymbol{\mu}, \mathbf{v} - \mathbf{g}) - c(\mathbf{v}) \tag{7}$$

where the Lagrange multipliers belong to the Hilbert space $\mathscr{M} = [H^{-\frac{1}{2}}(\Gamma_{\mathrm{D}})]^d$ and the following functional is defined

$$b(\cdot, \cdot) : \mathscr{M} \times \mathscr{U} \to \mathbb{R} \qquad b(\boldsymbol{\mu}, \mathbf{u}) = \int_{\Gamma_{\mathrm{D}}} \boldsymbol{\mu} \cdot \mathbf{u} \, d\Gamma \tag{8}$$

2.1. Finite element formulation

The domain is subdivided into finite elements by a Cartesian mesh in which the boundary of the domain does not necessarily coincide with the edge of an element, but can pass through it. The spaces of the finite element solution are denoted as $\mathscr{U}^h \subset \mathscr{U}$ for displacements and $\mathscr{M}^h \subset \mathscr{M}$ for multipliers. Substituting the finite element fields in Equation (7) and optimizing the Lagrangian we find the following system:

$$\begin{aligned} a\left(\mathbf{u}^h, \mathbf{v}^h\right) + b\left(\boldsymbol{\lambda}^h, \mathbf{v}^h\right) &= c\left(\mathbf{v}^h\right) \qquad && \forall \mathbf{v}^h \in \mathscr{U}^h \\ b\left(\boldsymbol{\mu}^h, \mathbf{u}^h\right) &= b\left(\boldsymbol{\mu}^h, \mathbf{g}\right) && \forall \boldsymbol{\mu}^h \in \mathscr{M}^h \end{aligned} \tag{9}$$

where $\mathbf{v}^h$ and $\boldsymbol{\mu}^h$ are the variations of the displacement and multiplier fields and $[\mathbf{u}^h, \boldsymbol{\lambda}^h]$ is the solution.

It is well-known [8, 33] that the finite element field of displacements and Lagrange multipliers must fulfill two conditions (*ElKer* and *InfSup*) to obtain an optimal convergence rate of the solution as the mesh is refined. The first of these, ellipticity of $a(\cdot, \cdot)$ in the kernel of $b(\cdot, \cdot)$, is easy to fulfill and ensures that there are enough multipliers to prevent rigid body motions. The second prevents too many multipliers being introduced and it is more difficult to fulfill in practice. There are examples in the bibliography that satisfy the *InfSup* condition, such as the 'vital vertex' method in 2D [9] and 3D [10] for linear discretizations. However, this method has the drawback of increasing the number of unknowns of the problem. In addition, the coefficient matrix of the system is indefinite, which can increase the computational cost as compared to semi-definite positive systems.

The following norms that will be used throughout the text can be defined: the L^2-norm, the energy norm and a mesh dependent norm whose approximation properties can be found in [34, 35]:

$$\begin{aligned}
\|\mathbf{u}^h\|^2_{L^2,\Omega} &= \int_{\Omega^h} \mathbf{u}^h \cdot \mathbf{u}^h \, d\Omega \\
\|\mathbf{u}^h\|^2_{E} &= \int_{\Omega^h} \boldsymbol{\sigma}(\mathbf{u}^h) : \boldsymbol{\epsilon}(\mathbf{u}^h) \, d\Omega \\
\|\mathbf{u}^h\|^2_{\mathscr{U}^h} &= \left|\mathbf{u}^h\right|^2_{H^1,\Omega} + \sum_e h_e^{-1} \|\mathbf{u}^h\|^2_{L^2,\Gamma^e_{\mathrm{D}}} \\
\|\boldsymbol{\lambda}^h\|^2_{\mathscr{M}^h} &= \sum_e h_e \|\boldsymbol{\lambda}^h\|^2_{L^2,\Gamma^e_{\mathrm{D}}}
\end{aligned} \tag{10}$$

Note that the L^2-norm can also be defined for the boundary $\|\cdot\|^2_{L^2,\Gamma}$ replacing the integration domain. In the last norms, the summation extends to all elements of the mesh that are intersected by Γ_{D} and h_e is the size of the element intersected by the contour.

3. Stabilized methods

In practice, the problem with the Lagrange multiplier formulation (Equation (9)) is that most natural choices of the Lagrange multiplier field do not fulfill the *InfSup* condition because they introduce too many constraints. The idea behind stabilized methods is to impose additional conditions on Lagrange multipliers without modifying the problem solution, at least at the limit when the element size approaches zero, in order to have more freedom to choose the Lagrange multiplier field.

3.1. Nitsche's method

Nitsche's method is one of the most widely used of the stabilization methods. It is related to the stabilized formulation proposed by Barbosa and Hughes [11, 12] to circumvent the Babuska-Brezzi condition. Its formulation is based on using the tractions on the boundary as a stabilization term of the multipliers. The original

formulation using Lagrange multipliers can be obtained from the following stabilized Lagrangian:

$$\mathscr{L}_N\left(\mathbf{v}^h, \boldsymbol{\mu}^h\right) = \mathscr{L}\left(\mathbf{v}^h, \boldsymbol{\mu}^h\right) - \frac{1}{2}\frac{h}{k}\int_{\Gamma_D} \left\|\boldsymbol{\mu}^h + \boldsymbol{\sigma}(\mathbf{v}^h)\mathbf{n}\right\|^2 d\Gamma \tag{11}$$

where $\mathscr{L}\left(\mathbf{v}, \boldsymbol{\mu}\right)$ was defined in Equation (7), h is the element size and k is a positive constant having the same units as the Young modulus. In order to use a dimensionless algorithmic constant we define $k = \kappa C_E$, using the constant defined in Equation (4) depending on the material properties.

The saddle point of the Lagrangian (11) is $[\mathbf{u}^h, \boldsymbol{\lambda}^h] \in \mathscr{U}^h \times \mathscr{M}^h$ such that:

$$\begin{aligned} &a\left(\mathbf{u}^h, \mathbf{v}^h\right) + b\left(\boldsymbol{\lambda}^h, \mathbf{v}^h\right) - \frac{h}{k}\int_{\Gamma_D}\left(\boldsymbol{\lambda}^h + \boldsymbol{\sigma}(\mathbf{u}^h)\mathbf{n}\right)\cdot\boldsymbol{\sigma}(\mathbf{v}^h)\mathbf{n}\, d\Gamma = c\left(\mathbf{v}^h\right) \\ &b\left(\boldsymbol{\mu}^h, \mathbf{u}^h\right) - \frac{h}{k}\int_{\Gamma_D}\boldsymbol{\mu}^h\cdot\left(\boldsymbol{\lambda}^h + \boldsymbol{\sigma}(\mathbf{u}^h)\mathbf{n}\right)\, d\Gamma = b\left(\boldsymbol{\mu}^h, \mathbf{g}\right) \end{aligned} \tag{12}$$

$\forall[\mathbf{v}^h, \boldsymbol{\mu}^h] \in \mathscr{U}^h \times \mathscr{M}^h$. Stenberg [35] shows that, for a suitable choice of the multiplier space in L^2, the Lagrange multiplier field can be eliminated element by element from Equation (12) to obtain the classical formulation of the Nitsche's method:

Find $\mathbf{u}^h \in \mathscr{U}^h$ such that:

$$\begin{aligned} a\left(\mathbf{u}^h, \mathbf{v}^h\right) - b\left(\boldsymbol{\sigma}(\mathbf{u}^h)\mathbf{n}, \mathbf{v}^h\right) - b\left(\boldsymbol{\sigma}(\mathbf{v}^h)\mathbf{n}, \mathbf{u}^h\right) + \frac{k}{h}\int_{\Gamma_D}\mathbf{u}^h\cdot\mathbf{v}^h\, d\Gamma = \\ c\left(\mathbf{v}^h\right) + \frac{k}{h}\int_{\Gamma_D}\mathbf{g}\cdot\mathbf{v}^h\, d\Gamma - b\left(\boldsymbol{\sigma}(\mathbf{v}^h)\mathbf{n}, \mathbf{g}\right) \qquad \forall\mathbf{v}^h \in \mathscr{U}^h \end{aligned} \tag{13}$$

Nitsche's method has been widely used in the context of immersed boundary mesh to solve interface problems (see for example [15, 16, 22, 23, 36]). However, the original Nitsche's method has some limitations when imposing Dirichlet boundary conditions, as has been pointed out in the bibliography [10, 17, 18, 19]. The optimal convergence rate of the finite element solution can only be achieved if the norm of the tractions on the contour can be bounded by the energy norm, i.e.

$$\left\|\boldsymbol{\sigma}(\mathbf{v}^h)\mathbf{n}\right\|_{L^2,\Gamma_D} \leq \frac{C_N}{h_e}\|\mathbf{v}^h\|_E \tag{14}$$

with a constant C_N independent of element size. In the case of immersed boundary meshes, in general, C_N cannot be bounded as the mesh is refined. To illustrate this problem, Figure 1 shows an element of a 2D mesh cut by the boundary of the problem domain. The shaded part indicates the internal area of the element Ω^e. If the boundary comes closer to the element edge as the mesh is refined, the size of the area Ω^e is reduced faster than the size of the boundary Γ_D^e, and the expression (14)

is fulfilled with unbounded values of C_N. High values of C_N increase the condition number of the system [19] and tend to overweight the boundary terms with respect to the domain energy, thus resulting in a finite element solution with a large error [18] (see numerical examples).

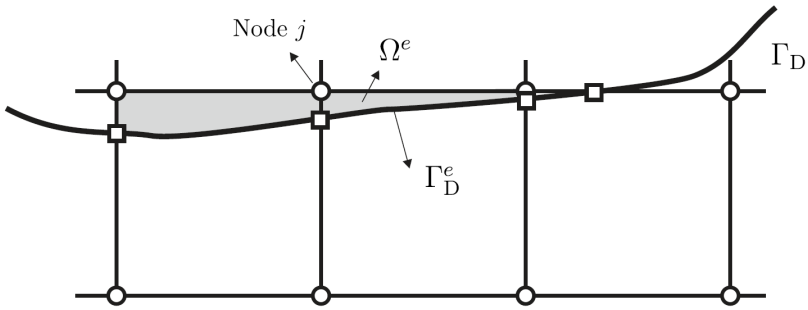


Figure 1: Cartesian element intersected by the geometry with small area.

Remark 1. *In practice, the number of elements with very small internal volume randomly changes between consecutive meshes or in different parts of the boundary, so the effect of this problem is limited. Furthermore, this problem can be minimized by the use of geometric tolerances that avoid the presence of elements with small internal volume, and acceptable results can be obtained from the engineering point of view [10].*

3.2. Proposed method

In this paper we propose the traction computed from a recovered stress field as the stabilization term. The method follows the ideas of extending the solution from the internal elements to the boundary elements [17, 24], but using the stress field instead of the displacement field. The smoothed stress field is obtained from the stresses calculated by the finite element method in the boundary elements and adjacent elements using the concept of the Superconvergence Patch Recovery [32] (see following section). The aim of this choice is to avoid the problems arising from the condition of Equation (14). As we shall see, this method allows the optimal convergence rate to be obtained for predefined bounded values of the penalty constant k, regardless of the boundary cutting pattern.

The proposed method is derived from the following Lagrangian:

$$\mathscr{L}_S\left(\mathbf{v}^h, \boldsymbol{\mu}^h\right) = \mathscr{L}\left(\mathbf{v}^h, \boldsymbol{\mu}^h\right) - \frac{1}{2}\frac{h}{k}\int_{\Gamma_{\mathrm{D}}} \left\|\boldsymbol{\mu}^h + \boldsymbol{T}(\hat{\mathbf{u}}^h)\right\|^2 d\Gamma \tag{15}$$

where $\boldsymbol{T}(\hat{\mathbf{u}}^h)$ is the smoothed traction that depends on the finite element solution computed from a previous iteration, $\hat{\mathbf{u}}^h$. Again the penalty constant can be defined as $k = \kappa C_E$.

The resulting saddle point problem reads as: Find $[\mathbf{u}^h, \boldsymbol{\lambda}^h] \in \mathscr{U}^h \times \mathscr{M}^h$ such that:

$$\begin{aligned} &a\left(\mathbf{u}^h, \mathbf{v}^h\right) + b\left(\boldsymbol{\lambda}^h, \mathbf{v}^h\right) = c\left(\mathbf{v}^h\right) \\ &b\left(\boldsymbol{\mu}^h, \mathbf{u}^h\right) - \frac{h}{k}\int_{\Gamma_D} \boldsymbol{\mu}^h \cdot \boldsymbol{\lambda}^h \, d\Gamma = b\left(\boldsymbol{\mu}^h, \mathbf{g}\right) + \frac{h}{k}\int_{\Gamma_D} \boldsymbol{\mu}^h \cdot \boldsymbol{T}(\hat{\mathbf{u}}^h) \, d\Gamma \end{aligned} \tag{16}$$

To obtain Equation (16) from the Lagrangian (15) we considered that $\boldsymbol{T}(\hat{\mathbf{u}}^h)$ is a predefined field, so that its variation is zero. As in Nitsche's method, the proposed formulation can be simplified by eliminating the Lagrange multipliers. Assuming that the multiplier field is in L^2 and following a method proposed in [35] we can solve the second equation in (16) for each element, to obtain the value of the multiplier $\boldsymbol{\lambda}_e^h$ as:

$$\boldsymbol{\lambda}_e^h = \frac{k}{h}\left(\mathbf{u}^h - \mathbf{g}\right) - \boldsymbol{T}(\hat{\mathbf{u}}^h) \tag{17}$$

Substituting (17) in the first equation of (16) we obtain a modified penalty method: Find the displacement field $\mathbf{u}^h \in \mathscr{U}^h$ such that

$$\begin{aligned} a\left(\mathbf{u}^h, \mathbf{v}^h\right) + \frac{k}{h}\int_{\Gamma_D} \mathbf{u}^h \cdot \mathbf{v}^h \, d\Gamma = \\ c\left(\mathbf{v}^h\right) + \frac{k}{h}\int_{\Gamma_D} \mathbf{g} \cdot \mathbf{v}^h + \int_{\Gamma_D} \boldsymbol{T}(\hat{\mathbf{u}}^h) \cdot \mathbf{v}^h \, d\Gamma \qquad \forall \mathbf{v}^h \in \mathscr{U}^h \end{aligned} \tag{18}$$

The second term on the left hand side of (18) is a penalty term with a constant k/h. The last term on the right hand side is the virtual work of reaction forces. It acts as a correction of the penalty term and is necessary for the finite element solution to converge to the exact solution of the problem as the mesh is refined.

As the traction field $\boldsymbol{T}$ depends on the finite element solution, an iterative procedure is proposed to solve the problem. In the first iteration, we solve the problem assuming that $\boldsymbol{T} = 0$. Then the smooth stress field is calculated. This stress field is used to update $\boldsymbol{T}$ in Equation (18) in order to solve the next iteration. This process runs until convergence is achieved.

The advantages of the proposed method are:

- As in Nitsche's method, the unknowns of the problem in the proposed formulation are the displacement degrees of freedom as the multipliers are condensed. Thus the size of the problem is not increased.
- Compared to Nitsche's method (Equation (13)), fewer integral terms are needed to compute the system. As we will see in the following section, the proposed method is stable and convergent in spite of the lacking of these terms.

- The method is stable for a mesh independent bounded value of the penalty constant κ. In the following section we obtain the value of this constant for 8-node linear and 20-node quadratic elements.
- The proposed method can be directly applied to solve problems with non-linear constitutive material behavior (see section 3.4).

The obvious drawback is that multiple iterations are needed to get the solution. However, this disadvantage can be minimized taking into account that the matrix to solve for each iteration is always the same for linear problems (since it is only mesh dependent). Therefore it is only necessary to factorize the matrix once and perform backward substitution every iteration.

3.3. Iterative Nitsche's method

The proposed method can also be used to define another formulation, by replacing the stabilizing smooth stress field $\boldsymbol{T}$ by the traction computed from the finite element solution of a previous iteration $\boldsymbol{\sigma}(\hat{\mathbf{u}}^h)\mathbf{n}$ instead of the recovered tractions. This formulation is also compared in the numerical results.

3.4. Plasticity

Although the equations and all theoretical analysis in this paper are done for the linear elastic case, the proposed method can be used to solve problems with elasto-plastic material behavior. In this case, Equation (18) is still valid if we replace the bilinear form $a\left(\mathbf{u}^h, \mathbf{v}^h\right)$ by the virtual work of internal forces. In the context of immersed boundaries, this term is computed in exactly the same way as is done for the standard finite element method, provided that the quadrature points are properly defined in the internal part of the elements.

The stabilizing terms depend on the displacement field and the stabilizing stress $\boldsymbol{T}$. As we shall see, the stress $\boldsymbol{T}$ is directly computed from the quadrature points of the domain (not the boundary) thus there is no need to change anything with respect to the linear elastic case. The method runs in two loops, one to update the stabilizing stress $\boldsymbol{T}$ and another, with $\boldsymbol{T}$ remaining constant, to solve the non-linear plasticity problem. In the numerical examples a plasticity problem is included to show the performance of the method.

4. Interpolation and numerical integration

In this work we use Cartesian meshes formed by 3D hexahedral elements whose contours are parallel to the Cartesian planes and consider linear 8-node $\mathcal{L}_8$ and quadratic 20-node $\mathcal{Q}_{20}$ hexahedral elements. For displacements $\mathbf{u}^h$ the usual finite element interpolation shape functions are defined using degree $p = 1$ for $\mathcal{L}_8$ elements and degree $p = 2$ for $\mathcal{Q}_{20}$ elements.

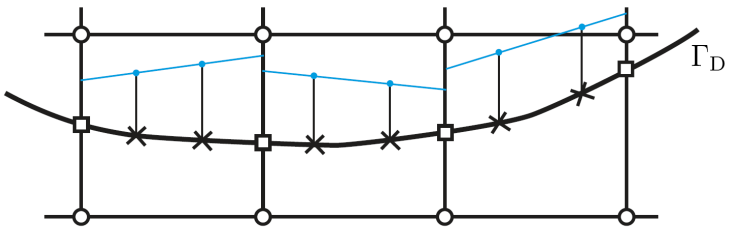


Figure 2: Examples of intersected linear elements in 2D. Segmentation of the boundary Γ_D based on the intersection of the boundary with the element edges (squares). The '×' symbols denote the quadrature points where the Lagrange multipliers are used to define a piecewise discontinuous linear interpolation.

The boundary integrals in the proposed formulation (Equation (18)) are numerically evaluated using Gaussian quadrature. This is equivalent to implicitly define a multiplier unknown at each quadrature point in the saddle point formulation (Equation (16)) and to eliminate it from the system of equations at element level. This implicit definition was proposed in [37] and used in [26]. A schematic representation of the interpolation is depicted in Figure 2, which shows three 2D elements of the boundary. The quadrature points at the boundary are used to define a polynomial in each boundary segment. A piecewise discontinuous interpolation is then obtained. In practice there is no need to explicitly define this polynomial, because we only need to evaluate it by numerical integration at the quadrature points, where the value is precisely that of the Lagrange multiplier.

In 2D problems, such as the one depicted in Figure 2, n_g is the number of quadrature points at each element. To exactly integrate the product of displacements (degree p) and multipliers (degree $n_g - 1$), if Gaussian quadrature is used, we need to fulfill that $2 \cdot n_g - 1 > p + n_g - 1$, so $n_g > p$. In 3D problems the part of the real boundary in each element is triangulated, and a piecewise discontinuous polynomial interpolation is defined in each triangle (see Figure 3). We use $n_g = 7$ quadrature points,

which means that a polynomial of degree 5 can be exactly integrated. The implicit interpolation of the multiplier has a complete degree $q = 2$.

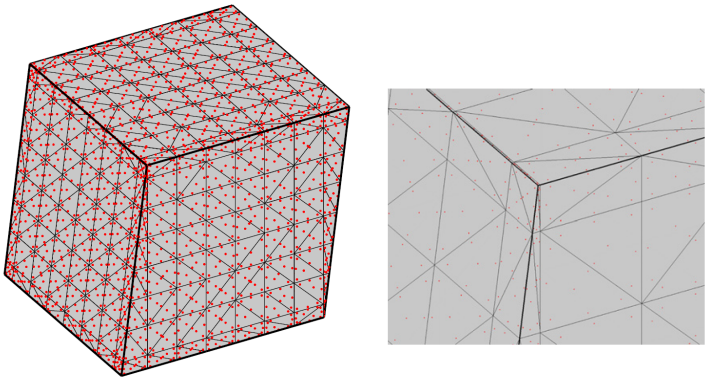


Figure 3: Triangulation of the boundary used to perform the numerical integration. The dots are located at the quadrature points (7 each facet). Complete domain and zoom at a vertex.

The best approximation error of the finite element spaces measured in mesh-dependent norms is discussed in [34, 35]. If the exact solution is smooth enough, namely, if $\mathbf{u} \in H^{p+1}(\Omega)$ and $\boldsymbol{\lambda} \in H^{q+1}(\Gamma)$ and the degree of displacement and multiplier interpolation are at least p and q respectively, it can be proved that the chosen finite element fields fulfill

$$\begin{aligned} \inf_{\mathbf{v}^h \in \mathscr{U}^h} \|\mathbf{u} - \mathbf{v}^h\|_{\mathscr{U}^h} &\leq C h^p \|\mathbf{u}\|_{p+1} \\ \inf_{\boldsymbol{\mu}^h \in \mathscr{M}^h} \|\boldsymbol{\lambda} - \boldsymbol{\mu}^h\|_{\mathscr{M}^h} &\leq C h^{q+3/2} \|\boldsymbol{\lambda}\|_{q+1,\Gamma} \end{aligned} \tag{19}$$

5. Recovered stress field

As mentioned above, the term $\boldsymbol{T}$ used to stabilize the Lagrange multipliers is a recovered stress field obtained from the finite element solution. The construction of the smoothing field is based on the SPR technique [32]. The computation is performed as follows:

1. For each vertex node i of the boundary elements (those intersected by the geometry), including the vertex nodes located outside the domain, we construct a patch with all the elements that contain this node. For illustration purposes, Figure 4 shows part of a 2D mesh close to a boundary of the domain. The elements of the patch are used to calculate the recovered stress at node i. In this case, two elements of the patch are internal and two are divided by the boundary. The shaded region is the part of the domain corresponding to the patch and is denoted as Ω_{patch}.

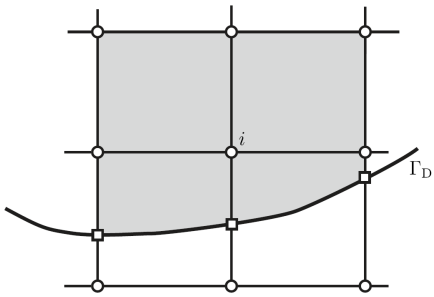


Figure 4: Elements used to compute the smooth stress field for node i.

2. Each component of the recovered stress field for node i is defined as a complete polynomial of degree p (being p the degree of the displacement interpolation) $\boldsymbol{S}_{ic} = \mathbf{x} \cdot \mathbf{w}_{ic}$, where $\mathbf{x} = \{1, x, y, z, xy, \ldots\}$ is the polynomial basis and $\mathbf{w}_{ic}$ is the coefficient vector. Subindex c is introduced to indicate the stress component $c = xx, yy, zz, xy, yz, zx$. These coefficients are calculated by minimizing the L^2-norm of the difference between the polynomial and the finite element solution in the patch, i.e. solving the following minimization problem:

$$\begin{aligned} \mathbf{w}_{ic} = \arg\min_{\mathbf{w}} \int_{\Omega_{patch}} (\boldsymbol{\sigma}_c^h - \boldsymbol{S}_{ic}) \cdot (\boldsymbol{\sigma}_c^h - \boldsymbol{S}_{ic})\, d\Omega \approx \\ \arg\min_{\mathbf{w}} \sum_{\forall g \in \Omega_{patch}} H_g J_g (\boldsymbol{\sigma}_g^h - \boldsymbol{S}_g) \cdot (\boldsymbol{\sigma}_g^h - \boldsymbol{S}_g) \end{aligned} \tag{20}$$

The integral is evaluated numerically with the same quadrature points used to calculate the stiffness matrix. The subscript g indicates that the variable is calculated at the quadrature point. H_g is the weight and J_g the Jacobian of the element subtriangulation performed to compute the volume integrals.

3. The recovered stress field in the domain $\boldsymbol{S}_c$ is an interpolation of the nodal polynomials obtained in step 2, $\boldsymbol{S}_c = \sum_{\forall i} N_i \boldsymbol{S}_{ic}$, by using the linear shape functions of the vertex nodes, N_i.

4. The stabilizing stress is the traction computed from the recovered stress tensor: $\boldsymbol{T} = \boldsymbol{S}\mathbf{n}$, where $\mathbf{n}$ is the normal vector and $\boldsymbol{S}$ is the tensor whose components are the $\boldsymbol{S}_c$ polynomials defined above .

The recovered stress polynomial $\boldsymbol{S}_i$ at each patch depends on the solution evaluated at both, boundary elements and internal elements. The integral used to calculate the field $\boldsymbol{S}_i$ gives weight to the integration points as a function of the volume associated to each of them. Therefore, the smaller the volume of the cut elements, the lower the weight and the smaller the influence on the smooth stress field.

The above definition of the recovered field fulfills three properties that will be used in the paper:

Property 2. *The L^2-norm of the recovered stress field, can be bounded by the L^2-norm of the finite element stress field in the solid domain Ω:*

$$\|\boldsymbol{S}\|^2_{L^2,\Omega} \leq C_r \left\|\boldsymbol{\sigma}(\mathbf{u}^h)\right\|^2_{L^2,\Omega} \tag{21}$$

with $C_r = 8$. Assuming a smooth enough exact solution $C_r \to 1$ as the mesh is refined.

Proof: Taking the derivative in Equation (20) we obtain the following system to evaluate the coefficient vector $\mathbf{w}_i$

$$\left(\int_{\Omega_{patch}} \mathbf{x}^T\, \mathbf{x}\, d\Omega\right) \mathbf{w}_{ic} = \int_{\Omega_{patch}} \mathbf{x}^T\, \boldsymbol{\sigma}^h_c\, d\Omega$$

Multiplying the previous expression by the solution $\mathbf{w}^T_{ic}$ we have:

$$\|\boldsymbol{S}_{ic}\|^2_{L^2,\Omega_{patch}} = \int_{\Omega_{patch}} \boldsymbol{S}_{ic} \cdot \boldsymbol{\sigma}^h_c\, d\Omega$$

Taking into account the last expression and Equation (20) we obtain:

$$\int_{\Omega_{patch}} (\boldsymbol{\sigma}^h_c - \boldsymbol{S}_{ic}) \cdot (\boldsymbol{\sigma}^h_c - \boldsymbol{S}_{ic})\, d\Omega \geq 0$$

$$\left\|\boldsymbol{\sigma}^h_c\right\|^2_{L^2,\Omega_{patch}} + \|\boldsymbol{S}_{ic}\|^2_{L^2,\Omega_{patch}} - 2\int_{\Omega_{patch}} \boldsymbol{\sigma}^h_c \cdot \boldsymbol{S}_{ic}\, d\Omega =$$

$$\left\|\boldsymbol{\sigma}^h_c\right\|^2_{L^2,\Omega_{patch}} - \|\boldsymbol{S}_{ic}\|^2_{L^2,\Omega_{patch}} \geq 0$$

As each nodal component $\boldsymbol{S}_{ic}$ can be bounded by the stress norm in every patch, the norm of the tensor is also bounded. On the other hand, the recovered stress field (step

3) is the interpolation of the nodal polynomials using the linear shape functions, whose value is between 0 and 1. Then, using the Cauchy-Schwartz inequality and taking into account that the number of nodes per elements is eight and that the patches contain at most eight elements, we obtain that $C_r = 8$ from:

$$\|\boldsymbol{S}\|^2_{L^2,\Omega} = \sum_{\forall e} \int_{\Omega^e} \left(\sum_{i=1}^{8} N_i \boldsymbol{S}_i \right)^2 d\Omega \leq \sum_{\forall e} \int_{\Omega^e} \sum_{i=1}^{8} (N_i)^2 \sum_{i=1}^{8} (\boldsymbol{S}_i)^2 \, d\Omega \leq$$

$$\sum_{\forall e} \int_{\Omega^e} \sum_{i=1}^{8} (\boldsymbol{S}_i)^2 \, d\Omega = \sum_{\forall patch} \|\boldsymbol{S}_i\|^2_{L^2,\Omega_{patch}} \leq \sum_{\forall patch} \left\|\boldsymbol{\sigma}^h\right\|^2_{L^2,\Omega_{patch}} = 8 \left\|\boldsymbol{\sigma}(\mathbf{u}^h)\right\|^2_{L^2,\Omega}$$

The value of $C_r = 8$ could be improved by taking into account the definition of the linear shape functions. Considering that the exact solution is smooth enough, we can assume that the recovered stress field tends to be uniform in each patch (each $\boldsymbol{S}_i$ is constant) as the element size becomes smaller. Under that assumption it follows that:

$$\|\boldsymbol{S}\|^2_{L^2,\Omega} = \sum_{\forall e} \int_{\Omega^e} \left(\sum_{i=1}^{8} N_i \boldsymbol{S}_i \right)^2 d\Omega \leq \frac{1}{8} \sum_{\forall e} \int_{\Omega^e} \sum_{i=1}^{8} (\boldsymbol{S}_i)^2 \, d\Omega = \left\|\boldsymbol{\sigma}(\mathbf{u}^h)\right\|^2_{L^2,\Omega}$$

and the constant $C_r = 1$.
□

Property 3. *Assuming that every boundary element is connected to at least one internal element, the L^2-norm of $\boldsymbol{T}$ in the boundary can be bounded by the energy norm of the finite element solution with a constant C_p independent of the mesh*

$$\|\boldsymbol{T}\|^2_{L^2,\Gamma_D} \leq \frac{C_E \, C_r \, C_p}{h} \left\|\mathbf{u}^h\right\|^2_E \tag{22}$$

where h is the uniform element size, C_E is a material dependent constant defined in (4) and C_r is defined in property 2. The value of the constant is $C_p = 13$ for $\mathcal{L}_8$ elements and $C_p = 21$ for $\mathcal{Q}_{20}$ elements.

Proof: First we want to bound the L^2-norm of $\boldsymbol{S}$ on the boundary with its norm in the domain. For a given boundary element (e), there exists at least one internal element (ie) that shares an element face. We assume that the worst case to bound Equation (22) occurs when the boundary of the domain practically coincides with an element face. This is schematically depicted in Figure 1 in 2D. Then we have to find the best value of C_p such that the following inequality holds:

$$\|\boldsymbol{S}\|^2_{L^2,\Gamma^e} = \int_{\Gamma^{ie}} \boldsymbol{S}^2 \, d\Gamma \leq h \, C_p \, \|\boldsymbol{S}\|^2_{L^2,\Omega^{ie}} = h \, C_p \int_{\Omega^{ie}} \boldsymbol{S}^2 \, d\Omega.$$

$\boldsymbol{S}$ is the product of two polynomials: the smooth field of degree p and the linear shape functions having terms of at most degree 1 in each direction. Thus $\boldsymbol{S}^2$ has terms of

degree $2(p+1)$. The above integrals can be evaluated in the reference $[-1,1]$ element:

$$h^2 \int_{\square,\Gamma} \boldsymbol{S}^2 \, d\Gamma \leq C_p \, h^3 \int_{\square,\Omega} \boldsymbol{S}^2 \, d\Omega$$

where the symbol $\square,\Gamma$ denotes the reference element face and $\square,\Omega$ its volume. We can use the Newton-Cotes quadrature with positive weights to exactly evaluate the above integrals. The volume integral can be bounded as

$$\int_{\square,\Omega} \boldsymbol{S}^2 \, d\Omega = H_1 \underbrace{\sum_{j=1}^{n_q} \sum_{k=1}^{n_q} H_j H_k \boldsymbol{S}^2}_{\int_{\square,\Gamma} \boldsymbol{S}^2 \, d\Gamma} + \underbrace{\sum_{i=2}^{n_q} H_i \sum_{j=1}^{n_q} \sum_{k=1}^{n_q} H_j H_k \boldsymbol{S}^2}_{\text{positive}} \leq H_1 \int_{\square,\Gamma} \boldsymbol{S}^2 \, d\Gamma$$

where H_1 is the weight of the first quadrature point and n_q the number of quadrature points. For linear elements, $\boldsymbol{T}^2$ is a polynomial of degree 4 and it results that $n_q = 5$ and $C_p = 13$. For quadratic elements the degree of the polynomial is 6, $n_q = 7$ and $C_p = 21$.

Now, it holds that $\|\boldsymbol{T}\|^2_{L^2,\Gamma^e} \leq \|\boldsymbol{S}\|^2_{L^2,\Gamma^e}$. Adding the contribution of all boundary elements, using Property 2 (Equation (21)) and taking into account the relationship between the L^2 norm of the stresses and the energy norm (Equation (4)) the result follows.

□

Property 4. *Let* $\mathbf{u}$ *be the exact solution of the problem. If* $\mathbf{u}$ *is assumed to be regular enough, the recovered traction evaluated for the exact solution* $\boldsymbol{T}(\mathbf{u})$ *fulfills the following property:*

$$\|\boldsymbol{\sigma}(\mathbf{u})\boldsymbol{n} - \boldsymbol{T}(\mathbf{u})\|^2_{L^2,\Gamma_D} \approx \mathcal{O}(h^{p+1}) \tag{23}$$

where h *is the element size.*

Proof: Let $T(\mathbf{u})$ be the boundary tractions evaluated from (20) by replacing $\boldsymbol{\sigma}^h$ by the exact stress. Therefore, $T(\mathbf{u})$ is a polynomial approximation to the exact traction in each patch. The result follows assuming that $\mathbf{u}$ can be expanded in its Taylor series.

□

Remark 2. *The SPR technique can be modified to fulfill certain equations, such as equilibrium of stresses, compatibility equations, imposed boundary conditions, etc. This modification improves the approximation of the recovered stress field [6, 38].*

Remark 3. *Although we have chosen the SPR technique, any recovered stress field that satisfies Properties 2, 3 and 4 could be used as the stabilizing term and the results of the following section would hold.*

6. Convergence of the finite element solution

In this section the convergence of the finite element solution is analyzed. We proceed in four steps: First we show that the bilinear form defining the problem is coercive. Then we will analyze the conditions under which the iterative method converges. Finally we will show the stability and convergence to the exact solution of the problem.

6.1. Coercivity

As pointed out above, problem (18) is iteratively solved with an initial value of the stabilization field $\boldsymbol{T} = 0$. The system can be expressed in matrix form as

$$\mathbf{A}\,\mathbf{d}^i = \mathbf{c} + \mathbf{B}\,\mathbf{d}^{i-1} \tag{24}$$

where $\mathbf{d}^i$ is the vector of nodal displacements at iteration i, $\mathbf{A}$ is the matrix on the left side of the Equation (18), $\mathbf{B}$ is the matrix obtained from the last term on the right side of (18) which depends on the method used to calculate the recovered stress field, and $\mathbf{c}$ is the vector derived from the other terms of the right hand side in (18).

In order to check the convergence of the iterative method, we define the residual of the equation as:

$$\mathbf{r}^i = \mathbf{A}\,\mathbf{d}^i - \mathbf{B}\,\mathbf{d}^{i-1} - \mathbf{c} \tag{25}$$

Convergence is achieved when the norm of the residual $\|\mathbf{r}^i\|$ as well as the norm of the difference between two consecutive solutions $\|\mathbf{d}^i - \mathbf{d}^{i-1}\|$ are lower than the given tolerances.

Remark 4. *Problem* (24) *can be solved without explicitly defining matrix* $\boldsymbol{B}$ *by directly computing the product* $\boldsymbol{B}\,\boldsymbol{d}^{i-1}$*, which is easily computed as a surface integral. This term corresponds to the equivalent nodal forces imposed by the stabilization traction.*

The bilinear form defined in Equation (18) from which the matrix $\mathbf{A}$ is obtained is:

$$a\left(\mathbf{u}^h, \mathbf{v}^h\right) + \frac{h}{k}\int_{\Gamma_\mathrm{D}} \mathbf{u}^h \cdot \mathbf{v}^h \, d\Gamma$$

For any $k > 0$ the bilinear form is coercive since $a(\cdot,\cdot)$ is symmetric and positive semi-definite, with $a(\mathbf{v},\mathbf{v}) = 0$ only for rigid-body motions. The penalty term ensures that rigid-body motions are not allowed. As a consequence, matrix $\mathbf{A}$ is invertible.

6.2. Convergence of the iterative method

The procedure set out to solve the problem can be considered as a Richardson's iterative method of solving linear systems of equations. This method is known to converge [39] if the spectral radius of the matrix $\mathbf{A}^{-1}\mathbf{B}$ is lower than 1. We obtain the following result:

Proposition 1. *The iterative procedure defined in* (24) *converges for a large enough but bounded value of the penalty constant* $k > C_E C_p C_r/4$ *(or* $\kappa > C_p C_r/4$*).*

Proof: The spectral radius of the matrix $\mathbf{A}^{-1}\mathbf{B}$ is defined as the maximum of its eigenvalue modulus. Any eigenvalue λ of this matrix fulfills

$$\mathbf{A}^{-1}\mathbf{B}\,\mathbf{d} = \lambda \mathbf{d} \quad \rightarrow \quad \mathbf{B}\,\mathbf{d} = \lambda \mathbf{A}\,\mathbf{d}$$

Premultiplying by $\mathbf{d}^T$ on both sides of the equation it follows

$$\mathbf{d}^T\mathbf{B}\,\mathbf{d} = \lambda \mathbf{d}^T\mathbf{A}\,\mathbf{d}$$

To prove that the modulus of λ is less than 1, we can see that the left side of the equation corresponds to the stabilization term for a given $\mathbf{v}^h$, so that

$$\mathbf{d}^T\mathbf{B}\,\mathbf{d} = \int_{\Gamma_\mathrm{D}} \mathbf{v}^h \cdot \boldsymbol{T}(\mathbf{v}^h)\, d\Gamma \tag{26}$$

Analogously, we can write:

$$\mathbf{d}^T\mathbf{A}\,\mathbf{d} = a\left(\mathbf{v}^h, \mathbf{v}^h\right) + \frac{k}{h}\int_{\Gamma_\mathrm{D}} \mathbf{v}^h \cdot \mathbf{v}^h\, d\Gamma \tag{27}$$

Applying the Cauchy-Schwarz inequality to Equation (26), using Equation (22) and considering that for two positive numbers x and y it holds that $2xy \leq x^2 + y^2$, we obtain:

$$\begin{aligned} \lambda \mathbf{d}^T\mathbf{A}\mathbf{d} = \mathbf{d}^T\mathbf{B}\mathbf{d} &\leq \|\mathbf{v}^h\|_{L^2,\Gamma_\mathrm{D}} \|\boldsymbol{T}\|_{L^2,\Gamma_\mathrm{D}} \leq \|\mathbf{v}^h\|_{L^2,\Gamma_\mathrm{D}} \left(\frac{C_E\, C_p\, C_r}{h}\right)^{1/2} \left\|\mathbf{v}^h\right\|_E \\ &\leq \left\|\mathbf{v}^h\right\|_E^2 + \frac{C_E\, C_p\, C_r}{4h} \|\mathbf{v}^h\|_{L^2,\Gamma_\mathrm{D}}^2 \end{aligned} \tag{28}$$

Comparing (27) with (28), it follows that if $k > \dfrac{C_E\, C_p\, C_r}{4}$, the modulus of any eigenvalue λ must be less than 1.

□

6.3. Stability of the formulation

We define the following bilinear form associated to our problem (18):

$$Q(\mathbf{w}^h, \mathbf{v}^h) = a\left(\mathbf{w}^h, \mathbf{v}^h\right) + \frac{k}{h}\int_{\Gamma_D} \mathbf{w}^h \cdot \mathbf{v}^h \, d\Gamma - \int_{\Gamma_D} \boldsymbol{T}(\mathbf{w}^h) \cdot \mathbf{v}^h \, d\Gamma \tag{29}$$

Here we prove the weak coercivity of the functional Q that will be used to demonstrate the optimal convergence of the proposed method. Considering first the last term of Q, we apply the Cauchy-Schwartz inequality and Equation (22) to obtain:

$$\begin{aligned} &-\left|\int_{\Gamma_D} \boldsymbol{T}(\mathbf{v}^h) \cdot \mathbf{v}^h \, d\Gamma\right| \geq -\|\mathbf{v}^h\|_{L^2,\Gamma_D}\|\boldsymbol{T}(\mathbf{v}^h)\|_{L^2,\Gamma_D} \geq \\ &-\|\mathbf{v}^h\|_{L^2,\Gamma_D}\left(\frac{C_E\, C_p\, C_r}{h_e}\right)^{1/2} \left\|\mathbf{v}^h\right\|_E \geq -\frac{1}{2}\left\|\mathbf{v}^h\right\|_E^2 - \frac{C_E\, C_p\, C_r}{2h_e}\|\mathbf{v}^h\|_{L^2,\Gamma_D}^2 \end{aligned} \tag{30}$$

Proposition 2. *The bilinear form Q is weakly coercive if the penalty parameter is chosen as $k > C_E\, C_p\, C_r$, that is:*

$$\sup_{\mathbf{v}^h \in \mathscr{U}^h} \frac{Q(\mathbf{w}^h, \mathbf{v}^h)}{\|\mathbf{v}^h\|_{\mathscr{U}^h}} \geq \beta \|\mathbf{w}^h\|_{\mathscr{U}^h} \qquad \forall \mathbf{w}^h \in \mathscr{U}^h \tag{31}$$

with $\beta = \frac{1}{2}\left(k - C_E\, C_p\, C_r\right)$.

Proof: It suffices to show that the inequality holds for a certain value of $\mathbf{v}^h = \mathbf{w}^h$. Using (30) and the definition of Q (Equation (36)), and considering the mesh dependent norm defined in (10), we have:

$$\begin{aligned} Q(\mathbf{v}^h, \mathbf{v}^h) = &\, a\left(\mathbf{v}^h, \mathbf{v}^h\right) + \frac{k}{h}\int_{\Gamma_D} \mathbf{v}^h \cdot \mathbf{v}^h \, d\Gamma - \int_{\Gamma_D} \boldsymbol{T}(\mathbf{v}^h) \cdot \mathbf{v}^h \, d\Gamma \geq \\ &\, a\left(\mathbf{v}^h, \mathbf{v}^h\right) + \frac{k}{h}\int_{\Gamma_D} \mathbf{v}^h \cdot \mathbf{v}^h \, d\Gamma - \frac{1}{2}\left\|\mathbf{v}^h\right\|_E^2 - \frac{C_E\, C_p\, C_r}{2h_e}\|\mathbf{v}^h\|_{L^2,\Gamma_D}^2 \geq \\ &\, \frac{1}{2}\left(k - C_E\, C_p\, C_r\right)\|\mathbf{v}^h\|_{\mathscr{U}^h}^2 \end{aligned} \tag{32}$$

□

6.4. Optimal convergence

Let $[\mathbf{u}, \boldsymbol{\lambda}]$ be the exact solution of the problem (7) and $[\mathbf{u}^h, \boldsymbol{\lambda}^h]$ the solution of the discretized problem (16). Taking variations in (7), a system is obtained that must be fulfilled $\forall [\mathbf{v}, \boldsymbol{\mu}] \in \mathscr{U} \times \mathscr{M}$. In particular, it will also be true for $\mathbf{v}^h \in \mathscr{U}^h \subset \mathscr{U}$ and $\boldsymbol{\mu}^h \in \mathscr{M}^h \subset \mathscr{M}$, and we can write:

$$\begin{aligned} a\left(\mathbf{u}, \mathbf{v}^h\right) + b\left(\boldsymbol{\lambda}, \mathbf{v}^h\right) &= c\left(\mathbf{v}^h\right) \\ b\left(\boldsymbol{\mu}^h, \mathbf{u}\right) &= b\left(\boldsymbol{\mu}^h, \mathbf{g}\right) \end{aligned} \tag{33}$$

Adding the stabilization term $\frac{h}{k}\int_{\Gamma_\mathrm{D}} \boldsymbol{\mu}^h \cdot (\boldsymbol{\sigma}(\mathbf{u})\mathbf{n} + \boldsymbol{\lambda})\ d\Gamma$, (which is zero since for the exact solution $\boldsymbol{\lambda} = -\boldsymbol{\sigma}(\mathbf{u})\mathbf{n}$), to the second equation and subtracting (16) it follows that:

$$\begin{aligned} &a\left(\mathbf{u} - \mathbf{u}^h, \mathbf{v}^h\right) + b\left(\boldsymbol{\lambda} - \boldsymbol{\lambda}^h, \mathbf{v}^h\right) = 0 \\ &b\left(\boldsymbol{\mu}^h, \mathbf{u} - \mathbf{u}^h\right) - \frac{h}{k}\int_{\Gamma_\mathrm{D}} \boldsymbol{\mu}^h \cdot \left(\boldsymbol{\lambda} - \boldsymbol{\lambda}^h\right)\ d\Gamma = \frac{h}{k}\int_{\Gamma_\mathrm{D}} \boldsymbol{\mu}^h \cdot \left(\boldsymbol{\sigma}(\mathbf{u})\mathbf{n} - \boldsymbol{T}(\mathbf{u}^h)\right)\ d\Gamma \end{aligned} \tag{34}$$

In (16) the solution upon completion of the iterative procedure has been considered, so that the traction $\boldsymbol{T}$ is calculated for the same $\mathbf{u}^h$ as the energy.

Operating as in the previous section, the multipliers can be eliminated to obtain the following orthogonality property of our formulation:

$$a\left(\mathbf{u} - \mathbf{u}^h, \mathbf{v}^h\right) + \frac{k}{h}\int_{\Gamma_\mathrm{D}} \left(\mathbf{u} - \mathbf{u}^h\right) \cdot \mathbf{v}^h\ d\Gamma - \int_{\Gamma_\mathrm{D}} \left(\boldsymbol{\sigma}(\mathbf{u})\mathbf{n} - \boldsymbol{T}(\mathbf{u}^h)\right) \cdot \mathbf{v}^h\ d\Gamma = 0 \tag{35}$$

Adding and subtracting the polynomial approximation of the traction in the Dirichlet boundary $\boldsymbol{T}(\mathbf{u})$ into the last integral of Equation 35 we obtain:

$$Q(\mathbf{u} - \mathbf{u}^h, \mathbf{v}^h) - \int_{\Gamma_\mathrm{D}} (\boldsymbol{\sigma}(\mathbf{u})\mathbf{n} - \boldsymbol{T}(\mathbf{u})) \cdot \mathbf{v}^h\ d\Gamma = 0 \tag{36}$$

Taking into account the stability of functional Q (Equation (31)) and using Equation (36) it follows that for any $\mathbf{w}^h \in \mathscr{U}^h$:

$$\|\mathbf{u}^h - \mathbf{w}^h\|_{\mathscr{U}^h} \leq \frac{1}{\beta}\frac{Q(\mathbf{u}^h - \mathbf{w}^h, \mathbf{v}^h)}{\|\mathbf{v}^h\|_{\mathscr{U}^h}} = \frac{1}{\beta}\frac{Q(\mathbf{u}^h - \mathbf{u}, \mathbf{v}^h) + Q(\mathbf{u} - \mathbf{w}^h, \mathbf{v}^h)}{\|\mathbf{v}^h\|_{\mathscr{U}^h}} \leq$$

$$\frac{1}{\beta}\frac{\left|\int_{\Gamma_\mathrm{D}} (\boldsymbol{\sigma}(\mathbf{u})\mathbf{n} - \boldsymbol{T}(\mathbf{u})) \cdot \mathbf{v}^h\ d\Gamma\right| + Q(\mathbf{u} - \mathbf{w}^h, \mathbf{v}^h)}{\|\mathbf{v}^h\|_{\mathscr{U}^h}}$$

Now we use the Cauchy-Schwartz inequality and the continuity of functional Q with constant C to obtain:

$$\begin{aligned}\|\mathbf{u}^h - \mathbf{w}^h\|_{\mathscr{U}^h} \leq & \frac{C}{\beta}\|\mathbf{u} - \mathbf{w}^h\|_{\mathscr{U}^h} + \frac{1}{\beta}\left\|\boldsymbol{\sigma}(\mathbf{u})\mathbf{n} - \boldsymbol{T}(\mathbf{u})\right\|_{L^2}\frac{\|\mathbf{v}^h\|_{L^2}}{\|\mathbf{v}^h\|_{\mathscr{U}^h}} \leq \\ & \frac{C}{\beta}\|\mathbf{u} - \mathbf{w}^h\|_{\mathscr{U}^h} + \frac{h^{1/2}}{\beta}\left\|\boldsymbol{\sigma}(\mathbf{u})\mathbf{n} - \boldsymbol{T}(\mathbf{u})\right\|_{L^2}\end{aligned} \tag{37}$$

Finally we obtain the optimal convergence result:

Proposition 3. *Let* $\mathbf{u}^h$ *be the solution of problem* (18) *with* $k > C_E C_p C_r$ *(*$\kappa > C_p C_r$*) and* $\mathbf{u}$ *the exact solution. Then it holds that*

$$\|\mathbf{u} - \mathbf{u}^h\|_{\mathscr{U}^h} \leq \mathcal{O}(h^p) \tag{38}$$

p being the degree of the polynomial interpolation.

Proof: Let $\mathbf{w}^h$ be any function in the finite element space $\mathscr{U}^h$. We can write

$$\|\mathbf{u} - \mathbf{u}^h\|_{\mathscr{U}^h} = \|\mathbf{u} - \mathbf{w}^h - \mathbf{u}^h + \mathbf{w}^h\|_{\mathscr{U}^h} \leq \|\mathbf{u} - \mathbf{w}^h\|_{\mathscr{U}^h} + \|\mathbf{u}^h - \mathbf{w}^h\|_{\mathscr{U}^h}$$

Now we use Equation (37), the best approximation property of the finite element space (Equation 19) and property 4 of the smooth stress field $\boldsymbol{T}$ (Equation (23))

$$\|\mathbf{u} - \mathbf{u}^h\|_{\mathscr{U}^h} \leq \|\mathbf{u} - \mathbf{w}^h\|_{\mathscr{U}^h} + \frac{h^{1/2}}{\beta}\left\|\boldsymbol{\sigma}(\mathbf{u})\mathbf{n} - \boldsymbol{T}(\mathbf{u})\right\|_{L^2} \leq \mathcal{O}(h^p) + \mathcal{O}(h^{p+3/2})$$

□

7. Numerical examples

In this section we illustrate the capabilities of the proposed formulation and explore the limitations of the methods. Three numerical examples with exact solution were solved and used to check the convergence of the Richardson iterations, the convergence rate of the finite element solution as the mesh is refined and the effect of the stability constant k on convergence. They were also used to compare the proposed method with Nitsche's method showing that, in general, both methods have similar behavior. The third example highlights the robustness of the proposed method in cases where Nitsche's method fails to provide the desired accuracy, *i.e.*, when the boundary of the domain comes close to the element faces. An additional example shows the behavior of the proposed method, without any modifications, when elasto-plastic behavior of

the material is considered. This example is particularly interesting as, to the authors' knowledge, Nitsche's method has not been used to solve these type of problem because the formulation of the method required for plasticity has not been derived. Linear ($\mathcal{L}_8$) and quadratic ($\mathcal{Q}_{20}$) elements are used in the examples.

7.1. Example 1: Tilted hexahedron

In the first example we consider an infinite domain subjected to 4^{th} order polynomial displacements and plain strain conditions. The exact solution reads as:

$$
\begin{aligned}
& E = 1000,\ \nu = 0.3 \\
u_x &= -\frac{25}{192} + \frac{75}{64}x^2 - \frac{25}{24}x^4 - \frac{25}{4}y + \\
&\qquad \frac{25}{4}x^2y - \frac{25}{8}y^2 + \frac{25}{8}x^2y^2 \\
u_y &= \frac{20}{3}x + \frac{65}{12}x^3 + \frac{65}{12}x^3y - 10xy^2 - \frac{10}{3}xy^3 \\
u_z &= 0 \\
t_{vx} &= \frac{-E}{1+\nu}(1+y) \\
t_{vy} &= \frac{-E}{1+\nu}(1-x) \\
t_{vz} &= 0
\end{aligned}
\qquad
\begin{aligned}
\sigma_{xx} &= \frac{5}{2}x - 3x^3 + 8xy + 4xy^2 \\
\sigma_{yy} &= \frac{5}{8}x + \frac{14}{3}x^3 - 18xy - 9xy^2 \\
\sigma_{xy} &= \frac{1}{6} + 9x^2 - \frac{5}{2}y + 9x^2y - 4y^2 - \frac{4}{3}y^3 \\
\sigma_{zz} &= -0.3\,(\sigma_{xx} + \sigma_{yy}) \\
\sigma_{yz} &= \sigma_{zx} = 0
\end{aligned}
$$

A finite portion of the infinite domain defined by a tilted hexahedron was considered for the analysis. The known values of the displacements were imposed as Dirichlet boundary conditions on the entire external surfaces of the domain. Figure 5 shows the exact geometry of the problem embedded in the Level 3 Cartesian grid. The same figure also shows the surface triangulation used to evaluate numerically the contour integrals.

The problem was solved using a sequence of Cartesian meshes obtained by element subdivision starting from a Level 2 mesh having 64 hexahedral elements (Level 0 has a single element and Level 1 has eight elements).

In order to test the influence of the stability constant on the convergence of the proposed iterative method and on the discretization error, the same mesh was analyzed with κ ranging from 0.04 to $4 \cdot 10^6$. We also considered Nitsche's method in the convergence analysis for comparison purposes.

The convergence of the normalized residual (Equation (25)) is plotted in Figure 6 for different values of κ using linear and quadratic elements of the Level 2 Cartesian mesh. As predicted, small values of the penalty constant can cause very slow convergence or even loss of convergence in the iterative method. The greater the value of κ the faster the convergence, but very large values of the penalty constant can increase the discretization error of the finite element solution. To illustrate this the energy

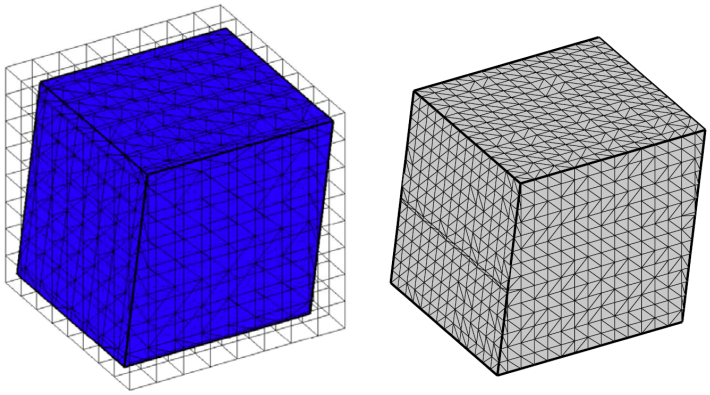


Figure 5: Example 1. Left: Geometry embedded in a Level 3 Cartesian mesh. Right: Triangulation of the surface in a Level 4 Cartesian mesh.

norm of the error is plotted in Figure 7 for different values of the penalty constant considering the Level 2 and 3 meshes (similar behavior was obtained for the rest of the mesh levels), and compared with the results obtained with Nitche's method. It can be seen that the proposed technique gave a wide range of κ values, from 4 to $4 \cdot 10^3$, for which the level of the error remains essentially unaffected for both $\mathcal{L}_8$ and $\mathcal{Q}_{20}$ elements. However, the values of κ for which the error level remains unaffected is narrower in Nitche's method, ranging from only $\kappa = 40$ to $\kappa = 4 \cdot 10^3$ for $\mathcal{L}_8$ elements and from $\kappa = 400$ to $\kappa = 4 \cdot 10^3$ for $\mathcal{Q}_{20}$ elements. It can also be observed that both techniques provide similar error levels for high levels of κ ($\kappa \geq 40$ for $\mathcal{L}_8$ elements and $\kappa \geq 400$ for $\mathcal{Q}_{20}$ elements).

In Figure 8 the energy and the L^2 norms of the discretization error are plotted as a function of the mesh size for $\mathcal{L}_8$ and $\mathcal{Q}_{20}$ elements, and for both the proposed technique (iRec) and Nitsche's method (Nit). The triangles in this figure show the theoretical optimal convergence rate that can be achieved. Three different values of κ were considered, taking into account the theoretical value $\kappa > C_p C_r$ that provides optimal convergence ($C_p = 13$ for $\mathcal{L}_8$ and $C_p = 21$ for $\mathcal{Q}_{20}$ elements, assuming that $C_r = 1$), $\kappa = 4$, $\kappa = 40$ and $\kappa = 400$.

As $\kappa = 4$ is lower than $C_p C_r$, the optimal convergence rate is not ensured. Indeed it is only achieved using the proposed method for $\mathcal{L}_8$ elements. The expected behavior in terms of convergence rate is observed in the numerical results for $\kappa = 40$ using the proposed method both for linear and quadratic elements, whereas Nitsche's method was only able to recover the optimum convergence rate for $\kappa = 400$ and linear elements. A reduction of the convergence rate for Nitsche's method can be seen in the last refinement step for $\mathcal{Q}_{20}$ elements. Nitsche's method would therefore require

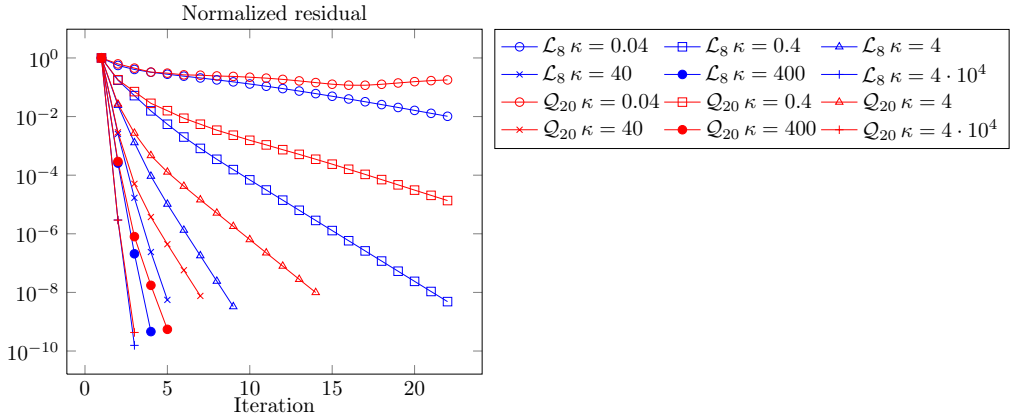


Figure 6: Example 1. Convergence of the residual in the Richardson iterations for the Level 2 mesh and different values of κ using linear and quadratic elements.

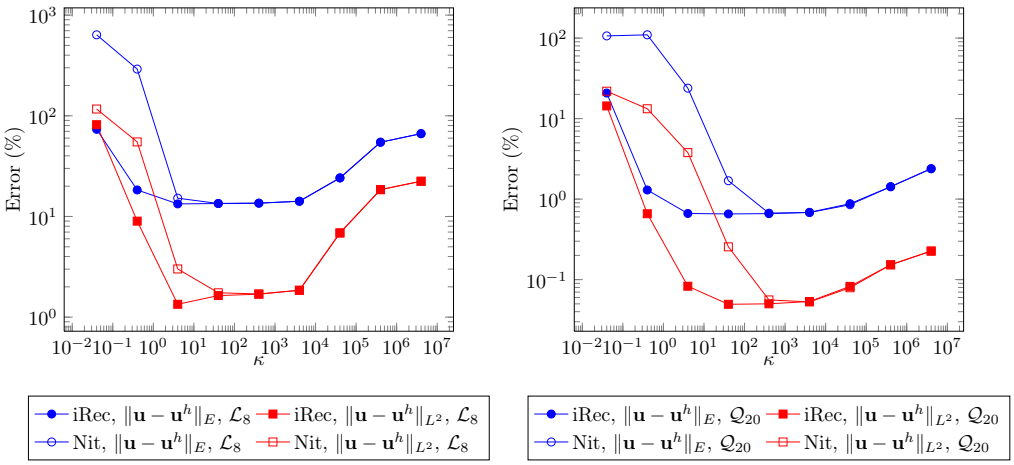


Figure 7: Example 1. Discretization error in energy norm and L^2-norm (%) for different values of the penalty constant κ. Results for Level 2 Cartesian mesh using linear and quadratic elements. The proposed method is denoted as iRec and Nitsche's method as Nit.

a κ higher than 400 to be able to successfully recover the optimum convergence rate during the entire refinement process for $\mathcal{Q}_{20}$ elements.

This example showed that, if a sufficiently high value of κ is used, the proposed technique provides results similar to those obtained with Nitsche's method. However, it is able to provide accurate results, with the optimal convergence rate, for considerably lower values of the stabilization parameter κ than Nitsche's method.

7.2. Example 2: Spherical domain

The second example uses a sphere as the exact geometry of the problem. Figure 9 shows the Level 3 and Level 4 meshes. The exact solution of the problem considered in this domain is a fourth-order polynomial defined in an infinite domain, which is imposed as a Dirichlet boundary condition on the surface of the sphere. The exact solution of the problem reads as:

$$
\begin{aligned}
E &= 1000,\ \nu = 0.3 \\
A &= \frac{E}{(2\,\nu - 1)\ (\nu + 1)} \\
u_x &= x^4 + 3\,y\,x^3 - 2\,x\,z^2 + y\,x\,z \\
u_y &= 7\,x^2\,y + y^4 - 2\,z\,y^3 \\
u_z &= -3\,x^2\,z^2 + 2\,y\,x\,z + z^3 \\
b_x &= A\left[(1-\nu)\ \left(12\,x^2 + 18\,y\,x\right) + \nu\ (14\,x + 2\,y - 12\,x\,z) + \left(\nu - \frac{1}{2}\right)\ (-2\,y + 12\,x\,z - 10\,x)\right] \\
\sigma_{xx} &= -A\left[\nu\ \left(7\,x^2 + 4\,y^3 - 6\,z\,y^2 - 6\,x^2\,z + 2\,y\,x + 3\,z^2\right) - (\nu - 1)\ \left(4\,x^3 + 9\,y\,x^2 - 2\,z^2 + y\,z\right)\right] \\
\sigma_{yy} &= -A\left[\nu\ \left(-6\,x^2\,z + 2\,y\,x + 3\,z^2 + 4\,x^3 + 9\,y\,x^2 - 2\,z^2 + y\,z\right) - (\nu - 1)\ \left(7\,x^2 + 4\,y^3 - 6\,z\,y^2\right)\right] \\
\sigma_{zz} &= -A\left[\nu\ \left(7\,x^2 + 4\,y^3 - 6\,z\,y^2 + 4\,x^3 + 9\,y\,x^2 - 2\,z^2 + y\,z\right) - (\nu - 1)\ \left(-6\,x^2\,z + 2\,y\,x + 3\,z^2\right)\right] \\
\sigma_{xy} &= \frac{E}{2\,(1+\nu)}\left(14\,x\,y + x\,z + 3\,x^3\right) \\
\sigma_{yz} &= \frac{E}{2\,(1+\nu)}\left(2\,x\,z - 2\,y^3\right) \\
\sigma_{xz} &= \frac{E}{2\,(1+\nu)}\left(x\,y - 4\,x\,z + 2\,y\,z - 6\,x\,z^2\right) \\
b_y &= A\left[(\nu - 1)\ \left(12\,y\,z - 12\,y^2\right) - \left(\nu - \frac{1}{2}\right)\ \left(9\,x^2 + 14\,y + z - 2\,x\right) + \nu\ \left(9\,x^2 + 2\,x + z\right)\right] \\
b_z &= A\left[\left(\nu - \frac{1}{2}\right)\ \left(6\,y^2 + 6\,z^2 + 4\,z - y\right) + \nu\ \left(y - 4\,z - 6\,y^2\right) - (\nu - 1)\ \left(6\,z - 6\,x^2\right)\right]
\end{aligned}
$$

In this example we use a technique that considers the exact geometry of the problem in the evaluation of volume and surface integrals. These integrals will be exactly evaluated up to the numerical integration errors [40]. Whatever the method used to impose Dirichlet boundary conditions in immersed boundary methods, considering the exact geometry of the domain in the evaluation of volume and surface integrals

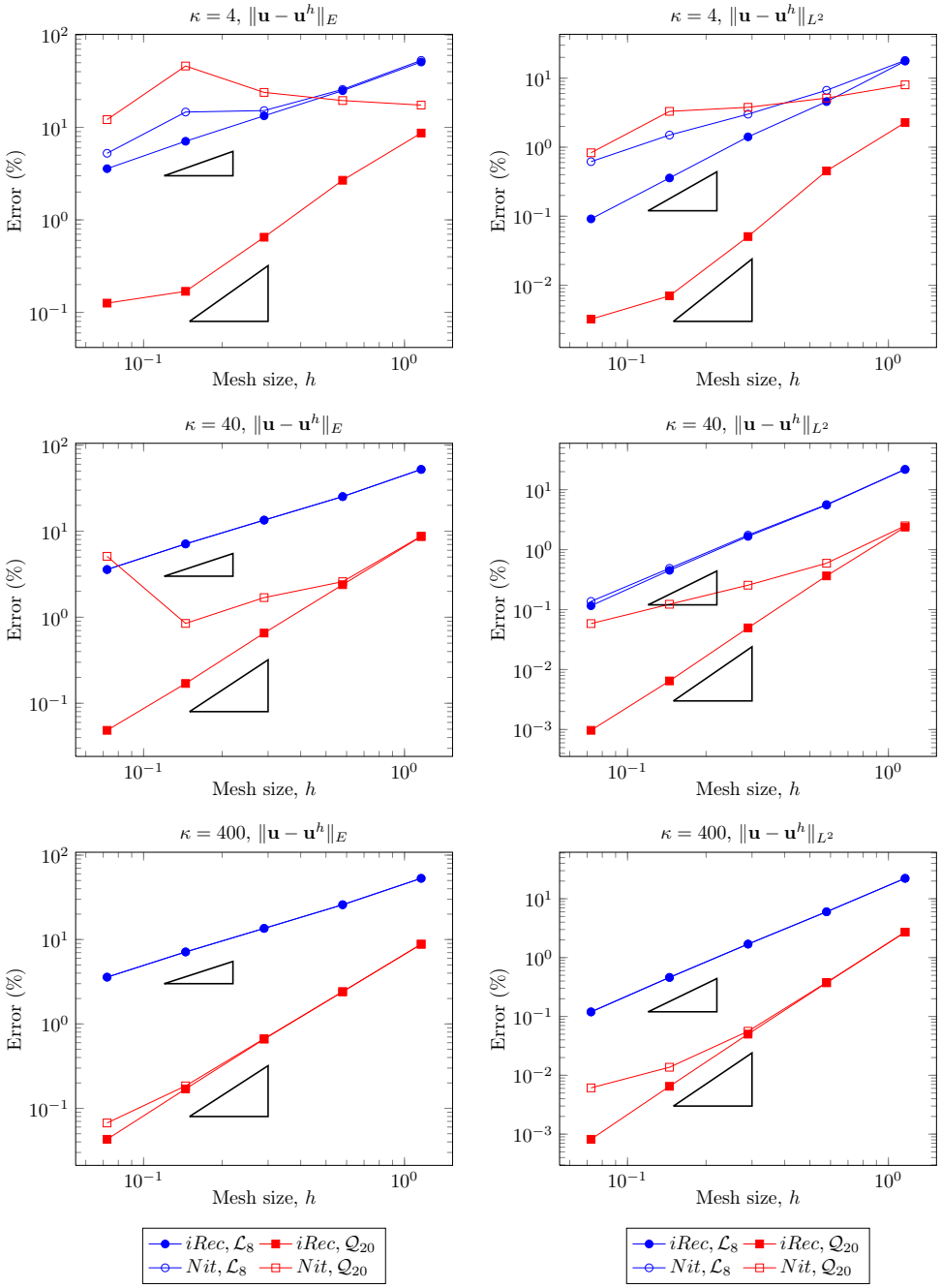


Figure 8: Example 1. Discretization error in energy norm as a function of the mesh size for $\mathcal{L}_8$ and $\mathcal{Q}_{20}$ elements. The triangles show the optimal convergence rate.

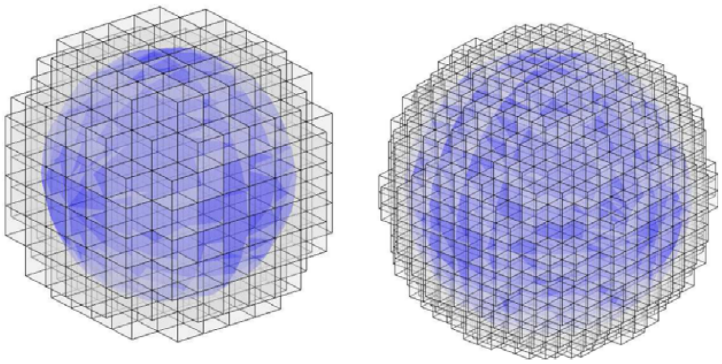


Figure 9: Spherical geometry embedded in a Cartesian mesh. Level 3 and Level 4 meshes.

is necessary to achieve the optimal convergence rate for both linear and quadratic elements. This is because approximations to the actual geometry, for example by a faceted representation of the surface, could lead to geometrical modeling errors that could spoil the convergence rate of the numerical method. Figure 10 shows the discretization error in energy and L^2 norms. The optimal values of the error convergence rates, represented by the triangles shown in the graphs, are obtained in all cases. We recall that the theoretical rate of convergence for linear elements is 1 in energy norm and 2 in L^2-norm. For quadratic elements it is 2 in energy norm and 3 in L^2-norm. The exact slopes of the finite element solution for $\mathcal{L}_8$ elements are 0.69, 0.97 and 0.97 in energy norm and 1.13, 1.79 and 1.96 in L^2-norm, whilst for $\mathcal{Q}_{20}$ elements the values are 1.89, 1.78 and 1.96 in energy norm and 2.67, 2.77 and 2.95 in L^2-norm. The value of the penalty constant was $\kappa = 40$.

7.3. Example 3: Cubic domain parallel to the Cartesian grid

Let us consider the exact solution of the fourth-order polynomial given in the second example, in a hexahedral domain with faces parallel and equidistant to the faces of the embedding mesh. This solution was used to impose the Dirichlet boundary condition on the surface. Figure 11 shows two Cartesian meshes of Level 3 corresponding to two different bounding boxes that could be used to analyze this geometry. Let η represent the ratio of the volume of domain contained in the boundary elements to the total volume of these elements ($\eta = 10\%$ in Figure 11 left and $\eta = 90\%$ in Figure

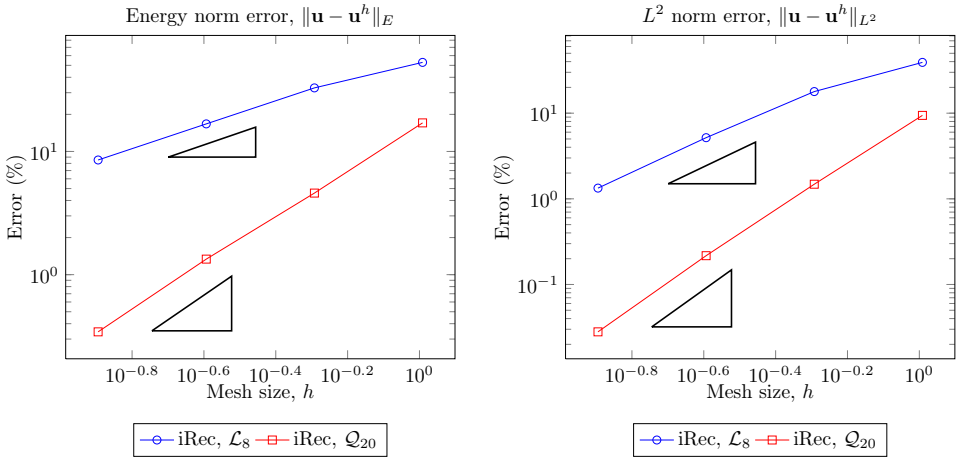


Figure 10: Example 2. Discretization error in energy norm for $\mathcal{L}_8$ and $\mathcal{Q}_{20}$ elements using $\kappa = 40$. The triangles show the optimal convergence rate.

11 right). We will use η to represent the different relative positions of the surface in the intersected boundary elements.

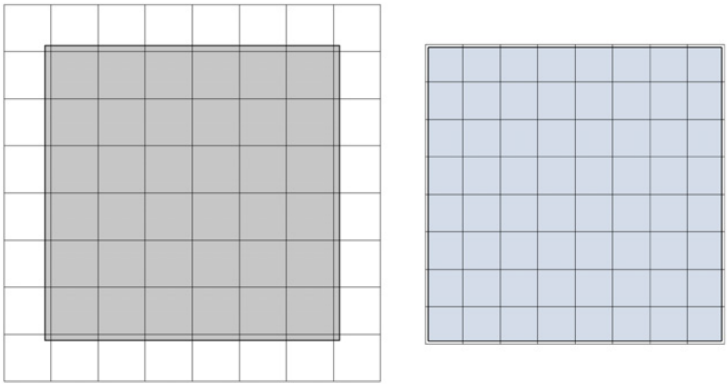


Figure 11: Example 3. Cartesian meshes with different values of the parameter η.

In order to test the influence of η on the error of the solution, sequences of uniformly refined meshes of $\mathcal{L}_8$ and $\mathcal{Q}_{20}$ elements were generated by adjusting the bounding box and the mesh level to keep the parameter η constant. The values of η selected for the analyses were 50%, 25%, 10%, 5%, 1%, 0.5%, 0.1% and 0.05%. This is a

challenging problem for Nitsche's method, as the constant that provides stability C_N (Equation (14)) increases as η is reduced and cannot be bounded for $\eta \to 0$.

With the numerical analyses of this problem we try to show that the proposed method is able to provide accurate solutions even when Nitsche's method fails to do so, *i.e.* for small values of η, but is numerically equivalent to Nitsche's method for larger values of η. In fact we compared three techniques, denoted as *iRec* - the iterative method proposed in this paper, *iNit* - the iterative method described in Section 3.3 and *Nit* - the Nitsche's method.

The magnitude used to compare the results obtained from these techniques is the exact L^2-norm error of tractions on the Dirichlet boundaries $\left\|\boldsymbol{\sigma}(\mathbf{u})\mathbf{n} - \boldsymbol{\sigma}(\mathbf{u}^h)\mathbf{n}\right\|_{L^2,\Gamma_D}$. Figure 12 shows in logarithmic scale the evolution of this magnitude for meshes of Level 3 and Level 4. The graphs for this example show that, in general terms, reducing η has a negative effect on the results obtained by all three techniques, but especially in the case of Nitsche's method. For $\eta = 0.05\%$, the best results obtained with Nitsche's method, which were for the highest value of the stabilization parameter ($\kappa = 4 \cdot 10^3$), are unacceptable ($\left\|\boldsymbol{\sigma}(\mathbf{u})\mathbf{n} - \boldsymbol{\sigma}(\mathbf{u}^h)\mathbf{n}\right\|_{L^2,\Gamma_D} = 46258.6\%$ for the Level 3 mesh and $\left\|\boldsymbol{\sigma}(\mathbf{u})\mathbf{n} - \boldsymbol{\sigma}(\mathbf{u}^h)\mathbf{n}\right\|_{L^2,\Gamma_D} = 7786.7\%$ for Level 4). Reducing η leads to increasingly higher required values of κ. On the one hand the use of these high κ values gives a high weight to the satisfaction of the Dirichlet boundary conditions and leads to an improvement of the magnitude considered in the comparisons in this example. However, on the other hand it increases the condition number of the system matrix, and might overweight the imposition of the Dirichlet BC on the surface at the expense of reducing the weight of the terms that account for the energy in the volume (similar ideas have already been put fordward by [18]). This implies high error levels in the results obtained in the elements cut by the Dirichlet boundary. This problem would be critical if the results obtained in these elements were magnitudes of interest to the analyst.

The *iNit* curves show a slightly better performance than the *Nit* curves for $\kappa = 4 \cdot 10^3$, and more significantly for $\kappa = 400$. However, further reducing the value of κ prevents the convergence of the iterative process in the full range of values of the volume ratio η (convergence was only obtained when $\kappa = 4$ for $\eta \geq 10\%$ and, when $\kappa = 40$, for $\eta \geq 1\%$ for the Level 3 mesh and for $\eta \geq 0.5\%$ in the case of Level 4). The results given in figure show that the best results are obtained with the proposed method (*iRec* curves). These curves are similar to the *Nit* and *iNit* curves for $\kappa = 4 \cdot 10^3$, but show a considerable improvement when κ is reduced. The optimum performance shown in the graphs for the *iRec* curves is at $\kappa = 4$ with the highest error levels around 30% and 26% (only obtained for $\eta \leq 0.1\%$) respectively for the Level 3 and Level 4 meshes, *i.e.* several orders of magnitude smaller than with the other two methods.

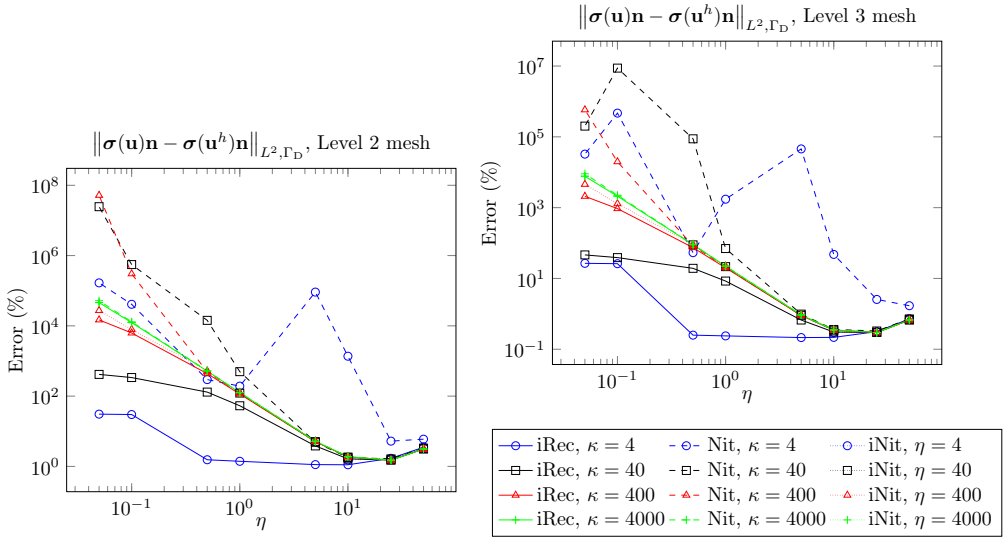


Figure 12: Example 3. Fourth-order polynomial in a cube. L^2-error of the traction field on the Dirichlet boundary for Level 2 and Level 3 meshes.

7.4. Example 4: Plasticity

In this last example we check the performance of the proposed technique when used to analyze a mechanical component considering elasto-plastic behavior of the material. The component analyzed is a rectangular plate with a central hole subjected to uniaxial monotonic traction, as represented in Figure 13. The highlighted 1/8 of the plate with the appropriate symmetry boundary conditions was used in the analyses. The material behavior is a von Mises plasticity bilinear model with yield stress $S_y = 24$ units of pressure and slope of the plastic zone $H = 225$ units of pressure. The Young modulus is $E = 1000$ units of pressure and the Poisson's ratio $\nu = 0.3$. The maximum traction applied is $\sigma_{max} = 0.9 \cdot S_y$. We use the proposed method to apply the Dirichlet boundary conditions on the symmetry boundaries. Although the theoretical analysis done in the paper to obtain the value of k is only valid for linear elasticity, we use the same values for non-linear problems, such as, the present example. Therefore, the value of the stabilizing constant is $\kappa = 40$ ($k = 100$ using the elastic material properties).

The reference solution was evaluated using ANSYS®[41]. We compared the displacement of Point **I** (shown in Figure 13) as a function of the load and the final distribution of the plastic zone, using the reference and the Cartesian grid methods. A sequence of analyses using mesh levels 2, 3, 4 and 5, with element faces not coplanar

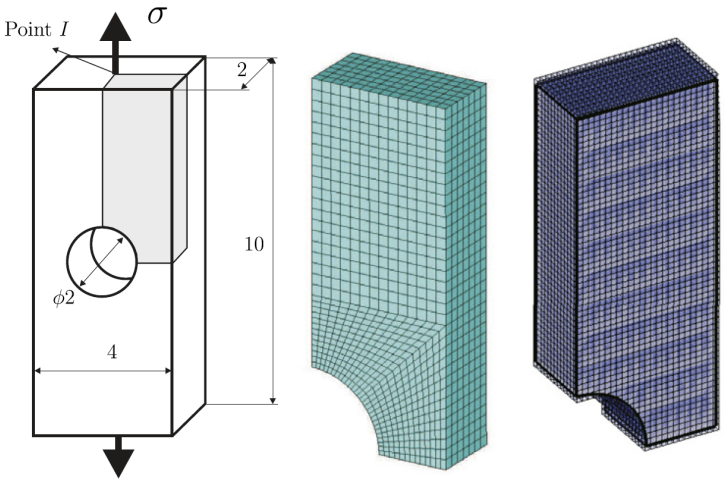


Figure 13: Example 4. Plate under traction. Geometrical model and finite element models. Centre: mesh of the reference model. Right: Level 5 Cartesian grid mesh.

to the symmetry surfaces, was run to test the convergence of the solutions obtained by the Cartesian grid method to the reference. The size of the elements in the reference model was similar to that of the Level 5 mesh, although the Ansys model gave smaller element sizes around the hole. The number of degrees of freedom of the Cartesian grid models was 396, 1935, 10392 and 65058, whereas the reference model had 92904 degrees of freedom. A comparison between the reference mesh and the Cartesian grid for Level 5 is shown in Figure 13.

Figure 14 compares, on the right, the zone of the models that underwent plasticity (shown in gray) evaluated by Ansys and by our Cartesian grid implementation. It can be observed that both zones are similar, even though different graphical representation techniques are used by each of the codes.

Figure 14 shows on the left plot the load-displacement curves obtained from the different analyses. It can be clearly observed that the curves obtained with the Cartesian grid smoothly converge to the reference curve, thus showing that the technique of imposing Dirichlet boundary conditions proposed in this paper can be used to provide similar results to those obtained with standard finite element implementations.

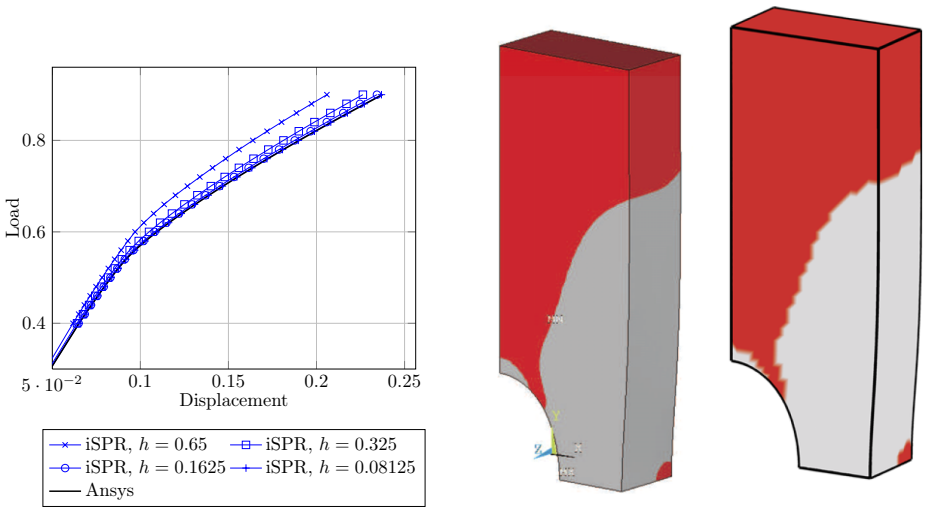


Figure 14: Example 4. Comparison of the FE models. Left: mesh of the reference model (Ansys). Right: Level 5 Cartesian grid mesh.

8. Conclusions

This paper describes a novel method of imposing Dirichlet boundary conditions suitable for immersed boundary Cartesian meshes in an approach based on the stabilized Lagrange multipliers method. This approach allows the Lagrange multipliers to be condensed element-by-element during the assembly process. The stabilization term is evaluated by using a smoothed stress field obtained from the Superconvergent Patch Recovery technique. An iterative procedure is defined to update the stabilizing term and the global convergence of the procedure is proved for a sufficiently large value of the stabilizing parameter κ. The numerical examples show the influence of κ both on the finite element error and on the convergence of the Richardson iterations. In the examples given, the finite element solution remains unchanged for a wide range of values of κ ($\kappa \in [40 - 4 \cdot 10^3]$) and the convergence of the Richardson iteration is verified for $\kappa > 4$.

The numerical results show that the optimal convergence rate of the finite element solution is obtained for both linear and quadratic elements, provided that the geometry of the problem is accurate enough. In particular, if the geometry is approximated using a linear subtriangulation the solution using linear element has an optimal convergence rate whilst the convergence using quadratic elements is suboptimal.

The numerical comparisons with Nitsche's method showed that in the general case, and especially for high values of the penalty parameter of the stabilization term κ, both techniques provide similar results. However, the proposed technique proved to be robust in providing accurate results even when the boundary of the domain comes very close to the element faces, where Nitsche's method is not robust. Moreover, the results obtained with the proposed technique improve as κ is reduced consequently with less likelihood of obtaining high condition numbers.

The proposed technique with no modifications was successfully applied to analyzing a problem with non-linear material behavior. Despite the fact that the plasticity covered a large area of the Dirichlet boundary, applying the proposed technique provided similar results to those obtained by the standard finite element method. Although further improvements could be gained by adapting the recovery technique to account for plastic deformations, the preliminary results of this work open up the possibility of obtaining the benefits of embedded domain methods in this type of non-linear problem.

Acknowledgments

The authors wish to thank the Spanish Ministerio de Economia y Competitividad for the financial support received through the project DPI2013-46317-R and Generalitat Valenciana through the project PROMETEO/2012/023. The authors are also grateful for the support of the Framework Programme 7 Initial Training Network Funding under grant number 289361 "Integrating Numerical Simulation and Geometric Design Technology (INSIST)".

References

[1] Cottrell JA, Hughes TJR, Bazilevs Y. *Isogeometric Analysis: Toward Integration of CAD and FEA*. Wiley, 2009. 1

[2] Bordas SPA, Rabczuk T, Ródenas JJ, Kerfriden P, Moumnassi M, Belouettar S. Recent Advances Towards Reducing the Meshing and Re-meshing Burden in Computational Sciences. *Computational Technology Reviews* 2010; **2**:51–82. 1

[3] Strouboulis T, Copps K, Babuška I. The generalized finite element method: an example of its implementation and illustration of its performance. *International Journal for Numerical Methods in Engineering* 2000; **47**(8):1401–1417. 1

[4] Moumnassi M, Belouettar S, Bechet E, Bordas SPA, Wuoirin D, Potier-Ferry M. Finite element analysis on implicitly defined domains: An accurate representation based on arbitrary parametric surfaces. *Computer Methods in Applied Mechanics and Engineering* 2011; **200**(5–8):774–796. 1

[5] Nadal E, Ródenas JJ, Albelda J, Tur M, Tarancón JE, Fuenmayor FJ. Efficient Finite Element Methodology based on Cartesian Grids: Application to Structural Shape Optimization. *Abstract and Applied Analysis* 2013; **2013**. 1

[6] Nadal E. *Cartesian Grid FEM (cgFEM): High Performance h-adaptive FE Analysis with Efficient Error Control. Application to Structural Shape Optimization. PhD Thesis*. Universitat Politècnica de València, 2014. 1, 2

[7] Sevilla R, Fernández-Méndez S, Huerta A. 3D-NURBS-enhanced Finite Element Method (NEFEM). *International Journal for Numerical Methods in Engineering* 2011; **88**(2):103–125. 1

[8] Brezzi F, Fortin M. *Mixed and hybrid finite element methods*. Springer-Verlag New York, Inc., 1991. 1, 2.1

[9] Béchet E, Moës N, Wohlmuth B. A stable lagrange multiplier space for stiff interface conditions within the extended finite element method. *International Journal for Numerical Methods in Engineering* 2009; **78**(8):931–954. 1, 2.1

[10] Hautefeuille M, Annavarapu C, Dolbow JE. Robust imposition of dirichlet boundary conditions on embedded surfaces. *International Journal for Numerical Methods in Engineering* 2012; **90**(1):40–64. 1, 2.1, 3.1, 1

[11] Barbosa HJC, Hughes TJR. The finite element method with lagrange multipliers on the boundary: circumventing the babuska-brezzi condition. *Computer Methods in Applied Mechanics and Engineering* 1991; **85**(1):109–128. 1, 3.1

[12] Barbosa HJC, Hughes TJR. Boundary lagrange multipliers in finite element methods: Error analysis in natural norms. *Numerische Mathematik* 1992; **62**(1):1–15. 1, 3.1

[13] Hansbo A, Hansbo P. An unfitted finite element method based on nitsche method for elliptic interface problems. *Computer Methods in Applied Mechanics and Engineering* 2002; **191**(47–48):5537–5552. 1

[14] Hansbo P, Lovadina C, Perugia I, Sangalli G. A lagrange multiplier method for the finite element solution of elliptic interface problems using non-matching meshes. *Numerische Mathematik* 2005; **100**(1):91–115. 1

[15] Dolbow J, Harari I. An efficient finite element method for embedded interface problems. *International Journal for Numerical Methods in Engineering* 2009; **78**(2):229–252. 1, 3.1

[16] Sanders JD, Dolbow JE, Laursen TA. On methods for stabilizing constraints over enriched interfaces in elasticity. *International Journal for Numerical Methods in Engineering* 2009; **78**(9):1009–1036. 1, 3.1

[17] Haslinger J, Renard Y. A new fictitious domain approach inspired by the extended finite element method. *SIAM J. Numer. Anal.* 2009; **47**(2):1474–1499. 1, 3.1, 3.2

[18] Schott B, Wall W. A new face-oriented stabilized xfem approach for 2d and 3d incompressible navier-stokes equations. *Computer Methods in Applied Mechanics and Engineering* 2014; **276**:233–265. 1, 3.1, 3.1, 7.3

[19] Burman E, Hansbo P. Fictitious domain finite element methods using cut elements: Ii. a stabilized nitsche method. *Applied Numerical Mathematics* 2012; **62**(4):328–341. 1, 3.1, 3.1

[20] Amdouni S, Hild P, Lleras V, Moakher M, Renard Y. A stabilized lagrange multiplier method for the enriched finite-element approximation of contact problems of cracked elastic bodies. *ESAIM: Mathematical Modelling and Numerical Analysis* 2012; **46**(4):813–839. 1

[21] Annavarapu C, Hautefeuille M, Dolbow JE. A robust nitche's formulation for interface problems. *Computer Methods in Applied Mechanics and Engineering* 2012; **225–228**:44–54. 1

[22] Sanders JD, Laursen TA, Dolbow JE. A nitsche embedded mesh method. *Computational Mechanics* 2012; **49**(2):243–257. 1, 3.1

[23] Annavarapu C, Hautefeuille M, Dolbow JE. A nitsche stabilized finite element method for frictional sliding on embedded interfaces. part i: single interface. *Computer Methods in Applied Mechanics and Engineering* 2014; **268**:417–436. 1, 3.1

[24] Sanders JD, Laursen TA, Dolbow JE. Approximate imposition of boundary conditions in immersed boundary methods. *International journal for numerical methods in engineering* 2009; **80**(11):1379–1405. 1, 3.2

[25] Burman E. Projection stabilization of lagrange multipliers for the imposition of constraints on interfaces and boundaries. *Numerical Methods for Partial Differential Equations* 2014; **30**(2):567–592. 1

[26] Tur M, Albelda J, Nadal E, Ródenas JJ. Imposing dirichlet boundary conditions in hierarchical cartesian meshes by means of stabilized lagrange multipliers. *International Journal for Numerical Methods in Engineering* 2014; **98**(6):399–417. 1, 4

[27] Lew A, Buscaglia G. A discontinuous-galerkin-based immersed boundary method. *International Journal for Numerical Methods in Engineering* 2008; **76**(4):427–454. 1

[28] Liu F, Borja RI. Stabilized low-order finite elements for frictional contact with the extended finite element method. *Computer Methods in Applied Mechanics and Engineering* 2010; **199**:2456–2471. 1

[29] Bochev PB, Dohrmann CR, Gunzburger MD. Stabilization of low-order mixed finite elements for the stokes equations. *SIAM J. Numer. Anal.* 2006; **44**(1):82–101. 1

[30] Barrenechea G, Chouly F. A local projection stabilized method for fictitious domains. *Applied Mathematics Letters* 2012; **25**(12):2071–2076. 1

[31] Baiges J, Codina R, Henke F, Shahmiri S, Wall WA. A symmetric method for weakly imposing dirichlet boundary conditions in embedded finite element meshes. *International Journal for Numerical Methods in Engineering* 2012; **90**(5):636–658. 1

[32] Zienkiewicz OC, Zhu JZ. The superconvergent patch recovery and a posteriori error estimates. Part 2: Error estimates and adaptivity. *International Journal for Numerical Methods in Engineering* 1992; **33**(7):1365–1382. 1, 3.2, 5

[33] Brezzi F. On the existence, uniqueness and approximation of saddle-point problems arising from lagrangian multipliers. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique* 1974; **8**(R2):129–151. 2.1

[34] Pitkaranta J. A conforming finite element method with lagrange multipliers for the biharmonic problem. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique* 1980; **14**(3):309–324. 2.1, 4

[35] Stenberg R. On some techniques for approximating boundary conditions in the finite element method. *Journal of Computational and Applied Mathematics* 1995; **63**(1–3):139–148. 2.1, 3.1, 3.2, 4

[36] Jiang W, Annavarapu C, Dolbow J, Harari I. A robust nitsche's formulation for interface problems with spline-based finite elements. *International Journal for Numerical Methods in Engineering* 2015; **104**(7):676–696. 3.1

[37] Gerstenberger A, Wall WA. An embedded dirichlet formulation for 3d continua. *International Journal for Numerical Methods in Engineering* 2010; **82**(5):537–563. 4

[38] Ródenas JJ, Tur M, Fuenmayor FJ, Vercher A. Improvement of the superconvergent patch recovery technique by the use of constraint equations: the SPR-C technique. *International Journal for Numerical Methods in Engineering* 2007; **70**(6):705–727. 2

[39] Saad Y. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, 2003. 6.2

[40] Marco O, Sevilla R, Zhang Y, Ródenas JJ, Tur M. Exact 3D boundary representation in finite element analysis based on Cartesian grids independent of the geometry. *International Journal for Numerical Methods in Engineering* 2015; **103**:445–468. 7.2

[41] *Ansys 15.0 ANSYS Academic Research, Release 15.0, Help System*. ANSYS, Inc. 7.4

PAPER C

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

O. Marco, J. J. Ródenas, J. M. Navarro-Jiménez and M. Tur

Preprint submitted to Computers & Structures

Abstract

This paper proposes a novel algorithm to generate 3D h-adaptive meshes for an Immersed Boundary Method based on the use of Cartesian Finite Element meshes and the integration techniques of the so-called NEFEM (NURBS-Enhanced Finite Element Method). In order to increase the accuracy of the results at a minimum computational cost, with this work we seek to keep the efficient Cartesian structure of the mesh and the associated data structure, during the whole analysis process, while considering the exact boundary representation of the domain given by CAD (NURBS or T-spline) surfaces. Within the framework of Cartesian Finite Element meshes, two important contributions of this paper are a) the methodology used for the intersection between mesh and geometry, which represents a relevant challenge due to their independence; and b) a robust procedure to generate the integration subdomains that exactly represent the domain given by a CAD model which is of major importance in problems like contact analysis and shape optimization. The contributions of this paper show that the Immersed Boundary Methods based on Cartesian grids can be considered as a robust and reliable tool in terms of accuracy and computational efficiency. Numerical examples will show proper convergence of the method and the capability of meshing complex 3D geometries.

Key words

Cartesian grids; h-refinement; NURBS; NEFEM

Contents

1 Introduction **177**

2 Cartesian grids with exact representation of the geometry **179**

3 Geometry-mesh intersection **182**

4 Integration patterns **187**

5 Mesh refinement **195**
5.1 Geometrical refinement 196
5.2 Error-based refinement 200

6 Numerical examples **202**
6.1 Convergence analysis 202
6.2 Geometrical refinement sample 207

7 Conclusions **210**

A *h*-refinement criteria based on error estimation **212**

Bibliography **216**

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 appearance 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 necessary for the resolution 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 meshing techniques, 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* [10,

11, 12]. In octree-based mesh generators [13, 14, 15] 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) [16], the Immersed Finite Element Method (IFEM) [17] or the Finite Cell Method (FCM)[18, 19, 20]. 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 [21, 22] or bio-mechanics [23, 24, 25]. 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[26, 27], 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[28] 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[29, 30] or T-Splines[31], 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) [32, 33].

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[34] or techniques in which the geometry is defined implicitly by level sets[35]. We used the NURBS-Enhanced integration techniques because their consistency considering the exact geometric description[36] is of major importance when dealing with CAD models in applications such as shape optimization or contact between bodies.

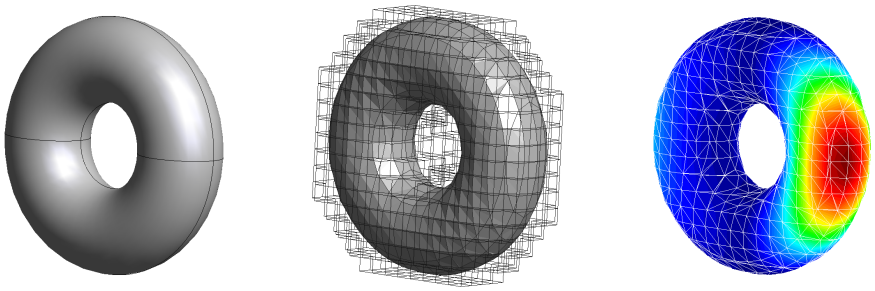
This contribution will show how the capabilities of the cgFEM methodology have been improved by developing h-adaptive analysis techniques. These techniques have been successfully implemented in FEAVox in order to handle complicated CAD models without renouncing the traditional properties of embedded 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 A.

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

The present work is the logical continuation of [28], 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.



(a) Torus geometry, $\Omega_{\texttt{Phys}}$. (b) Approximation mesh, $\Omega_{\texttt{Approx}}$. (c) Integration mesh, $\Omega^h_{\texttt{Phys}}$.

Figure 1: Immersed Boundary Method environment.

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 $\mathtt{n}^i_{\mathtt{el}}$ disjoint cubes of uniform size, where $\mathtt{n}^{i+1}_{\mathtt{el}} = 8\mathtt{n}^i_{\mathtt{el}}$. 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.

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 expense 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: bound-

ary, 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 a 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).

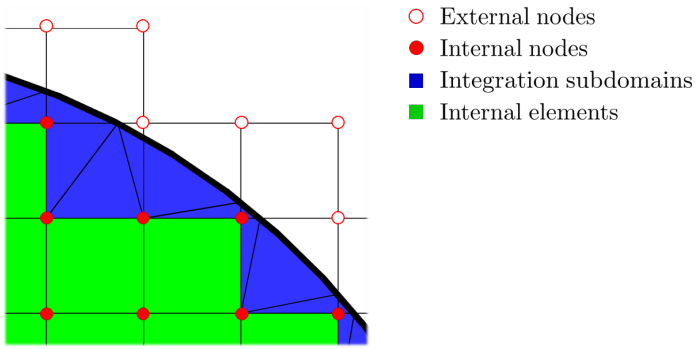


Figure 2: Section of a 3D Cartesian grid showing the different types of elements and nodes.

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 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 $\Omega_{\texttt{B}}^{\texttt{Phys}}$ employing a tetrahedralization of this region in each boundary element that incorporates the exact boundary representation of $\Omega_{\texttt{Phys}}$. Numerical integration over the region $\Omega_{\texttt{B}}^{\texttt{phys}}$ is then accomplished by integrating over each subdomain of the tetrahedralization. The strategy proposed in the NURBS-Enhanced Finite Element Method (NEFEM)[32, 33] 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 [37]. 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

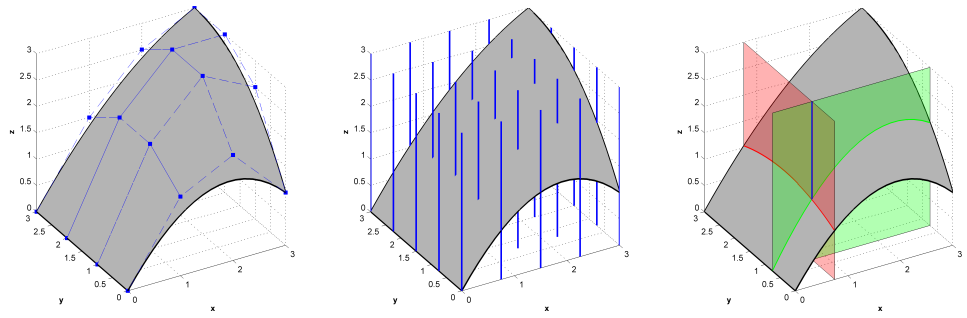
As a logical consequence of the independence between the analysis mesh and the geometrical model, the problem arises of discriminating the parts of the mesh inside and outside the model. The simplest approach is to find the intersections of the physical boundary with the edges of the Cartesian grid elements. 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[35], 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 involved 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[38, 39, 40, 41, 42].

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.



(a) Parametric surface. (b) Axes in direction Z. (c) Cartesian planes of an axis.

Figure 3: Example of surface and Cartesian axes.

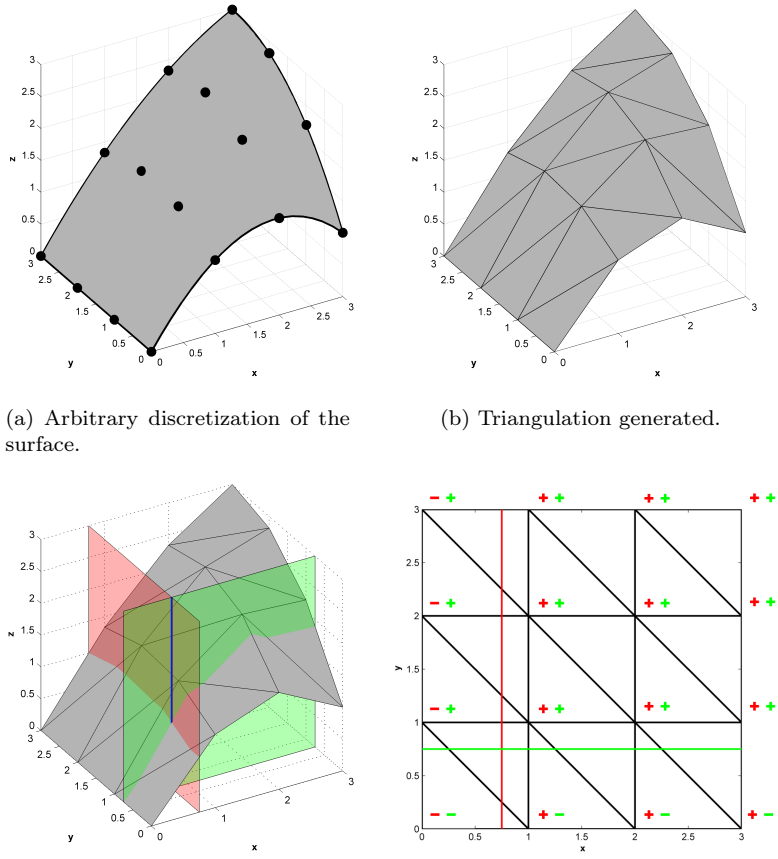
In addition, we know that each one of the Cartesian axes will be defined by two Cartesian planes, as shown in Figure 3c, where it can be seen that the intersection between a YZ-plane, defined by the coordinate x, and a XZ-plane, defined only by y, yields a Z-axis.

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 [28]). 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) Arbitrary discretization of the surface.

(b) Triangulation generated.

(c) Triangulation and intersection planes.

(d) 2D view of the level sets.

Figure 4: Procedure for the initial Newton-Raphson guess.

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

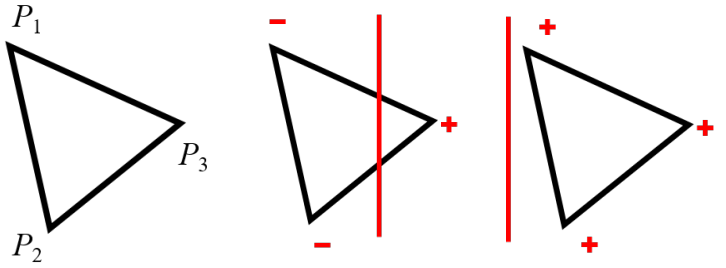
a view of the level sets calculated in the parametric space. The signs of the distances to the planes XZ and YZ are in red and green, respectively. Our strategy consists of finding triangles that are cut at the same time by the two planes defining the intersecting axis.

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).

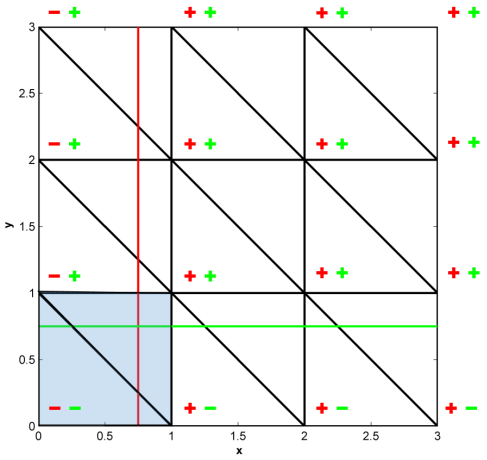
It should be noted that the fact of meeting the criteria does not necessarily mean the intersection exists. Indeed, if we use a linear triangulation to discretize an arbitrary parametric surface, it could happen that a plane intersects the triangles that had not been detected by the previous criteria. 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 $C_i = max\{C_i^x, C_i^y, C_i^z\}$ where $C_i^x = max\{P_i^x\} - min\{P_i^x\}$ 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 $\{|d_1|, |d_2|, |d_3|\} < C_i$, 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 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 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.



(a) Element of the triangulation. (b) Cutting plane. (c) Non-cutting plane.

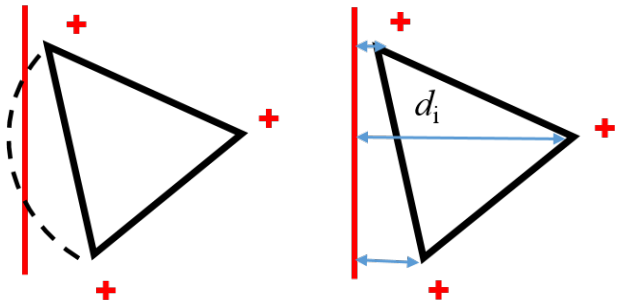


(d) Target triangles for intersection.

Figure 5: Intersecting the Cartesian planes and the triangulation.

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



(a) Case of intersection not detected.

(b) Distances of points to plane.

Figure 6: Ambiguous intersection between triangles and planes.

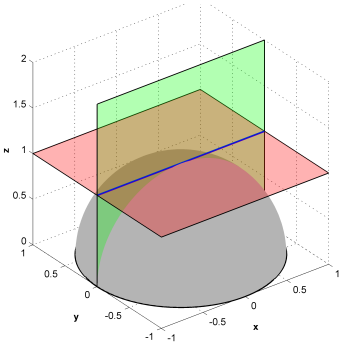
of elements as internal, boundary or external, is automatically achieved by counting the number of internal and external nodes in each element.

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

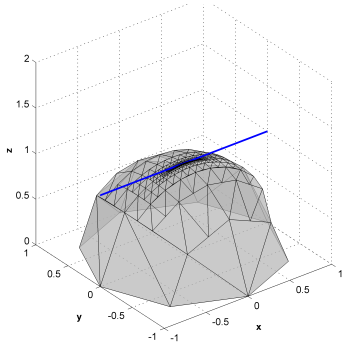
4. Integration patterns

The FEM requires the computation of integrals over the domain, $\Omega_{\mathtt{Phys}}$, 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.

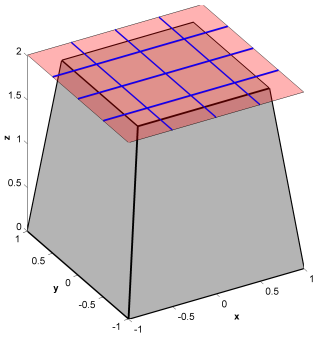
As explained in [28], internal elements, $\Omega_{\mathtt{I}}$, are standard finite elements in which the integration is performed using a tensor product of one-dimensional Gauss quadratures with the specified number of points in each direction. Nevertheless, the contribution of the boundary elements, $\Omega_{\mathtt{B}}$, 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 $\Omega_{\mathtt{B}}^{\mathtt{Phys}} = \Omega_{\mathtt{B}} \cap \Omega^{\mathtt{Phys}}$ (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, $\Omega_{\mathtt{B}}^{\mathtt{Phys}}$ can be extremely complex.



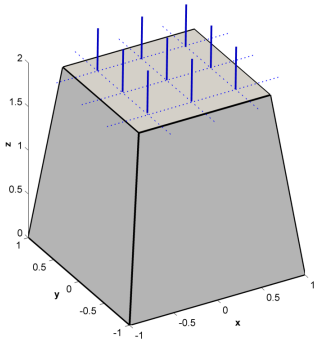
(a) Axis tangent to a surface.



(b) Resulting subdivided triangulation.



(c) Plane tangent to a surface.



(d) Intersection of normal axes only.

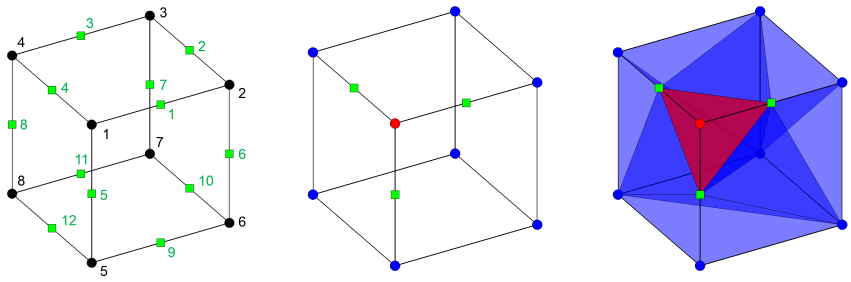
Figure 7: Treatment of tangent situations.

The approach proposed in [28] to perform the integration over $\Omega_B^{\texttt{Phys}}$ 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 [43], 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 8a, 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 8a 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 8a. Flat surface tetrahedrons will be considered to define the templates.

Figure 8b 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 tetrahedralization, for example using the Delaunay tetrahedralization, to discretize the element into subdomains (Figure 8c). 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.



(a) Numbering of nodes and edges in a reference element.

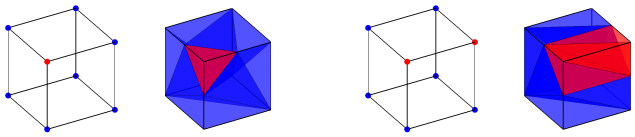
(b) Nodal and edge intersection topology.

(c) Tetrahedralization.

Figure 8: Integration pattern example.

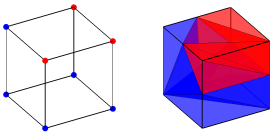
In the previous study [28], it was assumed that the edges of the elements are intersected, at most, once by the boundary of the physical domain. From this premise, we need only 7 out of 14 templates of the original MC algorithm (1, 2, 5, 8, 9, 11 and 14, see [43]). The seven patterns considered are depicted 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.

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 Fig-

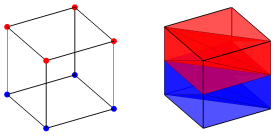


(a) Configuration 1.

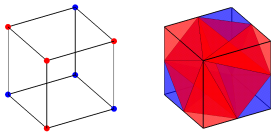
(b) Configuration 2.



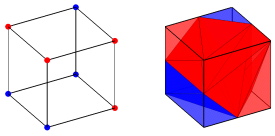
(c) Configuration 3.



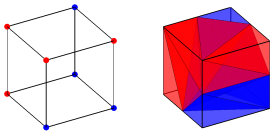
(d) Configuration 4.



(e) Configuration 5.



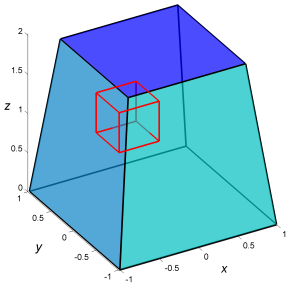
(f) Configuration 6.



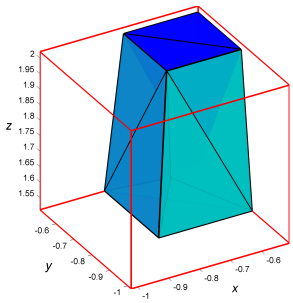
(g) Configuration 7.

Figure 9: Intersection patterns inspired on the MC algorithm. Nodal topology (left) and tetrahedralization (right).

ure 10a. The proposed method is to evaluate these elements individually, generating specific sets of tetrahedra, using for instance a Delaunay procedure, as in Figure 10b.



(a) Element intersected by several patches of a trapezoidal prism.



(b) Detail of the resulting tetrahedralization.

Figure 10: Exception to the intersection patterns.

Following [32], 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 11). 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.

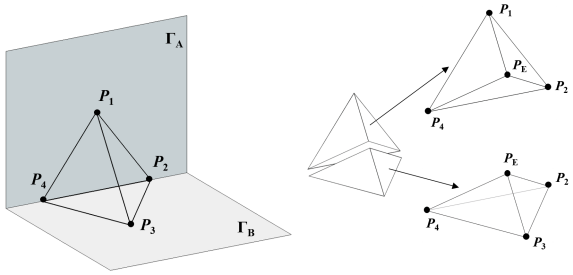


Figure 11: Subdivision of tetrahedrons with two faces on different parametric boundaries.

The second example considers an element Ω_T with three faces on different surfaces, as represented in Figure 12. 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 linear convex combination of P_F and original boundary faces of Ω_T, having at most one face on a parametric boundary.

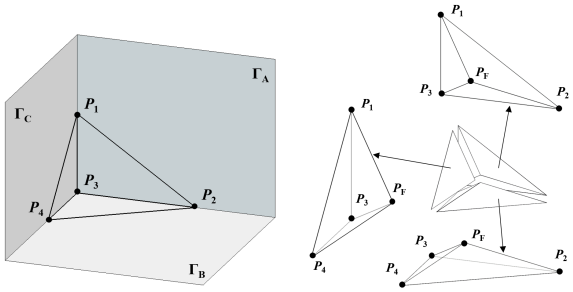
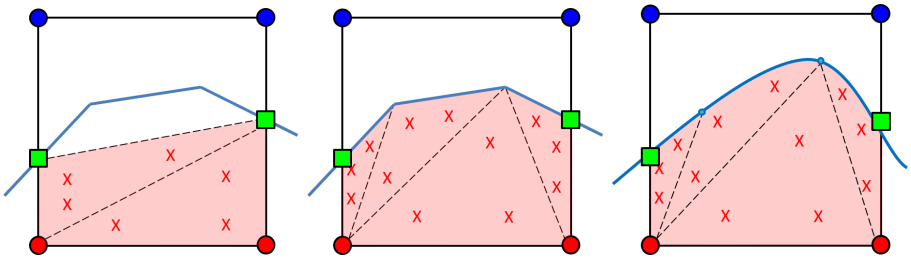


Figure 12: Subdivision of tetrahedrons with three faces on different parametric boundaries.

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 13a. 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 13b. Lastly, if the model is defined with high-order piecewise polynomials we could always use the NEFEM integration (Figure 13c). 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.*



(a) Subdomains template.

(b) Exclusive subdomain creation.

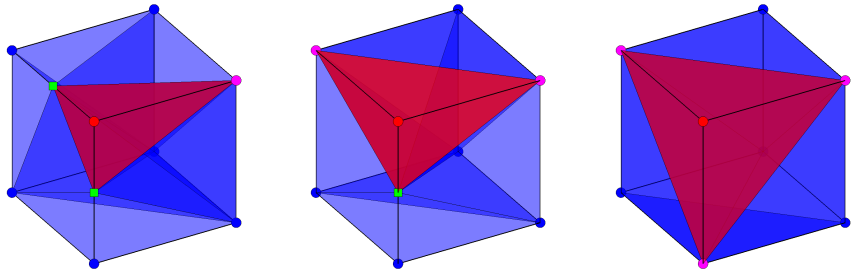
(c) Exclusive subdomain creation and NURBS-Enhanced integration.

Figure 13: Treatment of a linear piecewise boundary within an element.

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 9a, we can quickly see that we have as many options as intersections we can move to the nodes (see Figure 14) the boundary nodes being the magenta dots.



(a) One boundary node. (b) Two boundary nodes. (c) Three boundary nodes.

Figure 14: Configuration 1. Different options when considering surface intersections on 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 8, which yields the cases seen in Figure 14. The numbering used is that of the one defined in Figure 8a.

Starting with the case shown in Figure 14a, 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 14a 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 9a.

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.

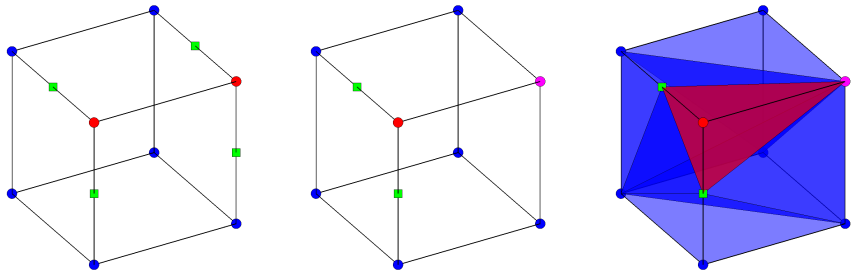
4. Boundary node to intersection. Figure 14a shows that we only count intersections 4 and 5, but Configuration 1 needs intersections 1, 4 and 5. Then, knowing that node 2 is related to intersections 1, 2 and 6, in which 2 and 6 are not used in this pattern, we can assume that intersection 1 is equivalent 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 14a, 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 14b and 14c, 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 15a 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 15b. The final tetrahedralization will be equivalent to the one shown in Figure 14a, which is topologically identical to the one obtained from Configuration 1 (Figure 14a), 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 15a and 15b we can observe how node 2 is transformed from an internal (red dot) to a boundary node (magenta dot). This violates the re-*

quirement 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.



(a) Intersection pattern. (b) One boundary node. (c) Resulting tetrahedralization.

Figure 15: Configuration 2. Degeneration equivalent to Configuration 1.

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)[44, 45] 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

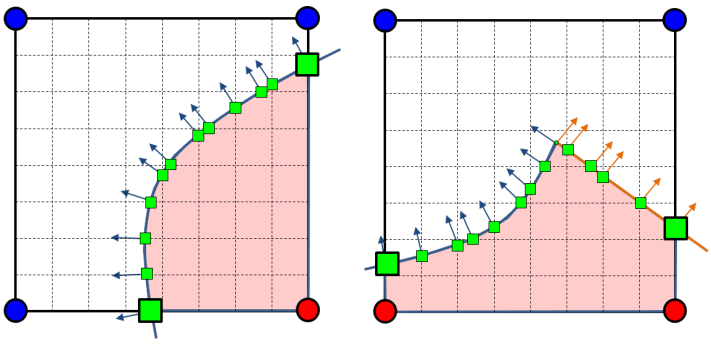
The refinement based on the features of the geometrical model is widely used in FEA, since the user can easily identify where the mesh should be finer to properly capture the boundaries of the models. For any kind of geometrical representation, from tesselations to advanced parametric surfaces, it is 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 16 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.



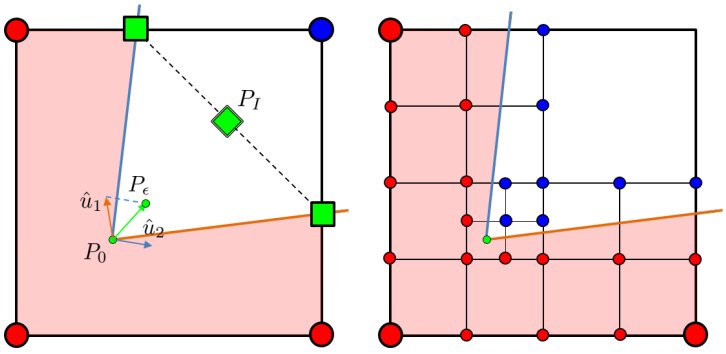
(a) Smooth boundary. (b) Sharp edge.

Figure 16: Geometrical features within elements. Variation of unit vectors.

Figure 16a 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 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 16b 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.

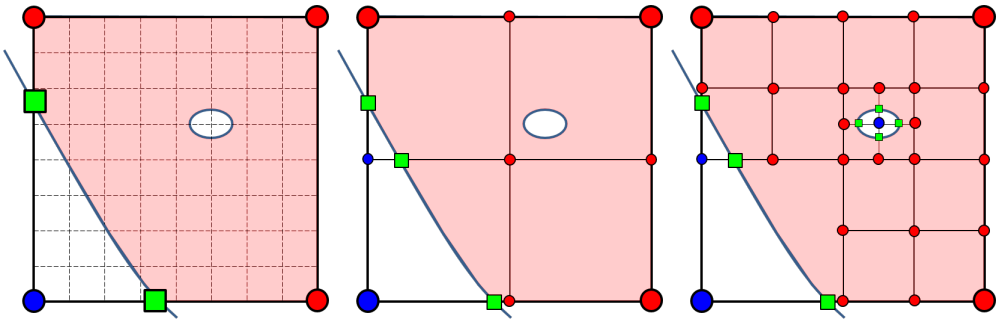
The next criterion completes the previous one and is related to the nature of the problem. FEAVox was first implemented to solve linear elasticity problems in which some internal corners could originate singularities and produce large gradients when evaluating displacements or stresses. These cases require an adequate discretization around the singularities to obtain a better representation of the singular solution. In Figure 17a 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 17b. In 3D this evaluation will occur at several points along the intersections curve between surfaces.



(a) Measure of distances. (b) Example of refinement.

Figure 17: Identification of singularities.

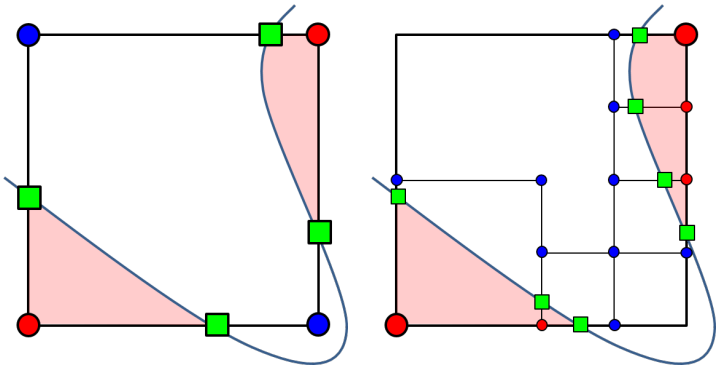
The ability to represent all the small features of a geometrical model is very important for all mesh generators, especially if we aim to develop a mesh generator in which the mesh is independent of the geometry. Figure 18a gives a clear example in which a small feature, a small ellipse in this case, 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 18b. 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.



(a) Coarse element. (b) First step of refinement. (c) Final step of refinement.

Figure 18: Detection of small features.

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 account because they refer to non-ambiguous configurations. In [28] 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 19a). 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 19b. 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.



(a) 2D projection of the pattern. (b) Solution through refinement.

Figure 19: Undefined pattern and elimination through refinement.

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

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

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

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, 5, 46, 47] use the residuals of the approximate solution to evaluate the error, the Constitutive Relation Error (CRE) [48] consisting in the comparison of statically admissible stress fields with

kinematically admissible stress fields, the estimators based on dual analysis [49, 50] and making use of two solutions of the problems. Finishing the classification, we find the recovery based error estimators. Proposed by Zienckevick and Zhu [51], these estimators use a recovered solution, $\boldsymbol{\sigma}^*$, instead of the exact solution $\boldsymbol{\sigma}$ to measure the error [52, 53].

Assuming 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 [26, 54].

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 [51, 55] in which the optimality criterion is that of equidistributing the error on the elements of the new mesh. In [55] 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).

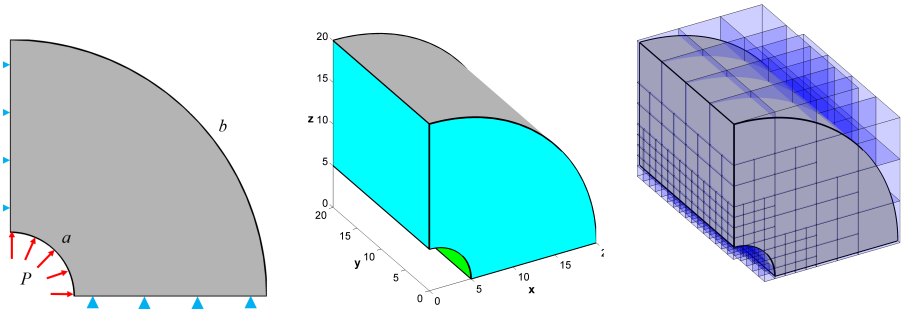
Replacing $\|\mathbf{e}^{(i)}\|$ in (4) by the estimation given in (3) we obtain the practical formula to evaluate the new element sizes. After obtaining the element size it is easy to find the refinement level necessary for the elements. Complete details of the procedure to reach this expression are given in 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 20. 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 $P = 1$.



(a) (b) (c)

Figure 20: 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:

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

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

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

$$\begin{gathered} u_x = u_r\cos(\phi), \qquad u_y = 0, \qquad u_z = u_r\sin(\phi) \\ \sigma_x = \sigma_r\cos(\phi)^2+\sigma_\phi\sin(\phi)^2, \qquad \sigma_z = \sigma_r\sin(\phi)^2+\sigma_\phi\cos(\phi)^2, \\ \sigma_y = \nu\,(\sigma_x+\sigma_z), \qquad \tau_{xz} = (\sigma_r-\sigma_\phi)\sin(\phi)\cos(\phi), \qquad \tau_{xz} = \tau_{yz} = 0 \end{gathered} \tag{7}$$

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 21. 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).

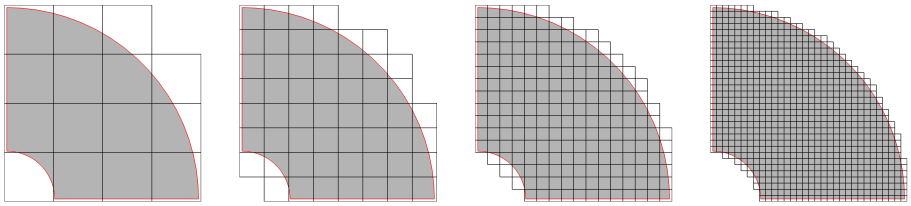


Figure 21: Front view of the first four meshes of the uniform refinement process (L8 and Q20).

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 22 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 [56], 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 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

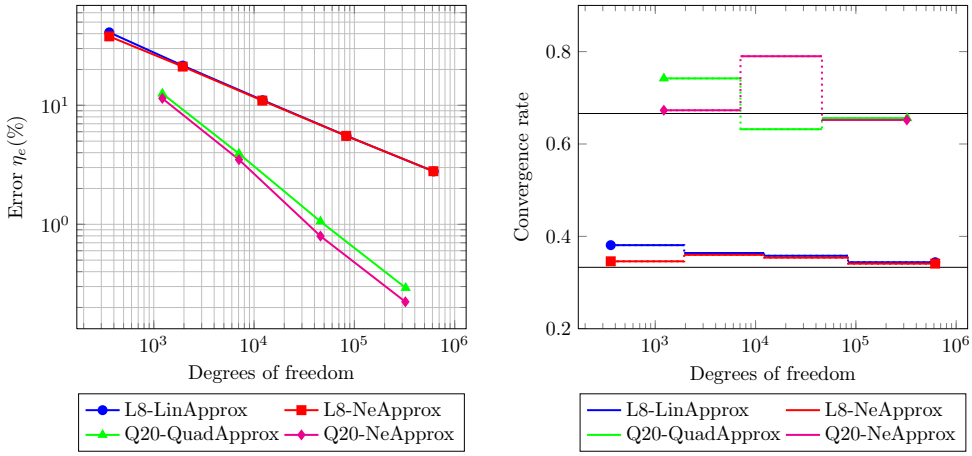


Figure 22: Cylinder convergence study. (Left) Relative error in energy norm. (Right) Convergence rate.

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.

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 23 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 is the volume integrated with the finite element mesh and V is the exact volume of the model shown in Figure 20. 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

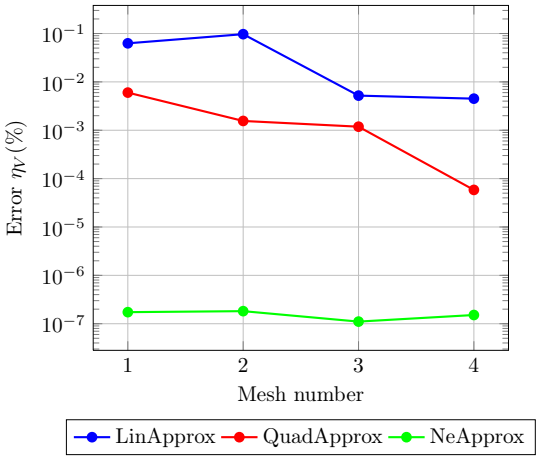


Figure 23: Cylinder volume convergence study.

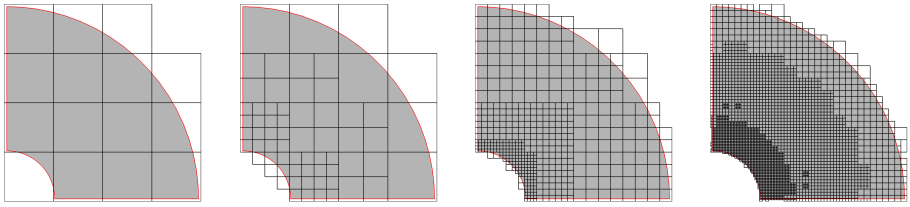
volume. Since the mesh is fixed, the refinement does not occur in the same uniform way in both surfaces thus making more pronounced this effect in coarse meshes. NURBS-Enhanced volume integration shows a very low almost constant error during the process, several orders of magnitude lower than approximating 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 21. 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.

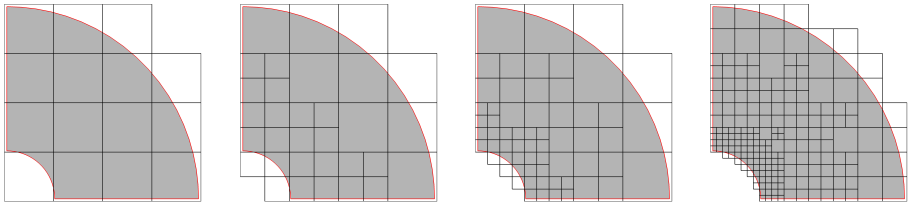
Approx. Type	Num. Gauss points	Error $\eta_V(\%)$	Integration time (s)
Linear	35524	0.0052	3.440
NEFEM	69226	1.4968e-5	14.746
NEFEM	76187	5.5125e-7	14.871
NEFEM	81754	4.2966e-9	15.549
NEFEM	94282	1.2212e-11	16.025

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

We also performed an h-adaptive refinement process guided by the local error estimation in energy norm, as explained in Section 5.2. Figure 24 shows the first four h-adapted meshes for tri-linear and tri-quadratic elements. In Figure 25 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.



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



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

Figure 24: Front view of the first four meshes of the h-refinement process.

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.

We used a complex model to test the proposed strategy. The model selected represents a perforated screw, as shown in Figure 26, with a topology as used in hydraulic applications. In this case, we restrained the displacements of the surfaces

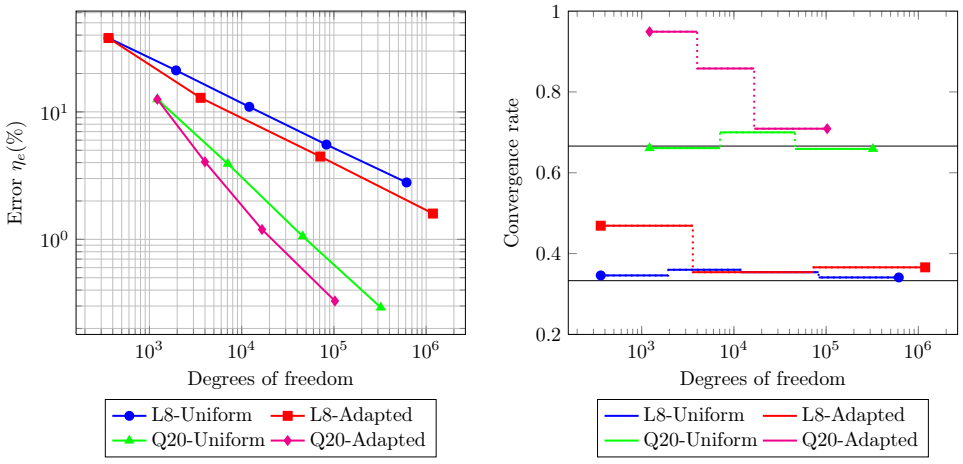
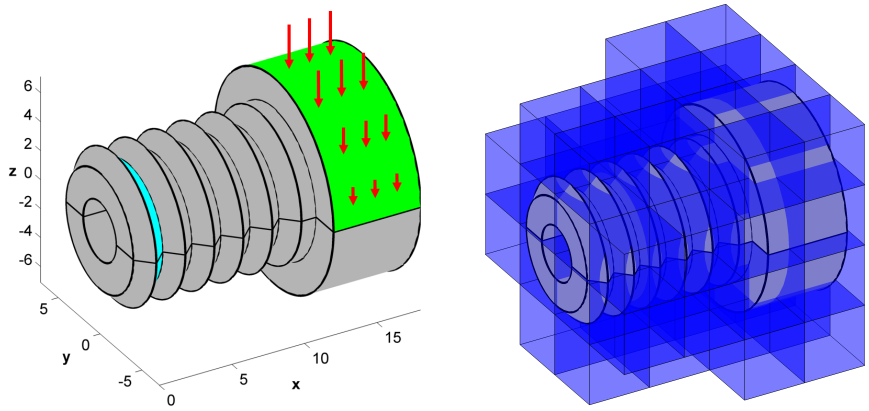


Figure 25: Cylinder convergence study. h-adaptive case. (Left) Relative error in energy norm. (Right) Convergence rate.

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 $\nu = 0,333$.

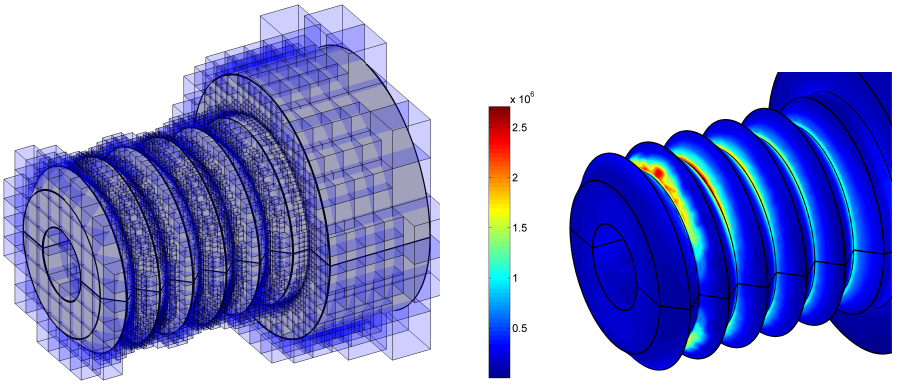


(a) CAD model and boundary conditions.

(b) Initial coarse mesh.

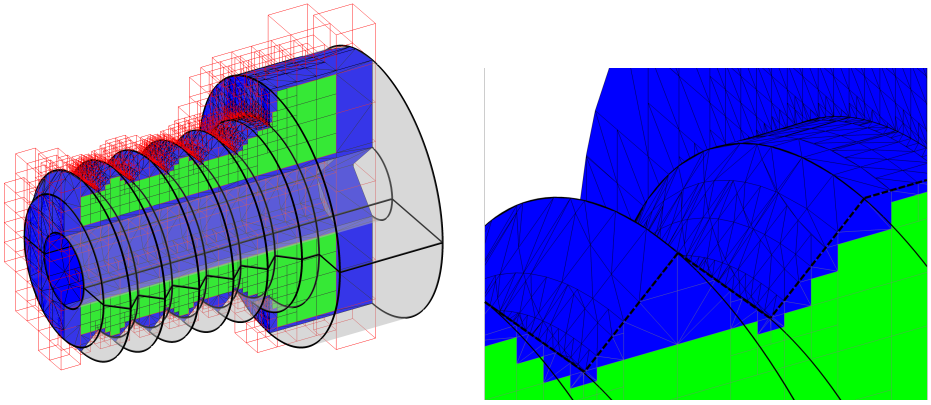
Figure 26: Model of an hydraulic screw.

Figure 26b 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 27a 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 27b represent the Von Mises stress field, where the stress concentration can be seen along the singularities produced by the re-entrant corners of the model.



(a) Approximation mesh.

(b) Von Mises stress field.



(c) Section of the mesh.

(d) Detail of the integration subdomains.

Figure 27: Geometrical h-refinement.

To make clear how boundary elements are treated, Figure 27c 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 27d 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 analysis. The first mesh is the geometrically refined mesh shown in Figure 27d and the second mesh corresponds to the one obtained from the error-based refinement, see Figure 28b. 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

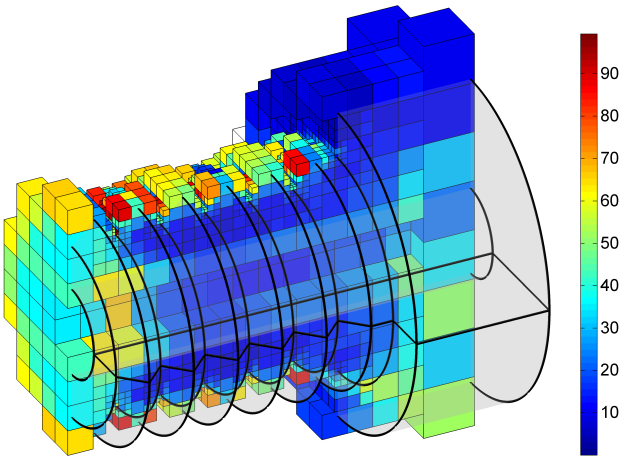
Table 5: Topology of the approximation meshes in terms of different types of elements and subdomains.

The global estimated error in energy norm for the first mesh is 20.86%. Figure 28a 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 28b. We can observe higher density of elements in this mesh compared to the first one analyzed. Figure 28c 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.

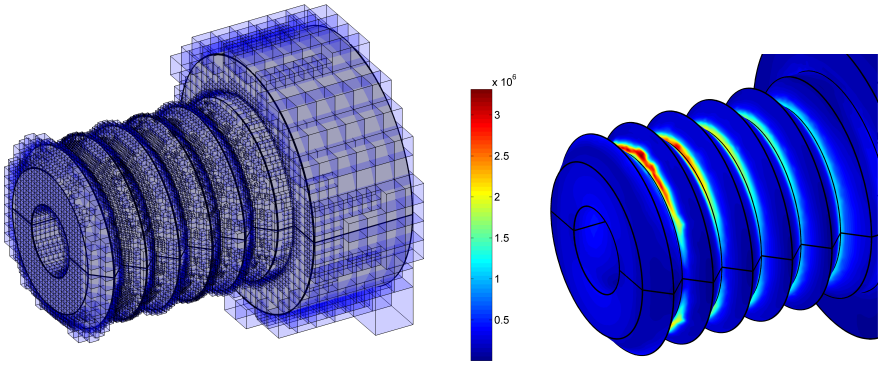
7. Conclusions

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.



(a) Element-wise error estimation.



(b) Approximation mesh.

(c) Von Mises stress field.

Figure 28: Error-based h-refinement.

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 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.

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 [55] 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:

$$\|\mathbf{e}\| \leq C_1 h^{min(p,\lambda)} \approx C_2 N^{-\frac{1}{d} min(p,\lambda)} \tag{9}$$

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[57]:

$$\|\mathbf{e}\| \leq C_2 N^{-\frac{1}{d} c} \tag{10}$$

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 (9), 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,

$$\frac{\|\mathbf{e}\|_n}{\|\mathbf{e}\|_{n-1}} \approx \left[\frac{h_n}{h_{n-1}}\right]^p \tag{11}$$

In the following we assume that the convergence shown in (11) 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:

$$\frac{\|\mathbf{e}^{(i),n-1}\|_n}{\|\mathbf{e}^{(i)}\|_{n-1}} \approx \left[\frac{h_n^{(i),n-1}}{h_{n-1}^{(i)}}\right]^p \tag{12}$$

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:

$$\|\mathbf{e}^{(i),n-1}\|_n^2 = \sum_{i=1}^{M_n^{(i),n-1}} \|\mathbf{e}^{(i)}\|_n^2 \tag{13}$$

We use (12) 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.

$$h_n^{(i),n-1} \approx h_{n-1}^{(i)} \left[\frac{\|\mathbf{e}^{(i),n-1}\|_n}{\|\mathbf{e}^{(i)}\|_{n-1}}\right]^{1/p} \tag{14}$$

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

$$M_n^{(i),n-1} \approx \left(\frac{h_n^{(i)}}{h_n^{(i),n-1}} \right)^d \tag{15}$$

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

$$M_n = \sum_{i=1}^{M_{n-1}} M_n^{(i),n-1} \approx \sum_{i=1}^{M_{n-1}} \left[\frac{h_{n-1}^{(i)}}{h_n^{(i),n-1}} \right]^d \tag{16}$$

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

As assumed in (9), 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

$$M_{n-1} h_{n-1}^d \approx M_n h_n^d \tag{17}$$

Taking into account (11), we can estimate the number of elements in the new mesh as a function of the number of elements in the previous mesh:

$$M_n \approx M_{n-1} \left[\frac{\|\mathbf{e}\|_{n-1}}{\|\mathbf{e}\|_n} \right]^{d/p} \tag{18}$$

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 (18). 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

$$\|\mathbf{e}\|_n^2 = \sum_{i=1}^{M_n} \|\mathbf{e}^{(i)}\|_n^2 = M_n \|\mathbf{e}^{(i)}\|_n^2 \tag{19}$$

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:

$$\|\mathbf{e}^{(i)}\|_n = \left[\frac{1}{M_n} \right]^{1/2} \|\mathbf{e}\|_n \tag{20}$$

Moreover, since we have Equation (18) to estimate the number of elements in the new mesh, we estimate the error in each element of the new mesh as:

$$\|\mathbf{e}^{(i)}\|_n = \left[\frac{1}{M_{n-1}}\right]^{1/2} \frac{\|\mathbf{e}\|_n^{(d/2p)+1}}{\|\mathbf{e}\|_{n-1}^{d/2p}} \tag{21}$$

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:

$$\|\mathbf{e}^{(i),n-1}\|_n^2 = \sum_{i=1}^{M_n^{(i),n-1}} \|\mathbf{e}^{(i)}\|_n^2 = M_n^{(i),n-1}\|\mathbf{e}^{(i)}\|_n^2 \tag{22}$$

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 (15), we obtain the estimated error $\|\mathbf{e}^{(i),n-1}\|_n$ as

$$\|\mathbf{e}^{(i),n-1}\|_n^2 \approx \left[\frac{h_{n-1}^{(i)}}{h_n^{i,n-1}}\right]^d \|\mathbf{e}^{(i)}\|_n^2 \tag{23}$$

Therefore, using Equation (21) we obtain $\|\mathbf{e}^{(i),n-1}\|_n$ by the following expression:

$$\|\mathbf{e}^{(i),n-1}\|_n \approx \left[\frac{h_{n-1}^{(i)}}{h_n^{(i),n-1}}\right]^{d/2} \left[\frac{1}{M_{n-1}}\right]^{1/2} \frac{\|\mathbf{e}\|_n^{(d/2p)+1}}{\|\mathbf{e}\|_{n-1}^{d/2p}} \tag{24}$$

Finally using Equation (14) we obtain the new element size for each of the elements of the previous mesh as

$$h_n^{(i),n-1} \approx h_{n-1}^{(i)} \left[\frac{1}{M_{n-1}}\right]^{1/2(p+1)} \left[\frac{\|\mathbf{e}\|_n}{\|\mathbf{e}\|_{n-1}}\right]^{\frac{d}{2p^2+pd}} \left[\frac{\|\mathbf{e}\|_n}{\|\mathbf{e}^{(i)}\|_{n-1}}\right]^{\frac{2}{2p+d}} \tag{25}$$

where all quantities are well defined.

References

[1] Babuška I, Rheinboldt C. A-posteriori error estimates for the finite element method. *International Journal for Numerical Methods in Engineering* 1978; **12**:1597–1615. 1, 5.2

[2] Babuška I, Rheinboldt C. Adaptive approaches and reliability estimates in finite element analysis. *Computer Methods in Applied Mechanics and Engineering* 1979; **17-18**(3):519–540. 1

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

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

[5] Ainsworth M, Oden JT. *A posteriori Error Estimation in Finite Element Analysis*. John Wiley & Sons, 2000. 1, 5.2

[6] Babuška I, Szabó B, Katz IN. The p-version of the finite element method. *SIAM Journal on Numerical Analysis* 1981; **8**(3):515–545. 2

[7] Dorr MR. The aproximation theory for the p-version of the finite element method. *SIAM Journal on Numerical Analysis* 1984; **21**(6):1180–1207. 2

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

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

[10] Meagher D. Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer. *Technical Report IPL-TR-80-11 I*, Rensselaer Polytechnic Institute 1980. 1

[11] Jackins CL, Tanimoto SL. Oct-tree and their use in representing three-dimensional objects. *Computer Graphics and Image Processing* 1980; **14**(3):249–270. 1

[12] Doctor LJ, Torborg JG. Display techniques for octree-encoded objects. *IEEE Comput. Graph. Appl.* 1981; **1**(3):29–38. 1

[13] Yerry MA, Shephard MS. Automatic three-dimensional mesh generation by the modified-octree technique. *International Journal for Numerical Methods in Engineering* 1984; **20**(11):1965–1990. 1

[14] Shephard MS, Schroeder WJ. A combined Octree/Delaunay method for fully automatic 3D mesh generation. *International Journal for Numerical Methods in Engineering* 1990; **29**(1):37–55. 1

[15] Shephard MS, Georges MK. Automatic three-dimensional mesh generation by the finite octree technique. *International Journal for Numerical Methods in Engineering* 1991; **32**(4):709–749. 1

[16] Peskin CS. Numerical Analysis of Blood Flow in the Heart. *Journal of Computational Physics* 1977; **25**:220–252. 1

[17] Zhang L, Gerstenberger A, Wang X, Liu WK. Immersed Finite Element Method. *Computer Methods in Applied Mechanics and Engineering* 2004; **293**(21):2051–2067. 1

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

[19] Düster A, Parvizian J, Yang Z, Rank E. The finite cell method for three-dimensional problems of solid mechanics. *Computer Methods in Applied Mechanics and Engineering* 2008; **197**(45-48):3768–3782. 1

[20] Schillinger D, Ruess M. 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* 2015; **22**(3):391–455. 1

[21] Haslinger J, Jedelsky D. Genetic algorithms and fictitious domain based approaches in shape optimization. *Struc. Optim.* 1996; **12**:257–264. 1

[22] Kunisch K, Peichl G. Numerical gradients for shape optimization based on embedding domain techniques. *Comput. Optim.* 1996; **18**:95–114. 1

[23] Liu WK, Liu Y, Darell D, Zhang L, Wang XS, Fukui Y, Patankar N, Zhang Y, Bajaj C, Lee J, *et al.*. Immersed Finite Element Method and its Applications to Biological Systems. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**(13):1722–1749. 1

[24] Liu WK, Tang S. Mathematical Foundations of the Immersed Finite Element Method. *Computational Mechanics* 2007; **39**(3):211–222. 1

[25] Gil AJ, Arranz-Carreño A, Bonet J, Hassan O. The Immersed Structural Potential Method for Haemodynamic Applications. *Journal of Computational Physics* 2010; **229**(22):8613–8641. 1

[26] Nadal E, Ródenas JJ, Albelda J, Tur M, Tarancón JE, Fuenmayor FJ. Efficient Finite Element Methodology based on Cartesian Grids: Application to Structural Shape Optimization. *Abstract and Applied Analysis* 2013; **2013**. 1, 5.2

[27] Nadal E. *Cartesian Grid FEM (cgFEM): High Performance h-adaptive FE Analysis with Efficient Error Control. Application to Structural Shape Optimization. PhD Thesis*. Universitat Politècnica de València, 2014. 1

[28] Marco O, Sevilla R, Zhang Y, Ródenas JJ, Tur M. Exact 3D boundary representation in finite element analysis based on Cartesian grids independent of the geometry. *International Journal for Numerical Methods in Engineering* 2015; **103**:445–468. 1, 2, 3, 4, 4, 5.1

[29] Piegl L, Tiller W. *The NURBS Book*. Springer-Verlag, 1995. 1

[30] Rogers DF. *An Introduction to NURBS: with Historical Perspective*. Elsevier, 2001. 1

[31] Sederberg TW, Zheng J, Bakenov A, Nasri A. T-splines and T-NURCCs. *ACM Transactions on Graphics (TOG)* 2003; **22**(3):477–484. 1

[32] Sevilla R, Fernández-Méndez S, Huerta A. NURBS-enhanced Finite Element Method (NEFEM): A Seamless Bridge Between CAD and FEM. *Archives of Computational Methods in Engineering* 2011; **18**(4):441–484. 1, 2, 4

[33] Sevilla R, Fernández-Méndez S, Huerta A. 3D-NURBS-enhanced Finite Element Method (NEFEM). *International Journal for Numerical Methods in Engineering* 2011; **88**(2):103–125. 1, 2

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

[35] Fries TP, Omerović S. Higher-order accurate integration of implicit geometries. *International Journal for Numerical Methods in Engineering* 2016; **106**(5):323–371. 1, 3

[36] Sevilla R, Fernández-Méndez S, Huerta A. Comparison of High-order Curved Finite Elements. *International Journal for Numerical Methods in Engineering* 2011; **87**(8):719–734. 1

[37] Tur M, Albelda J, Marco O, Ródenas JJ. Stabilized Method to Impose Dirichlet Boundary Conditions using a Smooth Stress Field. *Computer Methods in Applied Mechanics and Engineering* 2015; **296**:352–375. 2

[38] Kajiya JT. Ray Tracing Parametric Patches. *SIGGRAPH Comput. Graph.* 1982; **16**(3):245–254. 3

[39] Toth DL. On Ray Tracing Parametric Surfaces. *SIGGRAPH Comput. Graph.* 1985; **19**(3):171–179. 3

[40] Sweeney M, Bartels R. Ray tracing free-form b-spline surfaces. *IEEE Computer Graphics and Applications* 1986; **6**(2):41–49. 3

[41] Nishita T, Sederberg TW, Kakimoto M. Ray Tracing Trimmed Rational Surface Patches. *SIGGRAPH Comput. Graph.* 1990; **24**(4):337–345. 3

[42] Barth W, Stürzlinger W. Efficient ray tracing for Bezier and B-spline surfaces. *Computers & Graphics* 1993; **17**(4):423–430. 3

[43] Lorensen WE, Cline HE. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *ACM SIGGRAPH Computer Graphics* 1987; **21**(4):163–169. 4, 4

[44] Abel JF, Shephard MS. An algorithm for multipoint constraints in finite element analysis. *International Journal for Numerical Methods in Engineering* 1979; **14**(3):464–467. 2

[45] Farhat C, Lacour C, Rixen D. Incorporation of linear multipoint constraints in substructure based iterative solvers. Part 1: a numerically scalable algorithm. *International Journal for Numerical Methods in Engineering* 1998; **43**(6):997–1016. 2

[46] Díez P, Parés N, Huerta A. Recovering lower bounds of the error by postprocessing implicit residual a posteriori error estimates. *International Journal for Numerical Methods in Engineering* 2003; **56**(10):1465–1488. 5.2

[47] Gerasimov T, Rüter M, Stein E. 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* 2012; **90**(April):1118–1155. 5.2

[48] Ladevèze P, Leguillon D. Error estimate procedure in the finite element method and applications. *SIAM Journal on Numerical Analysis* 1983; **20**(3):485–509. 5.2

[49] Almeida Pereirea OJB, Moitinho de Almeida JP, Maunder EAW. Adaptive methods for hybrid equilibrium finite element models. *Computational Methods in Applied Mechanics and Engineering* 1999; **176**:19–39. 5.2

[50] Almeida Pereira OJB, Moitinho de Almeida JP. A posteriori error estimation for equilibrium finite elements in elastostatic problems. *Computer Assisted Mechanics and Engineering Sciences* 2001; **8**(2-3):439–453. 5.2

[51] Zienkiewicz OC, Zhu JZ. A Simple Error Estimator and Adaptive Procedure for Practical Engineering Analysis. *International Journal for Numerical Methods in Engineering* 1987; **24**(2):337–357. 5.2, 5.2

[52] Zienkiewicz OC, Zhu JZ. The superconvergent patch recovery and a posteriori error estimates. Part 1: The recovery technique. *International Journal for Numerical Methods in Engineering* 1992; **33**(7):1331–1364. 5.2

[53] Zienkiewicz OC, Zhu JZ. The superconvergent patch recovery and a posteriori error estimates. Part 2: Error estimates and adaptivity. *International Journal for Numerical Methods in Engineering* 1992; **33**(7):1365–1382. 5.2

[54] Ródenas JJ, Tur M, Fuenmayor FJ, Vercher A. Improvement of the superconvergent patch recovery technique by the use of constraint equations: the SPR-C technique. *International Journal for Numerical Methods in Engineering* 2007; **70**(6):705–727. 5.2

[55] Fuenmayor FJ, Oliver JL. Criteria to achieve nearly optimal meshes in the h-adaptive finite element mehod. *International Journal for Numerical Methods in Engineering* 1996; **39**(23):4039–4061. 5.2, A

[56] Zienkiewicz OC, Taylor RL, Zhu JZ ((eds.)). *The Finite Element Method: Its Basis and Fundamentals*. Butterworth-Heinemann: Oxford, 2013. 6.1

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

PAPER D

An extension of shape sensitivity analysis to an Immersed Boundary Method based on Cartesian grids

O. Marco, J. J. Ródenas, F. J. Fuenmayor and M. Tur

Preprint submitted to Computational Mechanics

Abstract

Gradient-based shape optimization processes of mechanical components require that the information of the gradients (sensitivity) of the magnitudes of interest be calculated with sufficient accuracy. Given the potential benefits on computational efficiency provided by the Finite Element Method (FEM) based on the use of Cartesian grids, the aim of this study was to develop algorithms for the calculation of shape sensitivities considering geometric representation by parametric surfaces (i.e. NURBS or T-splines) using 3D Cartesian h-adapted meshes independent of geometry.

A formulation of shape sensitivities was developed for an environment based on Cartesian meshes independent of geometry, which implies, for instance, the need to take into account the special treatment of boundary conditions imposed in non body-fitted meshes. The immersed boundary framework required to implement new methods of velocity field generation, which have a primary role in the integration of both the theoretical concepts and the discretization tools in shape design optimization.

Examples of elastic problems with three-dimensional components are given to demonstrate the efficiency of the algorithms.

Key words

Cartesian Grid-FEM; sensitivity analysis; velocity field; NURBS

Contents

1 Introduction **225**

2 Design velocity fields **228**
2.1 Generation of boundary velocity fields 230
2.1.1 Velocity field on trimmed surfaces 232
2.2 Generation of domain velocity fields 236
2.2.1 Least squares approach 239
2.2.2 Physical approach 240

3 Calculating shape sensitivities with FEAVox **241**
3.1 Evaluation of derivatives 243

4 Numerical examples **245**
4.1 Thick wall infinite cylinder under internal pressure 246
4.2 Sequence of collinear cracks in an infinite plate 248

5 Conclusions **254**

Bibliography **256**

1. Introduction

In optimal structural design, sensitivity analysis is the calculation of the derivatives of structural response (displacements, stresses, natural frequencies, etc.) with respect to design variables. The initial development of sensitivity analysis focused on the size design variables, i.e. thickness or cross-sectional areas of structural components, etc. In many structural problems it is necessary to consider shape as a design variable. This is particularly important in the optimal design of structural components. This paper will focus on the analysis of the sensitivities of the design variables that describe the geometry of the component to be optimized.

A large number of references have been published in the field of sensitivity analysis in shape design, especially during the 90s. Currently, the research efforts in the topic are focused on new implementations of the well-known approaches and their application to new problems. In a brief overview, four different approaches can be distinguished:

1. Global finite differences [1, 2, 3, 4]: finite difference expressions are used to obtain the derivatives from the output of repeated Finite Element Analysis (FEA) when small perturbations of the design variables are introduced.

2. Continuum approach. [5, 6, 7, 8, 9, 10, 11]. Derivatives are obtained differentiating the governing elasticity equations. For shape design variables, the two main approaches are the material derivative and the control volume approach. These relate changes in the geometrical shape with the structural characteristics leading to a set of continuum sensitivity equations that are then discretized and solved.

3. Discrete approach [12, 13, 14, 15]. The procedure derivation-to-discretization is reversed and the components of the discretized system of equations are differentiated with respect to the design variables.

4. Computational differentiation [16, 17] is related to the automatic differentiation of the routines within the computational code[18, 19, 20].

The different approaches can be evaluated from the point of view of accuracy, their relation to discretization and cost in computational and implementation terms. Their relationships and comparisons can be found in [21, 22, 23, 24, 25].

Gradient-based optimization requires the evaluation of the sensitivities to drive the optimization process. In shape optimization, this involves adapting or regenerating the Finite Element (FE) mesh for the different geometries to run the numerical simulation of each of these geometries. Reference [26] showed that the behavior of the optimization algorithm is strongly influenced by the accuracy of the results used

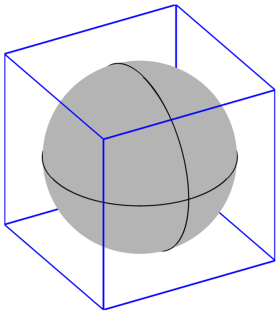
to drive the process (objective function, constraints and their derivatives). Any inaccuracy in these results can pollute the behavior of the optimization algorithm and reduce the convergence rate to the optimal solution, induce the convergence to a non-optimal or unfeasible solution or even prevent convergence. This requires high-quality FE analyses that can involve a considerable computational cost for each geometry and hence of the overall optimization process. The analysis cost of each geometry can be partially alleviated by the use of adaptive analysis techniques, which are intended to provide the optimal cost-effective FE models to obtain numerical results of the prescribed accuracy. Numerous methods of alleviating the mesh burden are reviewed in [27].

One of the options is the immersed boundary approach[28, 29, 30], which is a natural platform for structural shape optimization processes because its properties are suited to simplifying the mesh generation stage. Immersed Boundary Methods (IBM) have been studied by a number of authors for a wide range of problems such as shape optimization[31, 32] or bio-mechanics, see for instance[33, 34]. An example of this type of approach can be seen in Figure 1. The geometrically complex domain, $\Omega_{\texttt{Phys}}$, is embedded into a geometrically simpler domain, Ω, see Figure 1a. The embedding domain is often simply a cube (or a rectangular cuboid in general) that can be effortlessly discretized using a Cartesian mesh made out of hexahedra to create what we call an approximation mesh, $\Omega_{\texttt{Approx}}$, see Figure 1b. During the integration step, only the internal elements and the internal part of the elements cut by the boundary will be considered (Figure 1c).

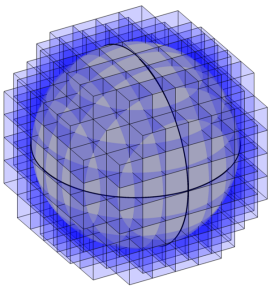
Another approach is to improve the geometrical accuracy of the models by integrating CAD representations with the FEM codes. Isogeometric Analysis (IGA)[35, 36] is a recent trend in this direction. The main idea is that the meshing procedure is circumvented since an existing CAD geometry is directly used for analysis, all the while keeping the exact geometry. However, in its finite element form, generating an analysis-suitable solid discretization is an open topic[37, 38, 39]. In order to bypass the internal domain parametrization, Boundary Element Methods have also been used in shape optimization [40, 41]. Studies on IGA sensitivity analysis can be found in [42, 43, 44].

The NURBS-enhanced Finite Element Method (NEFEM) [45, 46] employs NURBS for the geometric representation of the boundary, whilst maintaining the flexibility of FEM by using polynomial interpolation. NEFEM conveniently merges the accurate representation of the geometry alleviating the difficulty of generating interior isogeometric elements. However, mesh generation in NEFEM presents a difficulty similar to that found in FEM.

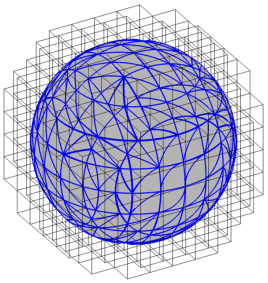
In [47, 48] we introduced cgFEM (Cartesian Grid FEM) as an alternative to solve these drawbacks. The 3D version of this methodology, based on the use of Cartesian grids independent of the geometry, was implemented in a computer code, named FEAVox [49], for the structural analysis of components. cgFEM is distinguished from other immersed approaches by considering the exact CAD representation of the



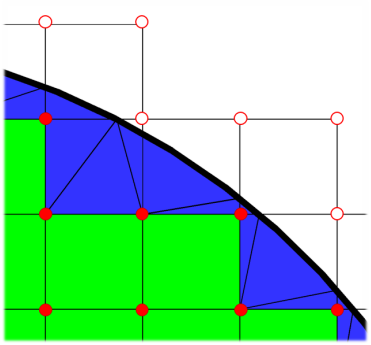
(a) Model Ω_{Phys} within the embedding domain, Ω.



(b) Discretization $\Omega_{\texttt{Approx}}$ of the embedding domain.



(c) Integration domain, Ω^h_{Approx}.



(d) Detail of a section.

Figure 1: Typical Immersed Boundary Method environment.

boundary of the domain by means of the use of the NURBS-enhanced integration techniques [46] to perform the numerical integration over the true computational domain.

In the approach proposed here, the properties related to all the immersed methods (automatic domain discretization, creation of hierarchical data structures for simple data transfer and re-use of calculations, etc.) are merged with the ability to consider the exact geometrical representation, instead of simplifying the embedded boundary (for instance using triangular facets for its definition).

From this point of view, we propose the implementation of sensitivity analysis in cgFEM, exploiting the features of our embedded methodology, aiming for the efficient calculation of sensitivities to reduce the computational cost of the optimization process. The strategy proposed in this paper belongs to the group of techniques denoted as *discrete semi-analytical*. This specification, in our case, means that some of the

discrete derivatives rely on analytical derivation and some on finite difference approximations. We use finite differences in order to differentiate the nodal locations with respect to design variables, the so-called velocity field, which is a challenging issue considering the immersed nature of the cgFEM.

The paper is organized as follows: the generation of velocity fields, both on the boundary and inside the domain, is addressed in Section 2. The formulation of shape sensitivity analysis using an immersed boundary approach is described in Section 3. The numerical results showing the performance of the proposed technique are given in Section 4 and the conclusions are reported in Section 5.

2. Design velocity fields

Sensitivity analysis is intended to find the change in the magnitude of response (displacements, stresses, etc.) with respect to design variables. In the case of shape design problems, the position of the material points depends on the design variables. Defining $\mathbf{a}$ as the vector of design variables, the position of an arbitrary point of the domain will be a function of the form $\mathbf{p} = \mathbf{p}(\mathbf{a})$.

In a previous step, the evaluation of shape sensitivities defines how to vary the position of material points of the domain in relation to the design variables, i.e. the sensitivity of the coordinates of the material points, usually called velocity fields, which for an arbitrary design variable a_m is defined as:

$$\mathbf{V}_m = \frac{\partial \mathbf{p}}{\partial a_m} \tag{1}$$

The quality of the velocity field influences the accuracy of the numerical solution, which determines the effective convergence rate of the gradient-based optimization algorithms. Nevertheless, while the numerical solution and sensitivity analysis are defined on the whole domain Ω_{Phys}, the velocity fields are defined only on the boundary of the domain and there is no closed conformation of this field in the interior.

The determination of the velocity fields is based on the theoretical features of the sensitivity expressions and practical requirements obtained from the features of the FE solution [50, 51]. Theoretically, the velocity field should have the same regularity as the displacements field and depend linearly on the alteration of the design variables. In practical terms, different applications can also impose certain practical additional requirements on the velocity field, such as the need to maintain the mesh topology, to provide FE nodes necessarily located on the boundary of the domain, to produce non-distorted meshes, to be naturally related with the design parameters of the CAD models or to be efficient and general.

Magnitudes like the sensitivity of the strain energy are not affected by the values of the velocity fields in the interior of the domain, provided the velocity fields meet the theoretical requirements and the exact structural response is used in the evaluation of this sensitivity. However, in practice, the FE approximation will be used instead of the exact structural response. As a consequence of this, the final sensitivity of the strain energy will be affected by the velocity fields considered in the interior of the domain [52]. In fact, the stability of the sensitivity of the strain energy can be used to assess the quality of the different techniques that can be used to define the design velocity field.

As mentioned above, the design velocity fields must be defined in the whole domain. To do this, the velocity field is usually defined at the nodes of the FE mesh and then interpolated using the shape functions used to interpolate the displacements, so that the velocity fields and the displacements field will have equivalent regularity. In the following subsections we will first describe a procedure to define the design velocity fields along the boundary of the domain, followed by the procedure to define it inside the domain. During the definition of the design velocity fields we will consider the special characteristics of cgFEM aimed at the development of an efficient procedure for shape sensitivity analysis.

Some of the methods found in the literature for velocity field definition are following:

- Finite Difference (FD) method[53]. This method defines the parametric nodal positions on the boundary and evaluates the change of the position due to a perturbation of the design variables using an FD scheme. After that, an interpolation technique has to be used to give the values of the velocity field to the internal nodes.

- Structured meshes[54]. These methods are based on the rules to generate structured meshes that provide a formulation for the position of internal nodes as a function of the boundary nodes.

- Boundary elements method[55]. This strategy considers a null velocity field on the domain except in the subdomain defined by the boundary elements. This reduces the shape sensitivity calculations to a small fraction of the domain.

- Physical approach[56]. This approach defines the evaluation of the velocity field as an equivalent linear elasticity problem where the velocity field evaluated on the boundary is considered as displacements applied on the boundary. Solving the elasticity problem will provide the displacements in the interior of the domain that will be interpreted as velocity field.

- Laplacian method[57]. As in the previous case, the velocity field is considered analogous to a displacement field. The velocity field on the boundary is considered as a perturbation (displacement) of the boundary. Laplacian smoothing is

then used to improve the nodal positions. The final displacements of the nodes will be considered as the velocity field at each node.

- Domain triangulation method[52]. This method uses an initial step of the Delaunay triangulation procedure, where the nodes of the triangulation are only placed on the boundary, to interpolate the velocity field through the domain.

Comparative studies of some of these methods can be found in [50, 51, 58, 59].

Due to the nature of IBM, the above-described algorithms cannot be used directly in an IBM context. In this paper we will therefore discuss some alternatives that can be used to generate adequate velocity fields for a Cartesian grid framework, taking into account the features of embedded methods.

2.1. Generation of boundary velocity fields

NURBS (Non-Uniform Rational B-Spline) curves and surfaces [60, 61] were used in the present study to describe the boundary of 2D and 3D domains. Existing works in the literature show the use of NURBS for sensitivity analysis have been used both as an analysis tool, i.e. Isogeometric Analysis[42, 62] and for the geometric description of the models[63, 64, 65].

A rational B-spline curve is given by

$$\mathbf{C}(\lambda) = \frac{\sum_{i=1}^{n} N_i^{(p)}(\lambda)\, w_i \mathbf{P}_i}{\sum_{i=1}^{n} w_i N_i^{(p)}(\lambda)} \tag{2}$$

Here $\mathbf{P}_i$ are the n control points given in d-dimensional space $\mathbb{R}^d$, w_i is referred to as the i-th weight, typically $w_i \geq 0 \; \forall \; i$ and $N_i^{(p)}(\lambda)$ are normalized B-spline basis functions of order p, which are defined recursively as

$$N_i^{(0)}(\lambda) = \begin{cases} 1 & \lambda_i \leq \lambda \leq \lambda_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$N_i^{(q)}(\lambda) = \frac{(\lambda - \lambda_i)\, N_i^{(q-1)}(\lambda)}{\lambda_{i+q} - \lambda_i} + \frac{(\lambda_{i+q+1} - \lambda)\, N_{\lambda+1}^{(q-1)}(\lambda)}{\lambda_{i+q+1} - \lambda_{i+1}} \tag{4}$$

for $q = 1, \ldots, p$ and $i = 1, \ldots, n$, where λ are the knots, which are assumed ordered $0 \leq \lambda_i \leq \lambda_{i+1} \leq 1$, forming the so-called knot vector

$$\Lambda = \{\lambda_1, \ldots, \lambda_{n+p+1}\}, \tag{5}$$

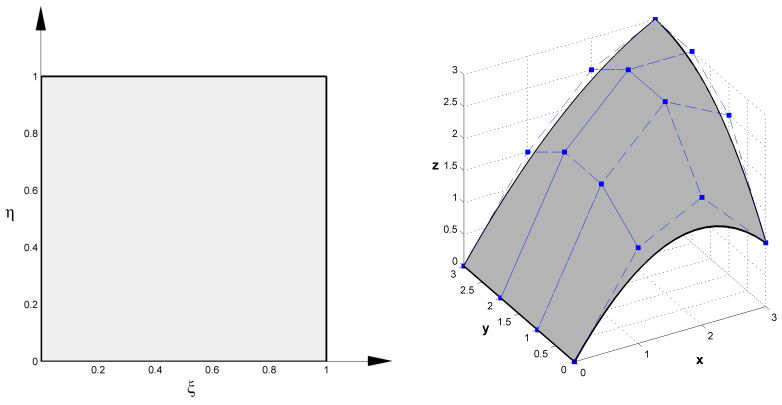
which uniquely describes the B-spline basis functions. The multiplicity of a knot, i.e. the number of times it is repeated in the knot vector, determines the decrease in the

number of continuous derivatives, so they are $\mathcal{C}^{p-1}$-continuous where the knots are not repeated. If a knot has multiplicity k, the basis is $\mathcal{C}^{p-k}$-continuous at that knot. Other properties of the basis functions can be found in [60, 61].

NURBS surfaces are obtained 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. 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

$$\mathbf{S}(\xi, \eta) = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{N_i^{(p)}(\xi) M_j^{(q)}(\eta) w_{i,j} \mathbf{P}_{i,j}}{\sum_{i=1}^{n} \sum_{j=1}^{m} N_i^{(p)}(\xi) M_j^{(q)}(\eta) w_{i,j}} \tag{6}$$

An example of a NURBS surface is represented in Figure 2 with the corresponding control net.



(a) Parametric space. (b) Surface and control net.

Figure 2: NURBS surface example.

In our approach, we define the design variables $\mathbf{a}$ that modify the control points $\mathbf{P}(\mathbf{a})$. Therefore, for a specific design variable a_m the calculation of the velocity field on the parametrized boundary $\mathbf{S}(\xi, \eta, \mathbf{a})$, is simple and can be expressed as:

$$\mathbf{V}_{m,\Gamma}(x(\xi,\eta), y(\xi,\eta), z(\xi,\eta)) = \frac{\partial \mathbf{S}(\xi, \eta, \mathbf{a})}{\partial a_m} = f\left(\frac{\partial \mathbf{P}(\mathbf{a})}{\partial a_m}\right) \tag{7}$$

The boundary velocity field on the discrete model is achieved by the evaluation of (7) using the parametric coordinates (ξ, η) of each surface point. The analytical evaluation of the derivatives of the NURBS and trimmed NURBS (see Subsection

2.1.1) can be cumbersome and will depend on how each CAD system generates the surfaces as a function of the parameters defined by the user. Therefore, for the sake of generality, we propose to approximate the analytical evaluation of these derivatives by a finite differences approximation as shown in the following equation:

$$\mathbf{V}_{m,\Gamma} \cong \frac{\Delta \mathbf{S}(\xi, \eta, \mathbf{a})}{\Delta a_m} = \frac{\mathbf{S}(\xi, \eta, \mathbf{a} + \Delta a_m) - \mathbf{S}(\xi, \eta, \mathbf{a})}{\Delta a_m} \tag{8}$$

where Δa_m is a perturbation of the design variable a_m.

In practical terms, the evaluation of the shape sensitivities will require the information of the design velocity field at certain points on the boundary. These points are: a) the points used for the numerical integration of boundary integrals, and b) the points of intersection of the NURBS with the edges of the elements, i.e. with the Cartesian axes that define the mesh. Equation (8) will be used to obtain the velocity field at the parametric coordinates (ξ, η) of these points.

2.1.1. Velocity field on trimmed surfaces

NURBS surfaces are inherently four-sided patches that do not allow for the presence of holes nor the direct creation of irregular shapes. Due to the limitation of a strict rectangular topology, trimming is a valuable procedure to devise complex objects. A trimmed NURBS surface consists of: a) a tensor product NURBS surface and, b) a set of properly arranged trimming curves lying within the parametric rectangle of the surface. The trimming curves can be of any form but, when dealing with NURBS entities, it is useful to represent them in NURBS form.

Assume that n_c NURBS curves are given defined as:

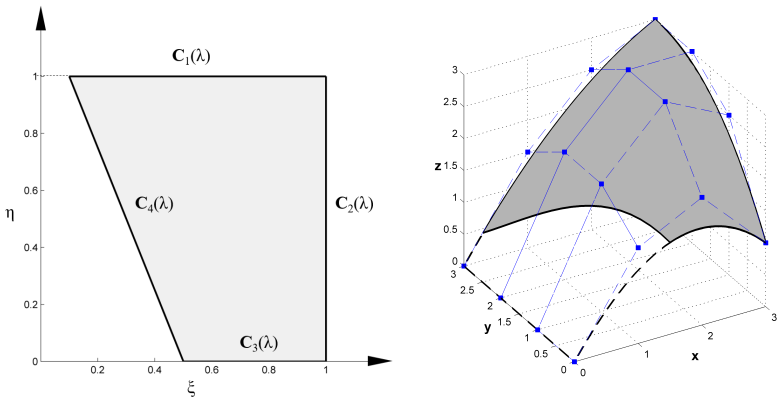
$$\mathbf{C}_k(\lambda) = (\xi_k(\lambda), \eta_k(\lambda)) \qquad k = 1, 2, \ldots, n_c \tag{9}$$

The curves $\mathbf{C}_k(\lambda)$ are all properly oriented forming loops. A loop establishes the boundary of the trimmed region such that, when advancing along the piecewise curve as indicated by its numbering, the real surface material is always on the same side, see Figure 3a. The trimmed surface boundaries are then retrieved by mapping the 2D trimming curves onto the surface. That is,

$$\mathbf{S}(\xi_k(\lambda), \eta_k(\lambda)) \qquad k = 1, 2, \ldots, n_c \tag{10}$$

are surface curves bounding the trimmed surface. Figure 3b shows the 3D mapping of the trimming loop.

Regarding the evaluation of velocity fields in geometries including trimmed NURBS, the trimming procedure in general does not allow application of the previously explained procedure to evaluate the boundary velocity field. The reason for this is that, when dealing with trimmed entities, in some cases the generation of new geometries is obtained by modifying the trimming curve into the parametric space but not the



(a) Trimmed parametric space. (b) Trimmed physical surface.

Figure 3: NURBS surface example.

parametric space itself, i.e. without modifying the control points of the surfaces, leading to $V_{m,\Gamma} = 0$ in (7).

Let us consider a trimmed NURBS surface defined as:

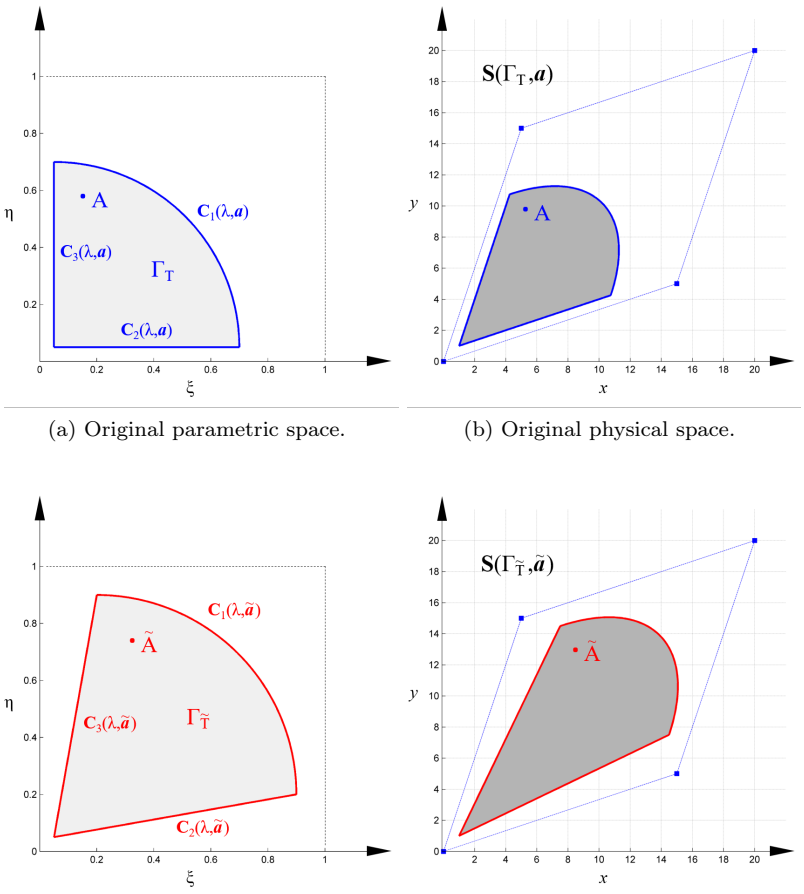
$$\mathbf{S}\left(\mathbf{C}_k\left(\lambda, \mathbf{a}\right), \mathbf{a}\right) = \mathbf{S}\left(\xi_k\left(\lambda, \mathbf{a}\right), \eta_k\left(\lambda, \mathbf{a}\right), \mathbf{a}\right) \qquad k = 1, \ldots, n_c \tag{11}$$

where $\mathbf{C}$ are NURBS curves and $\left(\xi_k\left(\lambda, \mathbf{a}\right), \eta_k\left(\lambda, \mathbf{a}\right)\right)$ are the surface parametric coordinates of the k trimming curve function of the parameter λ and the design variable vector, $\mathbf{a}$. In this definition, we assume that $\mathbf{a}$ can influence both the surface $\mathbf{S}$ and the trimming curves $\mathbf{C}$.

Figure 4a contains the representation of a surface as defined in (11) with the trimmed curves and the parametric subspace Γ_T bounded by them. In Figure 4b, the mapping of the parametric subspace to the physical space is shown along with the diamond-shaped control point polygon.

Now let us assume a change in the design variables such that $\widetilde{\mathbf{a}} = \mathbf{a} + \Delta a_m$, where Δa_m is a small increment in a single design variable. Figure 4c shows an illustration of this change in the parametric space Γ_T, that leads to the new domain $\Gamma_{\widetilde{T}}$. Figure 4d confirms how a change in the trimming loop yields a different mapping of the subspace, while keeping the control polygon in place. However, it is still necessary to evaluate the value of the velocity field for the points in the domain represented in Figure 4a.

Let A in Figures 4a and 4b be a point of interest of coordinates $(\xi, \eta)_A$ in the parametric space and $(x, y)_A$ in the physical space. The perturbation of the design variable a_m will modify the trimmed surface in the parametric space and hence in



(a) Original parametric space.

(b) Original physical space.

(c) Perturbed parametric space.

(d) Perturbed physical space.

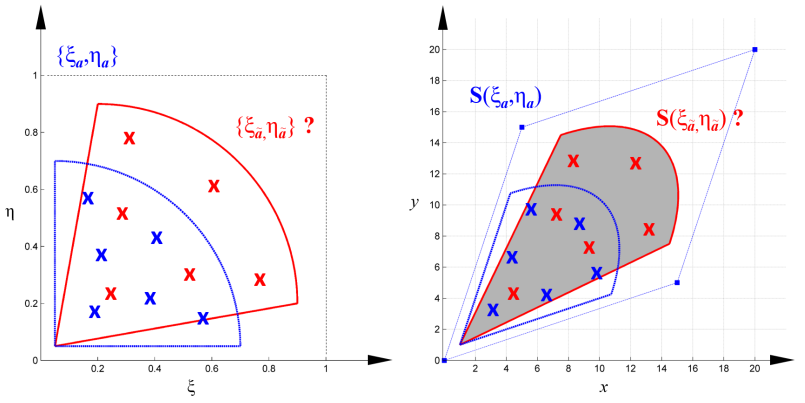
Figure 4: Trimmed NURBS surface example. Modifying trimming curves.

the physical space. This will perturb the position of A to $\widetilde{A}$ (see Figures 4c and 4d) of coordinates $(\xi,\eta)_{\widetilde{A}}$ in the parametric space and $(x,y)_{\widetilde{A}}$ in the physical space. The evaluation of the design velocity field at A will require the evaluation of this perturbation in the physical space. We therefore need to find how $(\xi,\eta)_A$ is mapped to $(\xi,\eta)_{\widetilde{A}}$. This can be evaluated on the trimming curves $\mathbf{C}_k(\lambda,\mathbf{a})$ but we also need this information in the interior of Γ_T.

The velocity field on the trimming curve $\mathbf{C}_k$ boundary for this particular problem can be written as:

$$\mathbf{V}_{m,\mathbf{C}} = \frac{\partial \mathbf{S}\left(\mathbf{C}_k\left(\lambda, \mathbf{a}\right)\right)}{\partial a_m} \tag{12}$$

Equation (12) will evaluate the velocity field only on the trimming curves. This means that the parametric coordinates of the points (ξ, η) on Γ_T will have to be updated to consider the change of the parametric subspace that leads to $\Gamma_{\widetilde{T}}$. Figure 5 shows a general case in which a number of points of interest on the surface, defined by their original parametric coordinates $\{\xi_a, \eta_a\}$), should be updated to new coordinates $\{\xi_{\widetilde{a}}, \eta_{\widetilde{a}}\}$ to obtain a transformation consistent with the trimming curves.



(a) Parametric space. (b) Physical space.

Figure 5: Problem transforming points within a trimmed NURBS surface.

We adapted the idea of the physical approach [56] to obtain this update from $\{\xi_a, \eta_a\}$ to $\{\xi_{\widetilde{a}}, \eta_{\widetilde{a}}\}$. Hence, we propose solving an auxiliary elasticity problem with imposed displacements on the boundary. It is possible to create a 2D finite element system in the original parametric space from information that is already at our disposal. In this case the intersections between the surface with the Cartesian axes would be the nodes and the elements would be defined by the faces of the integration subdomains on the surface (see [49]). Figure 6a shows this proposal. The discretized system of equations of the auxiliary problem can be written as:

$$\mathbf{KP} = \mathbf{F} \tag{13}$$

where $\mathbf{K}$ is the stiffness matrix, $\mathbf{F}$ is the vector of equivalent nodal forces. In this case $\mathbf{F} = 0$ as Neumann boundary conditions are not applied, and $\mathbf{P}$ contains the

prescribed displacements on the boundary and the unknown values of the field in the interior of the domain. These 'prescribed displacements' will correspond to the values of the change on the trimming curves coordinates such that:

$$\mathbf{P}_{m,k} = \left(\xi_k\left(\lambda, \mathbf{a} + \Delta a_m\right), \eta_k\left(\lambda, \mathbf{a} + \Delta a_m\right)\right) - \left(\xi_k\left(\lambda, \mathbf{a}\right), \eta_k\left(\lambda, \mathbf{a}\right)\right) \qquad k = 1, \ldots, n_c \tag{14}$$

In this way the displacements imposed are those that change the position of the trimming curves in the parametric space associated with the design variable under study.

Solving (13) after applying the Dirichlet boundary conditions provides the perturbation of the position of all the nodes of the mesh shown in Figure 6b, which can be interpolated into the elements. The result will be the position of the original points mapped into the new subspace defined by the perturbed boundary in the parametric space:

$$\left\{\widetilde{\xi}, \widetilde{\eta}\right\} = \left\{\xi_{\mathbf{a}+\Delta a_m}, \eta_{\mathbf{a}+\Delta a_m}\right\} = \left\{\xi_{\mathbf{a}}, \eta_{\mathbf{a}}\right\} + \mathbf{P}_m\left(\xi_{\mathbf{a}}, \eta_{\mathbf{a}}\right) \tag{15}$$

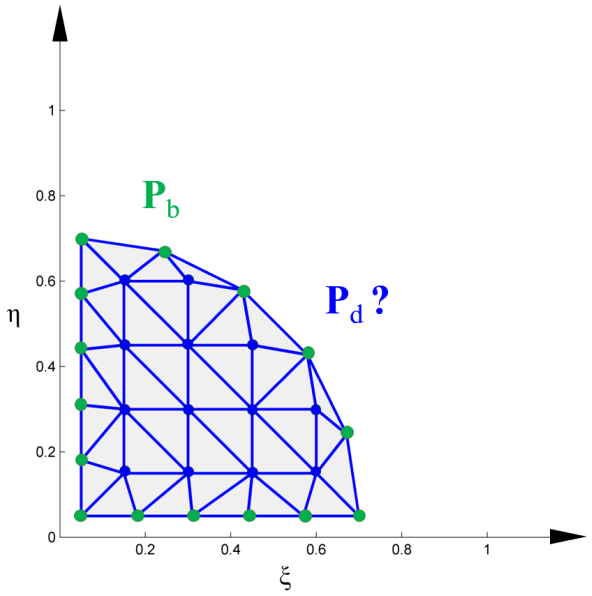
The velocity field on these surfaces will be calculated as:

$$\mathbf{V}_{m,\Gamma} \cong \frac{\Delta \mathbf{S}(\xi, \eta, \mathbf{a})}{\Delta a_m} = \frac{\mathbf{S}(\widetilde{\xi}, \widetilde{\eta}, \mathbf{a} + \Delta a_m) - \mathbf{S}(\xi, \eta, \mathbf{a})}{\Delta a_m} \tag{16}$$

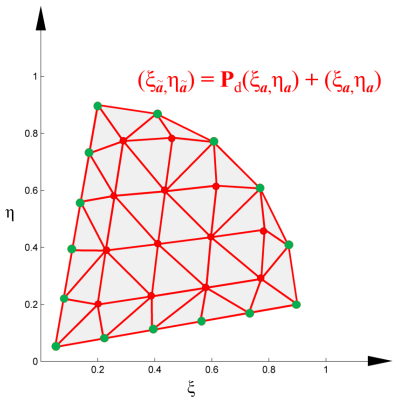
To our knowledge, none of the previous works in the bibliography are related to the evaluation of the design velocity field for trimmed surfaces. The procedure proposed to evaluate the design velocity field for these surfaces involves solving a 2D FE problem. However, the associated computational cost is low, as: a) the FE mesh used for the analysis is that of a previously evaluated triangulation of the trimmed surface in the parametric space required for intersecting the surface with the Cartesian mesh, b) the mesh is a relatively coarse 2D mesh, thus involving a low computational cost and c) the factorization of $\mathbf{K}$ obtained during the process is common to all the design variables.

2.2. Generation of domain velocity fields

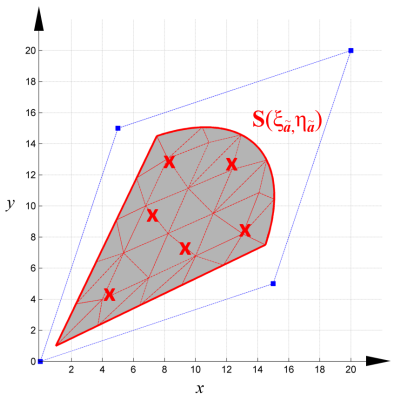
After describing the method of evaluating the design velocity field at any point on the boundary of the domain, this section deals with the methods used to obtain the velocity fields in the domain of the models from the boundary values. As explained above, due to the fixed Cartesian configuration of the meshes used in FEAVox, standard velocity field generation techniques cannot be used directly. Figure 7a uses a 2D case to show that, using fixed Cartesian grids, it is possible to find nodes external to the domain (green dots) that will be involved in the evaluation of the design velocity



(a) FEM system using the parametric space of a trimmed NURBS surface.



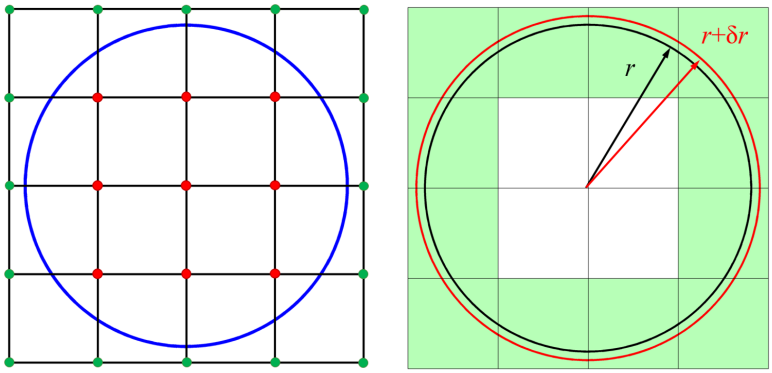
(b) Solution in the parametric space.



(c) Solution in the physical space.

Figure 6: NURBS surface example solved using the proposed strategy.

field. Strategies are needed to assign the velocity field both to internal and external nodes so that we can interpolate the velocity field at any point on the elements.



(a) Existence of external nodes.

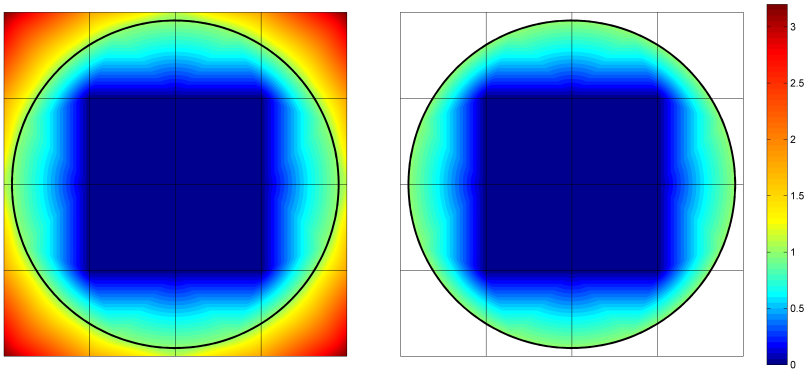
(b) Perturbation of geometry only in boundary elements.

Figure 7: Embedded methods and velocity fields.

It should be noted that a perturbation of the boundary will not induce a perturbation of the Cartesian nodes internal to the surface. This is an important feature imposed by the use of Cartesian grids, since the velocity field will be zero in the internal elements, thus reducing the computational cost associated with the evaluation of their volume integrals for shape sensitivity analysis, e.g. the non-shaded elements in Figure 7b. However, we still have to evaluate the velocity field on the external nodes to produce an interpolated velocity field on the surface of the domain that cuts the boundary elements (shaded elements in Figure 7b) equal to the prescribed one.

We propose two different velocity field generators that will represent the geometry changes only in these elements. Both methods can be classified as boundary element methods [55], as the velocity fields will be non-zero only on the band of elements intersected by the parametrized boundary. At these elements, the velocity field on the boundary will be extended to the internal and external nodes on this layer, considering an FE nodal interpolation with the regularity of the displacements field. In the first method we propose the use of a least squares approach for this extension, whereas the second method will be based on the physical approach [56] used in standard FEM.

Figure 8 shows an example of the velocity field for the example in Figure 7 that would be obtained by the proposed methods using the radius as design variable. Figure 8a shows the interpolation of the velocity field even in the external nodes, while Figure 8b represents the actual velocity field necessary to evaluate the integrals of the sensitivity analysis only of the internal elements and the integration subdomains of the boundary elements.



(a) Velocity field including the values of external nodes.

(b) Effective velocity field.

Figure 8: Representation of a velocity field with the proposed strategies.

2.2.1. Least squares approach

In this first method we use a least squares procedure to extrapolate the values to the external nodes imposing the velocity field on the boundary and the zero velocities on the internal nodes of the elements of the boundary of the layer.

An FE nodal interpolation for each component of the design velocity field is fitted into each element with the velocity field values at the surface integration points of the elements and to $\mathbf{V} = 0$ at the internal nodes of the boundary elements. By using a least square approach we obtain the linear system of equations:

$$\mathbf{M}\mathbf{V}_{m,q} = \mathbf{G}_{m,q} \qquad q = x, y \text{ and } z \tag{17}$$

The system matrix $\mathbf{M}$ is obtained by the assembly of the mass matrix-type array of each element along the boundary. The global mass matrix is given by:

$$\mathbf{M} = \sum^{n_e} \int_{\Gamma_D^e} \mathbf{N}^T \mathbf{N} |\mathbf{J}| \mathrm{d}\Gamma \tag{18}$$

where

Γ_D^e is the portion of the boundary within the element,

$\mathbf{N}$ corresponds to the matrix of finite element interpolation functions.

On the other side of the equation, the vector $\mathbf{G}_{m,q}$ is evaluated by adding the contribution of elements:

$$\mathbf{G}_{m,q} = \sum^{n_e} \int_{\Gamma_D^e} \mathbf{N}^T \mathbf{V}_{m,q}^e |\mathbf{J}| \mathrm{d}\Gamma \qquad q = x, y, z \tag{19}$$

with $\mathbf{V}_{m,q}^e$ as the q^{th} component of the velocity field on the boundary related to the design variable m within each element. Note that this is a low-cost procedure as it only involves the elements along the boundary, which is of interest for 3D domains.

2.2.2. Physical approach

This method consists of solving a linear elasticity problem in which the velocity field on the boundary is considered as the displacements applied on the boundary. This auxiliary problem will have, for example, the following characteristics:

- The body $\Omega_{\texttt{Phys}}$ is characterized with a linear elastic material with Young modulus equal to one and zero Poisson ratio;
- The discretization used to evaluate the design velocity field is the discretization used to evaluate the displacements;
- Every single shape design variable gives a non-zero velocity field on the elements cut by the boundary and zero velocity on the rest of the domain, which ensures the equilibrium of each auxiliary problem. The unknowns are the velocities for all external nodes of the actual FE mesh $\Omega_{\texttt{Approx}}$.

Even though this method needs the resolution of a system of equations as large as the original problem, the associated computational cost is reduced, as: a) the FE mesh used for the analysis is the same Cartesian mesh used for the sensitivity analysis, b) we can remove the internal nodes from the system since the velocity field is set to 0 on these nodes, leading to a problem only associated with the domain's boundary, which can be seen as a 2D problem, and c) the stiffness matrix $\mathbf{K}$ can be factorized during the process and used for all the design variables.

3. Calculating shape sensitivities with FEAVox

This section describes the adaptation of the discrete analytical method to evaluate shape sensitivities when using the cgFEM methodology. In order to do this we have to take into consideration that imposing Dirichlet boundary conditions in an immersed boundary environment requires different strategies from those used in standard FEM. This also has to be considered for the calculation of the shape sensitivities dealt with in this section.

Imposing Dirichlet boundary conditions in Cartesian grids is not a trivial issue, as the degrees of freedom do not necessarily lie on the Dirichlet boundary, so the direct enforcement of the this type of boundary conditions is in general not feasible.

The case of the Neumann boundary conditions can be easily undertaken by simply considering that the integration surface can cut the element and does not necessarily has to correspond with the element faces. In cgFEM we use a stabilized method [66] similar to Nitche's method that modifies the classical structure of the FE linear elastic stiffness matrix and force vector.

The global stiffness matrix is obtained by the contribution of the classical stiffness matrix of each element $\mathbf{k}^e$ and a stabilization term $\mathbf{k}_D^e$ for all the boundary elements containing the Dirichlet boundary.

The stiffness matrix of each element is computed by

$$\mathbf{k}^e = \int_{\Omega^e} \mathbf{B}^T \mathbf{D} \mathbf{B} |\mathbf{J}| \mathrm{d}\Omega \tag{20}$$

where

- Ω^e is the domain in local element coordinates,
- $\mathbf{B}$ is the nodal strains-displacements matrix,
- $\mathbf{D}$ is the stiffness matrix that relates stresses with strains. In this work we consider linear elasticity where, under isotropic behavior, this matrix depends only on E, the Young modulus, and ν, the Poisson ratio of the material,
- $|\mathbf{J}|$ is the determinant of the matrix $\mathbf{J}$, representing $\mathbf{J}$ the Jacobian matrix of transformation of the global coordinates (x, y, z) to the local element coordinates (ξ, η, τ).

and the stabilization term:

$$\mathbf{k}_D^e = \int_{\Gamma_D^e} \frac{\kappa^*}{h} \mathbf{C}^T \mathbf{C} |\mathbf{J}| \mathrm{d}\Gamma \tag{21}$$

where

Γ_D^e is the portion of the Dirichlet boundary within the element,

κ^* is the penalty constant, being $\kappa^* = \kappa \cdot E$,

h is the element size,

$\mathbf{C}$ is the matrix of finite element interpolation if Dirichlet conditions are applied on the three displacement components x, y and z.

$$\mathbf{C} = \mathbf{N} = \begin{bmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 & \dots & N_{nnod} & 0 & 0 \\ 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & \dots & 0 & N_{nnod} & 0 \\ 0 & 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & \dots & 0 & 0 & N_{nnod} \end{bmatrix}$$

with $nnod$ as the number of nodes per element. Otherwise $\mathbf{C} = \mathbf{SN}$, where $\mathbf{S}_{ii} = \sum_d \delta_{i,d}$ would be a diagonal matrix and d is the direction where Dirichlet boundary conditions are applied.

On the other side of the equation, the equivalent force vector $\mathbf{f}$ is evaluated by adding the contribution of the standard FE vector of equivalent forces on nodes $\mathbf{f}_q$, the point loads applied on nodes, the stabilization term of the Dirichlet boundary $\mathbf{f}_g$ and the stabilizing component $\mathbf{f}_s$.

The vector $\mathbf{f}_q$ is the standard FE vector due to point forces, volumetric forces, forces distributed over the Neumann surface of the element, evaluated assembling the contribution $\mathbf{f}_q^e$ of every element e on the domain:

$$\mathbf{f}_q^e = \int_{\Gamma_N^e} \mathbf{N}^T \mathbf{t} |\mathbf{J}| \mathrm{d}\Gamma + \int_{\Omega^e} \mathbf{N}^t \mathbf{b} |\mathbf{J}| \mathrm{d}\Omega + \mathbf{p} \tag{22}$$

where vectors $\mathbf{t}$, $\mathbf{b}$ and $\mathbf{p}$ correspond to the surface, body and point loads, respectively.

The vector $\mathbf{f}_g$ is due to the non-homogeneous Dirichlet condition $\mathbf{u}^h = \mathbf{g}$ on Γ_D and it is evaluated assembling the contribution of every element on the Dirichlet boundary:

$$\mathbf{f}_g^e = \int_{\Gamma_D^e} \frac{\kappa^*}{h} \mathbf{C}^T g |\mathbf{J}| \mathrm{d}\Gamma \tag{23}$$

Finally, $\mathbf{f}_s$ is the stabilizing term which depends on the stress field. In our formulation we use the recovered tractions on Γ_D evaluated from the recovered stress field $\boldsymbol{\sigma}^*$[67] to stabilize, solving the problem iteratively updating the stress field value [66, 68], $\boldsymbol{\sigma}^*(\hat{\mathbf{u}}^h)$ being the FE recovered stress field calculated for an FE solution from a previous iteration (or mesh) $\hat{\mathbf{u}}^h$. The traction on the boundary is defined as $\mathbf{T}(\hat{\mathbf{u}}^h) = \boldsymbol{\sigma}^*(\hat{\mathbf{u}}^h) \cdot \mathbf{n}$ where $\mathbf{n}$ is the unit vector normal to the boundary, then

$$\mathbf{f}_s^e = \int_{\Gamma_D^e} \mathbf{C}^T \mathbf{T}(\hat{\mathbf{u}}^h) |\mathbf{J}| \mathrm{d}\Gamma \tag{24}$$

which modifies the global system such that:

$$(\mathbf{K} + \mathbf{K}_D)\,\mathbf{u} = \mathbf{f}_q + \mathbf{f}_g + \mathbf{f}_s \tag{25}$$

The derivative of (25) with respect to any design variable a_m provides the sensitivity of the calculation

$$\left(\frac{\partial \mathbf{K}}{\partial a_m} + \frac{\partial \mathbf{K_D}}{\partial a_m}\right)\mathbf{u} + (\mathbf{K} + \mathbf{K}_D)\,\frac{\partial \mathbf{u}}{\partial a_m} = \frac{\partial \mathbf{f}_q}{\partial a_m} + \frac{\partial \mathbf{f}_g}{\partial a_m} + \frac{\partial \mathbf{f}_s}{\partial a_m} \tag{26}$$

then, rearranging, yields

$$(\mathbf{K} + \mathbf{K}_D)\,\frac{\partial \mathbf{u}}{\partial a_m} = \left(\frac{\partial \mathbf{f}_N}{\partial a_m} + \frac{\partial \mathbf{f}_g}{\partial a_m} + \frac{\partial \mathbf{f}_s}{\partial a_m}\right) - \frac{\partial \mathbf{K}}{\partial a_m}\mathbf{u} = \mathbf{f}_{ps_m} \tag{27}$$

The discrete analytical method consists of obtaining analytical expressions of the sensitivities of the external forces and stiffness matrix. In our case we used the finite differences approximation of Eqs. (8) and (16) in the evaluation of the velocity field that will be used to obtain the derivatives of the previous equation. Therefore the method used to evaluate the shape sensitivities can be classified as a discrete semi-analytical method. Then using (27) the sensitivities of the displacements are obtained. From these sensitivities other response magnitudes are calculated.

3.1. Evaluation of derivatives

In this section we derive the components of (27) to be able to evaluate the shape sensitivities in the Cartesian grid framework. First, starting with $\mathbf{k}^e$ and considering that the derivative of $\mathbf{D}$ with respect to design variables is zero

$$\frac{\partial \mathbf{k}^e}{\partial a_m} = \int_{\Omega^e} \left[\frac{\partial \mathbf{B}^T}{\partial a_m}\mathbf{D}\mathbf{B} + \mathbf{B}^T\mathbf{D}\frac{\partial \mathbf{B}}{\partial a_m}\right]|\mathbf{J}|\mathrm{d}\Omega + \int_{\Omega^e}\left[\mathbf{B}^T\mathbf{D}\mathbf{B}\frac{\partial |\mathbf{J}|}{\partial a_m}\right]\mathrm{d}\Omega \tag{28}$$

As can be found in [69], this expression depends on known magnitudes and the factors $\frac{\partial \mathbf{B}}{\partial a_m}$ and $\frac{\partial |\mathbf{J}|}{\partial a_m}$, which are a function of the velocity field evaluated above.

To evaluate $\frac{\partial \mathbf{k}^e_D}{\partial a_m}$ it is necessary to take into account that, as the Cartesian grid will not be modified by the design variables, i.e. h and $\mathbf{C}$ do not depend on a_m. Therefore, the only non-zero partial derivative with respect to a_m is $\frac{\partial |\mathbf{J}|}{\partial a_m}$, which leads to:

$$\frac{\partial \mathbf{k}^e_D}{\partial a_m} = \int_{\Gamma^e_D} \frac{\kappa^*}{h}\mathbf{C}^T\mathbf{C}\frac{\partial |\mathbf{J}|}{\partial a_m}\mathrm{d}\Gamma \tag{29}$$

In general, the sensitivity with respect to design variables of the nodal equivalent forces $\mathbf{f}_q$ will have two terms, one dependent on the variation of the forces (punctual,

volumetric, etc.), with respect to design variables, and the second depending on the velocity field. In the first case the expression defining the dependence of the acting forces with respect to the design variables must be available but in this work we considered constant acting forces. In the second case the derivatives had to be calculated by the same procedure as for the stiffness matrix components.

The remaining components of $\mathbf{f}$ have to be derived as

$$\frac{\partial \mathbf{f}_g^e}{\partial a_m} = \int_{\Gamma_D^e} \frac{\kappa^*}{h} \mathbf{C}^T g \frac{\partial |\mathbf{J}|}{\partial a_m} \mathrm{d}\Gamma \tag{30}$$

where we have assumed that the Dirichlet boundary conditions are not a function of the design variables,

$$\frac{\partial \mathbf{f}_s^e}{\partial a_m} = \int_{\Gamma_D^e} \left[\mathbf{C}^T \frac{\partial \mathbf{T}(\hat{\mathbf{u}}^h)}{\partial a_m} |\mathbf{J}| + \mathbf{C}^T \mathbf{T}(\hat{\mathbf{u}}^h) \frac{\partial |\mathbf{J}|}{\partial a_m} \right] \mathrm{d}\Gamma \tag{31}$$

where

$$\frac{\partial \mathbf{T}(\hat{\mathbf{u}}^h)}{\partial a_m} = \frac{\partial \boldsymbol{\sigma}^*}{\partial a_m} \mathbf{n} + \boldsymbol{\sigma}^* \frac{\partial \mathbf{n}}{\partial a_m} \tag{32}$$

As mentioned above, the term $\mathbf{T}$ used to stabilize the Lagrange multipliers is a recovered stress field obtained from the FE solution. To evaluate the stresses in linear elasticity we consider the general expression for the calculation of the FE stresses $\boldsymbol{\sigma}_h$ in continuous isoparametric elements

$$\boldsymbol{\sigma}_h = \mathbf{D}\mathbf{B}\mathbf{u}_h^e \tag{33}$$

$\mathbf{u}_h^e$ being the vector of nodal displacements of element e. Taking the derivative with respect to the design variable a_m yields

$$\frac{\partial \boldsymbol{\sigma}}{\partial a_m} = \mathbf{D}\mathbf{B} \frac{\partial \mathbf{u}_h^e}{\partial a_m} + \mathbf{D} \frac{\partial \mathbf{B}}{\partial a_m} \mathbf{u}_h^e \tag{34}$$

where all terms on the right can be evaluated using the development of the preceding sections. Once we have evaluated both $\boldsymbol{\sigma}$ and $\frac{\partial \boldsymbol{\sigma}}{\partial a_m}$ we can apply the construction of the smoothing field based on a recovery technique shown in [67].

Remark 1. *To simplify the evaluation of* $\frac{\partial \boldsymbol{\sigma}^*}{\partial a_m}$ *we considered* $\frac{\partial \boldsymbol{\sigma}^*}{\partial a_m} = \left(\frac{\partial \boldsymbol{\sigma}}{\partial a_m} \right)^*$. *The numerical results will show that this approximation, previously used in [70], does not influence the results.*

4. Numerical examples

From the premise of the approximate nature of the FEM, the error of the solution associated with the size of the elements of the FE mesh can be termed FE discretization error. Usually this error is quantified in terms of the energy norm $\|\cdot\|$ as:

$$\|\mathbf{e}(\mathbf{u})_{ex}\|^2 = \int_\Omega (\boldsymbol{\sigma}_h - \boldsymbol{\sigma})^T \mathbf{D}^{-1} (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}) \, \mathrm{d}\Omega \tag{35}$$

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

The sensitivity analysis results evaluated by the FEM are also influenced by the discretization error associated with the FE model. Therefore, a way must be defined to evaluate the discretization error in the evaluation of the sensitivities. Following [70] we use the sensitivity of the squared energy norm with respect to each design variable, i.e.

$$\chi_m = \frac{\partial \|\mathbf{u}\|^2}{\partial a_m} = \frac{\partial}{\partial a_m} \int \boldsymbol{\sigma}^T \mathbf{D}^{-1} \boldsymbol{\sigma} \mathrm{d}\Omega \tag{36}$$

and following a similar procedure to that used to derivate the expression (28) we obtain:

$$\mathbf{e}(\chi_m)_{ex} = \sum^{n_e} \int_{\Omega_e} (\boldsymbol{\sigma}_h - \boldsymbol{\sigma})^T \mathbf{D}^{-1} \left(2 \left(\frac{\partial (\boldsymbol{\sigma}_h - \boldsymbol{\sigma})}{\partial a_m} \right) + \frac{(\boldsymbol{\sigma}_h - \boldsymbol{\sigma})}{|\mathbf{J}|} \frac{\partial |\mathbf{J}|}{\partial a_m} \right) |\mathbf{J}| \, \mathrm{d}\Omega_e \tag{37}$$

The following definition of relative error in sensitivities can be used to make the error in sensitivities comparable with the error in energy norm in relative terms:

$$\eta(\chi_m)_{ex} = \sqrt{\left| \frac{e(\chi_m)_{ex}}{\chi_{m_{ex}}} \right|} \tag{38}$$

In the absence of singularities, with this definition, the optimal convergence rate of the relative error in sensitivities with respect to the number of degrees of freedom will have the convergence rate of the relative error in energy norm, i.e. $-p/d$, being p the interpolation order of the FE solution and d the problem dimensionality. Therefore, the convergence rate of $\eta(\chi)$ as defined in (38) will be $-1/3$ for linear elements and $-2/3$ for quadratic elements in 3D problems

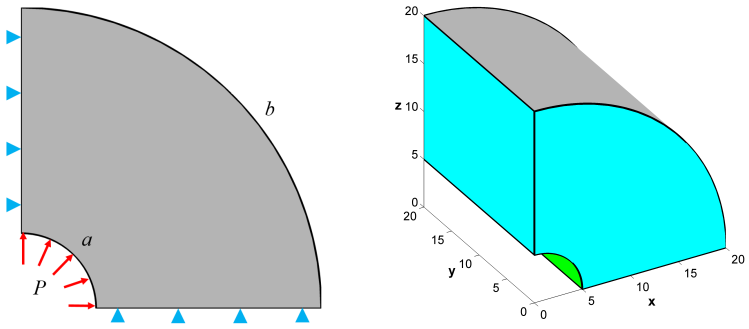
In addition, we know from [52] that there is a relationship between the discretization error in energy norm and the so-called sensitivity discretization error, such that:

$$\frac{e(\chi_m)_{ex}}{\|e(\mathbf{u})_{ex}\|^2} \approx R_m \tag{39}$$

This expression shows that, only with the presence of the discretization error, the discretization error in the sensitivity of the squared energy norm and the squared discretization error in energy norm will both be related by a constant R_m. This relationship between the two types of errors can be adopted as an indicator to measure the quality of the procedure to generate the velocity field.

4.1. Thick wall infinite cylinder under internal pressure

The geometrical model for this test is represented in Figure 9. A linear-elastic analysis is performed on a domain whose boundary representation is defined by NURBS. Appropriate symmetry boundary conditions allow only 1/4 of the section to be modeled. The internal and external surfaces are of radius a and b, with $a = 5$ and $b = 20$. Young's modulus is $E = 1000$, Poisson's ratio is $\nu = 0.3$ and the applied load is $P = 1$. The shape sensitivity analysis of this example considers only one design variable corresponding to the outer radius of the cylinder, thus taking $a_m = b$.



(a) Front view with boundary conditions. (b) 3D representation.

Figure 9: Model of a cylinder under internal pressure.

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

$$u_r = \frac{P(1+\nu)}{E(k^2-1)}\left(r\,(1-2\nu) + \frac{b^2}{r}\right) \tag{40}$$

$$\sigma_r = \frac{P}{c^2-1}\left(1 - \frac{b^2}{r^2}\right) \qquad \sigma_\phi = \frac{P}{c^2-1}\left(1 + \frac{b^2}{r^2}\right) \tag{41}$$

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

$$\begin{gathered} u_x = u_r \cos(\phi), \qquad u_y = 0, \qquad u_z = u_r \sin(\phi) \\ \sigma_x = \sigma_r \cos(\phi)^2 + \sigma_\phi \sin(\phi)^2, \qquad \sigma_z = \sigma_r \sin(\phi)^2 + \sigma_\phi \cos(\phi)^2, \\ \sigma_y = \nu \left(\sigma_x + \sigma_z\right), \qquad \tau_{xz} = (\sigma_r - \sigma_\phi) \sin(\phi) \cos(\phi), \qquad \tau_{xz} = \tau_{yz} = 0 \end{gathered} \tag{42}$$

The analytical sensitivity of the squared energy norm (1/4 of cylinder) is:

$$\chi = \frac{\partial \|\mathbf{u}_{ex}\|^2}{\partial a_m} = 2 \frac{\partial \Pi}{\partial a_m} = 2\pi \frac{p^2(1+\nu)}{E} + \frac{a^4 b(\nu - 1)}{\left(a^2 - b^2\right)^2} \tag{43}$$

For the data used in the model we will have:

$$\begin{aligned} \|\mathbf{u}_{ex}\|^2 &= 0.055815629478779 \\ \chi &= -5.082398781807488 \cdot 10^{-4} \end{aligned} \tag{44}$$

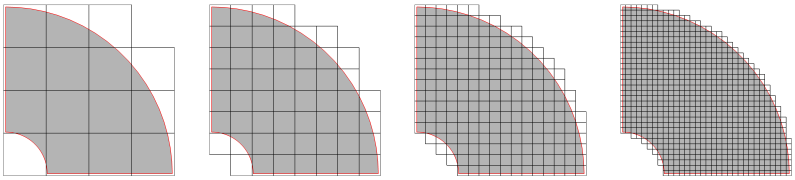
For this problem we will analyze the behavior of the methods used to generate the velocity field, i.e. the proposed least squares approach (LS) and the physical approach (PA), in various FE analyses. In the first analysis we will study the convergence for tri-linear elements (L8) with meshes uniformly refined and h-adapted meshes. The meshes used in this simulation can be seen in Figures 10a and 10b.

Figure 11a shows the relative error of the sensitivity analysis $\eta(\chi)$ and Figure 11b shows its convergence rate as a function of the number of degrees of freedom. The theoretical convergence rate for tri-linear elements $(-1/3)$ is indicated in the plot with a black horizontal line.

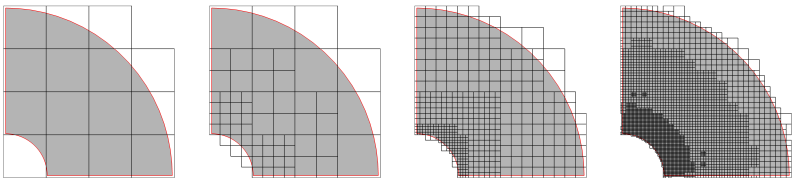
The convergence plots show almost optimal rates for tri-linear elements using the two velocity fields proposed in this contribution, although the physical approach provides more stable results. Regarding the quality of the velocity fields, estimated using R_m (see Figure 11c) we can conclude that all the analyses show good behavior, but the velocity field calculated using the physical approach shows slightly better stability.

In Figure 12 we compare the results for tri-quadratic elements (Q20). Repeating strategy, we analyze uniform meshes (Fig. 10a) and h-adapted meshes (Fig. 10c). Figure 12a shows similar behavior in the convergence of the relative error in sensitivities for the two methods proposed to construct the velocity field. For 3D problems with a non-singular solution, the theoretical convergence rate is $-2/3$ if tri-quadratic elements are used. Figure 12b shows that both velocity fields provide optimal convergence rates. In addition, h-adapted meshes present convergence above the optimal rate, which could indicate that the analysis is still in the pre-asymptotic range.

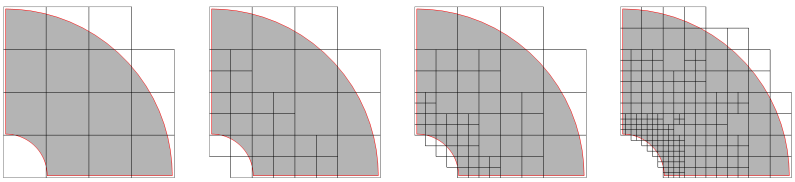
Finally, the quality of the velocity fields quantified by R_m (see Figure 12c) show less stability than that obtained for tri-linear elements, although the interval of oscillation is narrow enough to consider the proposed velocity fields acceptable.



(a) First four meshes of the uniform refinement analysis (L8 and Q20).



(b) h-adaptive meshes with tri-linear elements (L8).



(c) h-adapted meshes with tri-quadratic elements (Q20).

Figure 10: 2D view of the first four meshes of the h-refinement analyses.

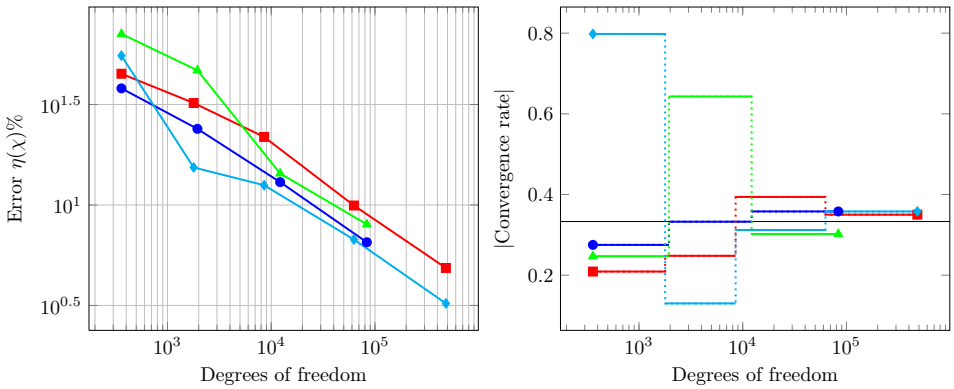
4.2. Sequence of collinear cracks in an infinite plate

In problems of Linear Elastic Fracture Mechanics (LEFM), the Stress Intensity Factor (SIF) is the parameter that characterizes the stress field near a crack tip. This parameter is vital to assess the maximum allowable stress, critical crack size, fatigue life of a component with cracks, etc.

The evaluation of the SIF represents an interesting challenge for shape sensitivity analysis, given the singular nature of the problem[71, 72, 73, 74]. In fact, the quantity called *energy release rate* $\mathcal{G}$ is the variation of the total potential energy of a component as a function of the crack size growth.

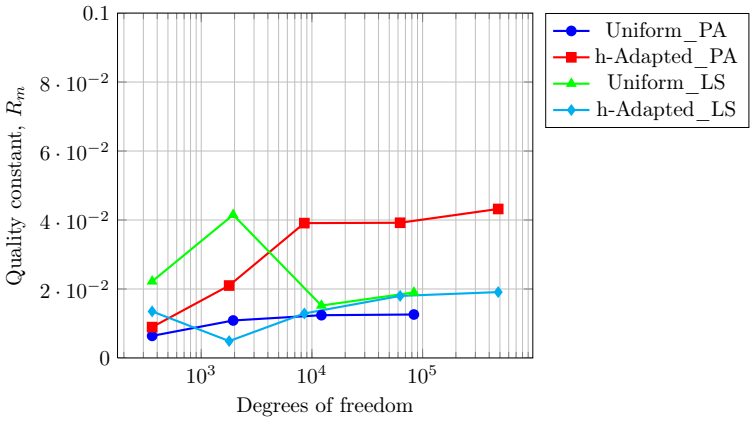
The energy release rate $\mathcal{G}$ for a two-dimensional LEFM problem under mode I loads can be defined as:

$$\mathcal{G} = -\frac{\mathrm{d}\Pi_p}{\mathrm{d}a} = \frac{\mathrm{d}\Pi}{\mathrm{d}a} \tag{45}$$



(a) Evolution of $\eta(\chi)\%$

(b) Convergence rate of $\eta(\chi)\%$



(c) Quality constant, $R_m \approx \frac{e(\chi)_{ex}}{\|e(\mathbf{u})_{ex}\|^2}$

Figure 11: Thick wall infinite cylinder. Analysis with tri-linear elements (L8)

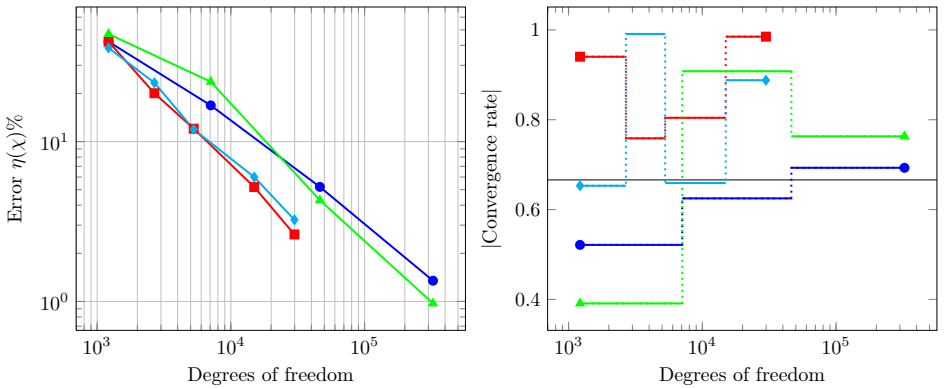
where

Π_p is the total potential energy. In the case where the load remains constant. $\Pi_p = -\Pi$, where Π is the strain energy and

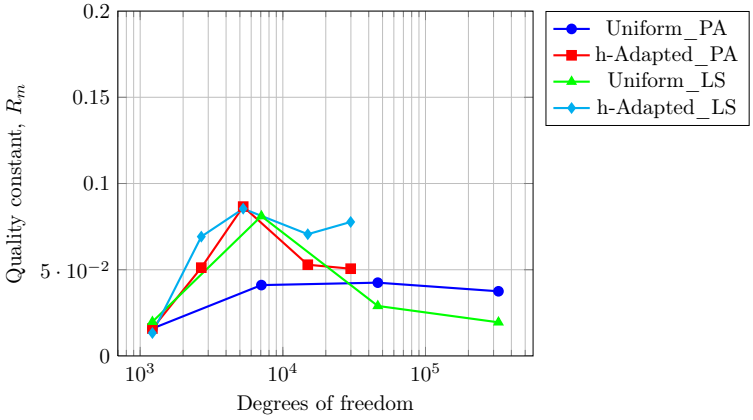
a is the length of the crack.

Since the energy norm squared is equal to twice the strain energy ($\|\mathbf{u}\|^2 = 2\Pi$) it follows:

$$\mathcal{G} = \frac{1}{2}\frac{\mathrm{d}\,\|\mathbf{u}\|^2}{\mathrm{d}a} \tag{46}$$



(a) Evolution of $\eta(\chi)\%$ (b) Convergence of $\eta(\chi)\%$



(c) Quality constant, $R_m \approx \frac{e(\chi)_{ex}}{\|e(\mathbf{u})_{ex}\|^2}$

Figure 12: Thick wall infinite cylinder. Analysis with tri-quadratic elements (Q20)

We can therefore evaluate $\mathcal{G}$ using shape sensitivity analysis, assuming that a, the crack length, is the design variable. From (46) and from the definition of χ_m in (36) we have:

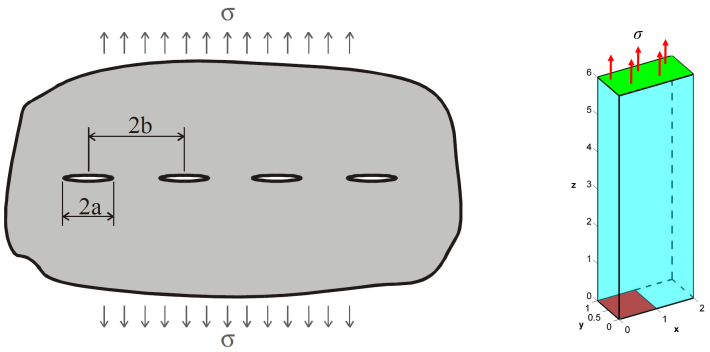
$$\mathcal{G} = \frac{1}{2}\chi \tag{47}$$

On the other hand, the energy release rate $\mathcal{G}$ and the SIF $\mathbf{K}_I$ are related by the following expression:

$$\mathbf{K}_I = \sqrt{E'\mathcal{G}} \tag{48}$$

where $E' = E$ in plane stress and $E' = E/\left(1-\nu^2\right)$ for plane strain, E is the elasticity modulus and ν is the Poisson's ratio.

To evaluate the effectiveness of the sensitivity calculation module in singular problems we are going to use the problem of a sequence of collinear cracks in an infinite plate (see Figure 13).



(a) Plate with a crack of infinite length.

(b) 3D representation of the mode I crack model.

Figure 13: Infintie plate with a sequence of collinear cracks.

Considering the symmetries of the problem in Figure 13a, the FE model involves taking into account the strip bounded by the center of the crack and the equidistant point between the ends of two consecutive cracks, together with the appropriate boundary conditions, as shown in Figure 13b. On the blue surfaces the normal displacements have been blocked to simulate the symmetry and plain strain conditions. The red surface is free and the constant stress σ is applied on the top surface. A considerable height was used in the analysis to make certain that the effect of the finite height on the SIF was insignificant.

In an infinite sequence of collinear cracks subjected to constant stress σ, the exact value of Stress Intensity Factor $\mathbf{K}_I$ is given by equation (Kanninen and Popelar [75]):

$$\mathbf{K}_I = \sigma\sqrt{\pi a}\sqrt{\frac{2b}{\pi a}\tan\left(\frac{\pi a}{2b}\right)} \tag{49}$$

For the data used in the model it yields:

$$\begin{aligned} \mathbf{K}_I &= 200 \\ \chi = 2\mathcal{G} &= 0.007112888 \end{aligned} \tag{50}$$

Since the crack has a top and a bottom, the value of χ in the upper part would be obtained when modeling both sides. However, the model used in the numerical

analysis uses only the top of the crack, so that the value of χ obtained directly through the shape sensitivity analysis approximates to half the value displayed. Thus the value of χ to be compared with the numerical results is:

$$\chi = 0.003556444 \tag{51}$$

In order to evaluate the behavior of the velocity fields, as in the previous problem, we consider using the ratio R_m defined previously. However there is no expression to evaluate the exact energy norm for this problem. To determine the error in strain energy we have taken, as a reference, the solution obtained from a very refined 2D h-adapted mesh (23811 degrees of freedom, 12059 nodes, 5918 elements) with quadratic triangular elements, with an estimated error of 0.0753%, evaluated using the ZZ error estimator [76] i.e. using (35) but substituting the exact stress field $\boldsymbol{\sigma}$ by a recovered stress field $\boldsymbol{\sigma}^*$ obtained by the recovery technique described in [67]. In this mesh, the value of $\|\mathbf{u}\|^2$, which will be considered as exact in the analyses, is:

$$\|\mathbf{u}^*_{ex}\|^2 = 0.009582263 \tag{52}$$

The J integral [77] is commonly used to characterize the singularity at the crack tip in LEFM problems. The behavior of this contour integral can be considerably improved by means of the Equivalent Domain Integral (EDI) method[78, 79]. The transformation of the J contour integral into a domain integral leads to the appearance of an auxiliary function, usually denoted as q, which must be defined by the analyst. This function used in the EDI method is equivalent to the velocity field used to characterize the singularity through the use of shape sensitivity analysis. Therefore, in this problem, we will compare the behavior of the velocity field based on the physical approach (PA) with the behavior of an auxiliary function q commonly used in the EDI method, a *Plateau* function. Hence, as in the case of the auxiliary function q, the Plateau velocity field will be defined as a vector field in the direction of the crack propagation, with a maximum constant value within the volume defined by a radius R_{in} around the crack tip and a linear decrease to 0 within R_{in} and an outer radius R_{out}. Figure 14 shows the appearance of these velocity fields.

As in the previous test problem, a sequence of uniformly refined meshes has been used with both tri-linear (L8) and tri-quadratic (Q20) elements, Figure 15a, and two sequences of h-adapted meshes, one for linear elements, Figure 15b, and another for quadratic elements, presented in Figure 15c.

Figure 16a shows the variation of the relative error in sensitivities $\eta(\chi)$ obtained for each velocity field, while Figure 16c represents the evolution of $\mathbf{K}_I$ in terms of the number of degrees of freedom. Tri-linear elements and both uniform and h-adaptive refinement have been used in the analysis represented. Figure 16b shows the convergence rate of $\eta(\chi)$ with respect to the number of degrees of freedom for the different meshes. We can see how only the h-adaptive simulations reach the optimal theoretical convergence rate of $-1/3$, for 3D tri-linear elements. The convergence rate for singular problems and uniform refinement is dominated by the intensity of

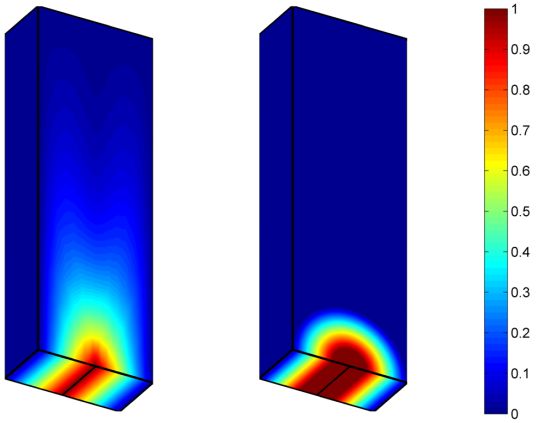
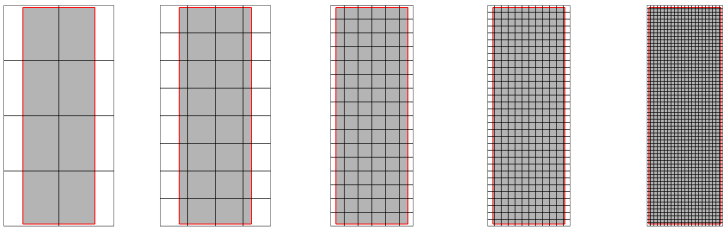


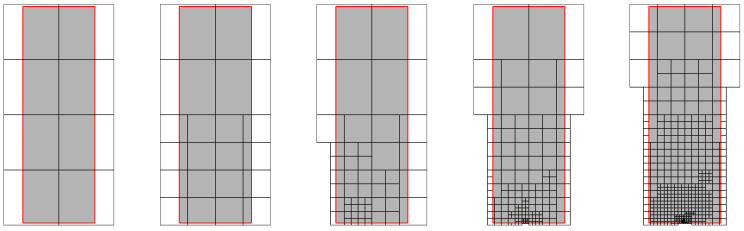
Figure 14: Velocity fields for the crack model. (a) Physical approach and (b) Plateau function.

the singularity λ and not by the degree of the interpolation of the solution. In this case $\lambda = 0.5$, thus, the convergence rate in terms of the number of degrees of freedom will be $-\lambda/d$, d being the dimensionality of the problem. Thus, for $\lambda = 0.5$ and $d = 3$, the theoretical convergence rate with respect to the number of degrees of freedom for uniform refinement is $-1/6$. The convergence rate for uniform refinements is close to this value. Regarding the quality of the velocity fields evaluated using R_m, we observe in Figure 16d how the velocity field defined using the physical approach is more stable for both uniform meshes and for h-adapted meshes but especially for uniform refinement processes.

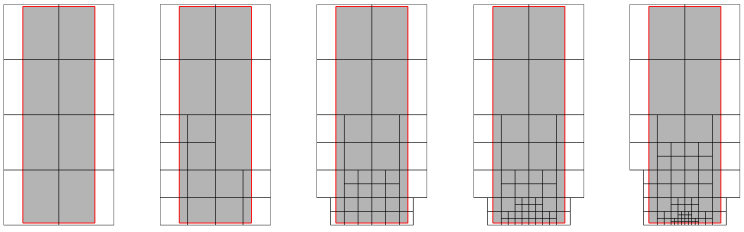
The same analyses were carried out with tri-quadratic elements. Figure 17b shows the convergence rate for the different meshes. We can observe that only the h-adaptive simulations reach the optimal theoretical convergence rate of $-2/3$, for 3D tri-quadratic elements. The uniformly refined meshes have a convergence rate close to the convergence rate for tri-linear elements with uniform refinement, i.e. close to the theoretical convergence rate for this type of mesh, $-1/6$, as expected. Figure 17c represents the evolution of $\mathbf{K}_I$ with respect to the number of degrees of freedom. The evolution of the quality constant for tri-quadratic elements shown in Figure 17d is similar to that of tri-linear elements. The results obtained by the physical approach are clearly more stable than with the plateau velocity, which produces oscillatory behavior, especially in h-adapted meshes.



(a) First five meshes of the uniform refinement analysis (L8 and Q20).



(b) h-adaptive meshes with tri-linear elements (L8).

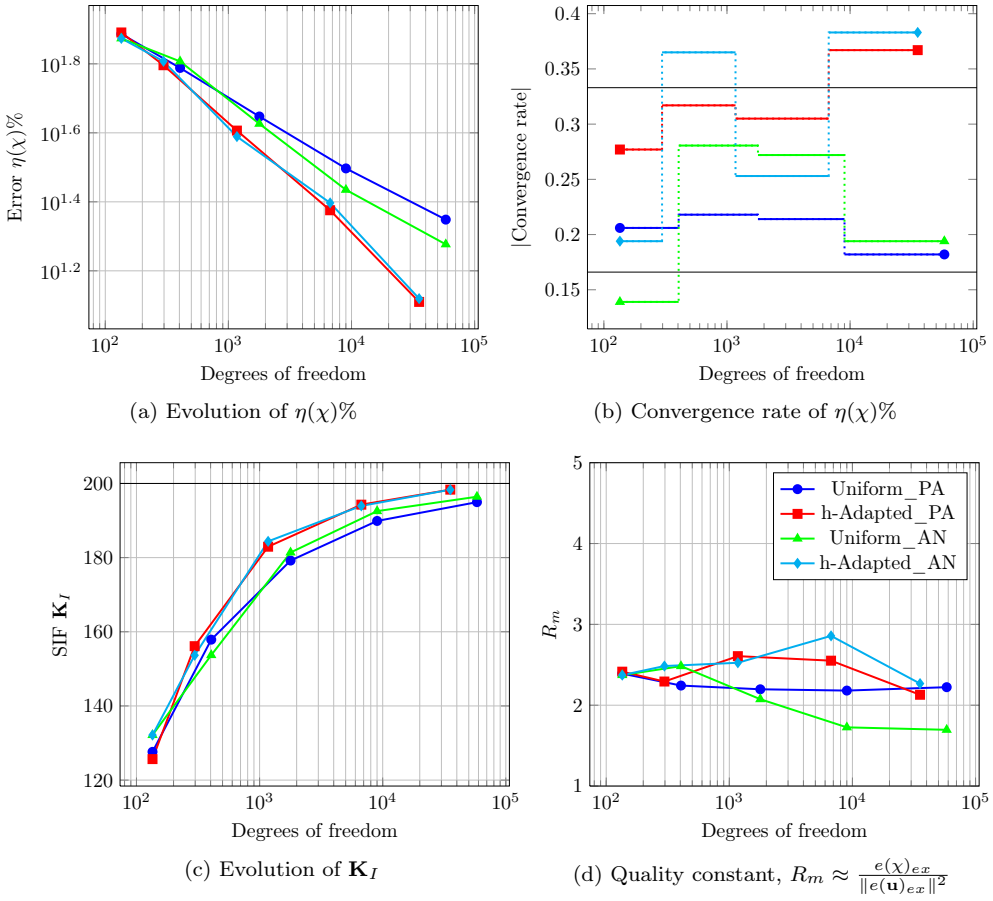


(c) h-adaptive meshes with tri-quadratic elements (Q20).

Figure 15: Infinite sequence of cracks. First five meshes of the h-refinement process.

5. Conclusions

This paper proposes an extension of a methodology for the calculation of shape sensitivities, together with the definition of two methodologies that define the design velocity field for an immersed boundary method where an h-adapted Cartesian grid is used to mesh the embedding domain. This includes adapting the formulation of shape sensitivities, taking into account the special treatment of boundary conditions required by the use of non body-fitted meshes.



(a) Evolution of $\eta(\chi)\%$

(b) Convergence rate of $\eta(\chi)\%$

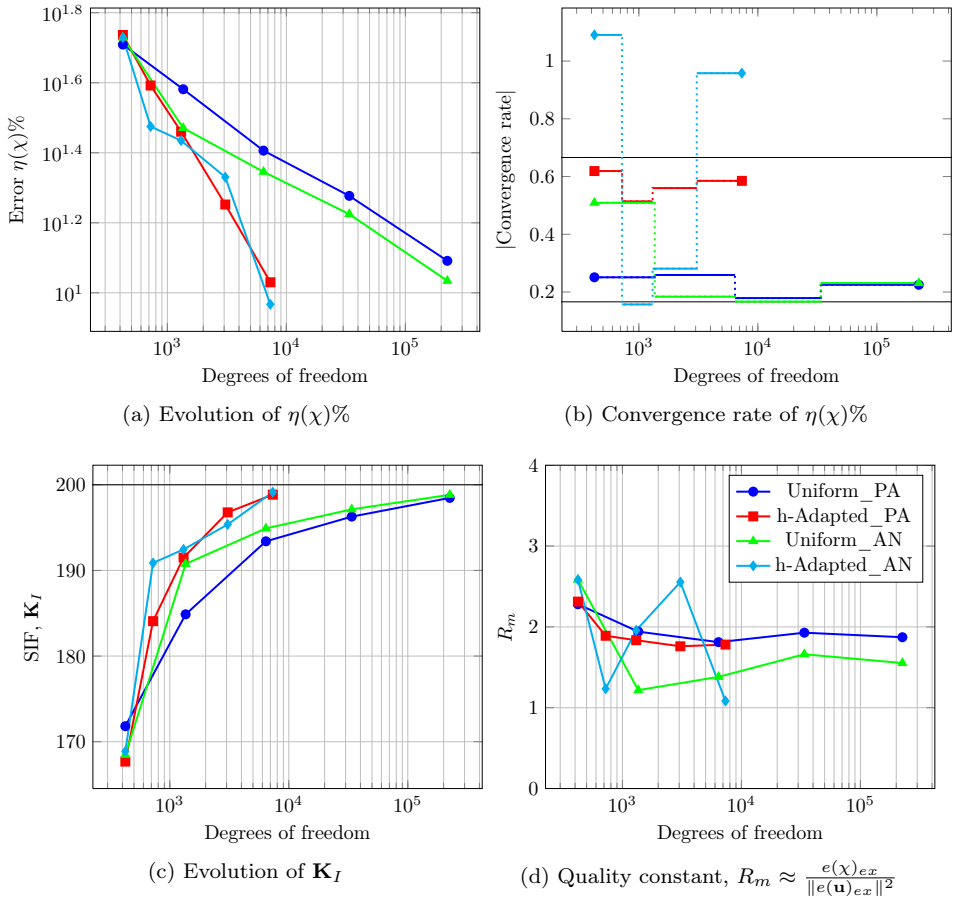
(c) Evolution of $\mathbf{K}_I$

(d) Quality constant, $R_m \approx \frac{e(\chi)_{ex}}{\|e(\mathbf{u})_{ex}\|^2}$

Figure 16: Infinite sequence of cracks. Analysis with tri-linear elements (L8)

Two problems with known analytical solutions, one with a smooth and another with a singular solution, were used in the section devoted to numerical examples. These examples were used to compare the performance of the proposed methodologies in defining the design velocity field and to demonstrate their appropriate behavior in shape sensitivity analysis, within the framework of immersed boundary techniques based on Cartesian grids.

The proposed physical approach for the definition of the design velocity field provided the best behavior. Furthermore, this approach outperforms the behavior of an analytical velocity field commonly used in the context of fracture mechanics to evaluate the Stress Intensity Factor.



(a) Evolution of $\eta(\chi)\%$

(b) Convergence rate of $\eta(\chi)\%$

(c) Evolution of $\mathbf{K}_I$

(d) Quality constant, $R_m \approx \frac{e(\chi)_{ex}}{\|e(\mathbf{u})_{ex}\|^2}$

Figure 17: Infinite sequence of cracks. Analysis with tri-quadratic elements (Q20)

Acknowledgments

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

References

[1] Yoon BG, Belegundu AD. Iterative methods for design sensitivity analysis. *AIAA Journal* 1988; **26**(11):1413–1415. 1

[2] Haftka RT. Semi-analytical static nonlinear structural sensitivity analysis. *AIAA Journal* 1993; **31**(7):1307–1312. 1

[3] Akgün MA, Garcelon GH, Haftka RT. Fast exact linear and nonlinear structural reanalysis and the sherman-morrison-woodbury formulas. *International Journal for Numerical Methods in Engineering* 2001; **50**(7):1587–1606. 1

[4] Kirsch U. *Design-oriented Analysis: a Unified Approach*. Springer Netherlands, 2002. 1

[5] Phelan DG, Haber RB. Sensitivity analysis of linear elastic systems using domain parametrization and a mixed mutual energy principle. *Computer Methods in Applied Mechanics and Engineering* 1989; **77**(1–2):31–59. 2

[6] Arora JS, Lee TH, Cardoso JB. Structural shape sensitivity analysis: relationship between material derivative and control volume approaches. *AIAA Journal* 1992; **30**(6):1638–1648. 2

[7] Arora JS. An exposition of the material derivative approach for structural shape sensitivity analysis. *Computer Methods in Applied Mechanics and Engineering* 1993; **105**(1):41–62. 2

[8] Lee BY. Consideration of body forces in axisymmetric design sensitivity analysis using the bem. *Computers & Structures* 1996; **61**(4):587–596. 2

[9] Lee BY. Direct differentiation formulation for boundary element shape sensitivity analysis of axisymmetric elastic solids. *International Journal of Solids and Structures* 1997; **34**(1):99–112. 2

[10] Navarrina F, López-Fontán S, Colominas I, Bendito E, Casteleiro M. High order shape design sensitivities: a unified approach. *Computer Methods in Applied Mechanics and Engineering* 2000; **188**(4):681–696. 2

[11] Choi KK, Duan W. Design sensitivity analysis and shape optimization of structural components with hyperelastic material. *Computer Methods in Applied Mechanics and Engineering* 2000; **187**(1–2):219–243. 2

[12] Kibsgaard S. Sensitivity analysis-the basis for optimization. *International Journal for Numerical Methods in Engineering* 1992; **34**(3):901–932. 3

[13] Poldneff MJ, Rai IS, Arora JS. Implementation of design sensitivity analysis for nonlinear structures. *AIAA Journal* 1993; **31**(11):2137–2142. 3

[14] Pandey PC, Bakshi P. Analytical response sensitivity computation using hybrid finite elements. *Computers & Structures* 1999; **71**(5):525–534. 3

[15] Moita JS, Infanta J, Mota CM. Sensitivity analysis and optimal design of geometrically non-linear laminated plates and shells. *Computers & Structures* 2000; **76**(1–3):407–420. 3

[16] Ozaki I, Kimura F, Berz M. Higher-order sensitivity analysis of finite element method by automatic differentiation. *Computational Mechanics* 1995; **16**(4):223–234. 4

[17] Wujek BA, Renaud JE. Automatic differentiation for more efficient system analysis and optimization. *Engineerign Optimization* 1998; **31**(1):101–139. 4

[18] Bischof C, Carle A, Khademi P, Mauer A. The adifor 2.0 system for the automatic differentiation of fortran 77 programs. *IEEE Computational Science and Engineering* 1996; **3**(3):18–32. 4

[19] Griewank A, Juedes D, Utke J. Adol-c, a package for the automatic differentiation of algorithms written in c/c++. *ACM Transactions on Mathematical Software (TOMS)* 1996; **22**(2):131–167. 4

[20] Shiriaev D, Griewank A. Adol-f: Automatic differentiation of fortran codes. *Computational Differentiation: Techniques, Applications, and Tools (SIAM)* 1996; **1**:375–384. 4

[21] Choi KK, Twu S. Equivalence of continuum and discrete methods of shape design sensitivity analysis. *AIAA Journal* 1989; **27**(10):1419–1424. 1

[22] Haftka RT, Barthelemy B. *Discretization Methods and Structural Optimization — Procedures and Applications: Proceedings of a GAMM-Seminar October 5–7, 1988, Siegen, FRG*, chap. On the Accuracy of Shape Sensitivity Derivatives. Springer: Berlin, Heidelberg, 1989; 136–144. 1

[23] Haftka R, Adelman H. Recent developments in structural sensitivity analysis. *Structural Optimization* 1989; **1**(3):137–151. 1

[24] Salmenjoki K, Neittaanmäki P. *Computer Aided Optimum Design of Structures: Recent Advances*, chap. Comparison of Various Techniques for Shape Design Sensitivity Analysis. Springer-Verlag: Berlin, Heidelberg, 1989; 367–377. 1

[25] van Keulen F, Haftka R, Kim N. Review of options for structural design sensitivity analysis. Part I: linear systems. *Computer Methods in Applied Mechanics and Engineering* 2005; **194**(30-33):3213–3243. 1

[26] Ródenas JJ, Bugeda G, Albelda J, Oñate E. On the need for the use of error-controlled finite element analyses in structural shape optimization processes. *International Journal for Numerical Methods in Engineering* 2011; **87**(11):1105–1126. 1

[27] H Lian RS S P A Bordas, Simpson RN. Recent developments in the integration of computer aided design and analysis. *Computational Technology Reviews* 2016; **6**:1–36. 1

[28] Peskin CS. Numerical Analysis of Blood Flow in the Heart. *Journal of Computational Physics* 1977; **25**:220–252. 1

[29] Zhang L, Gerstenberger A, Wang X, Liu WK. Immersed Finite Element Method. *Computer Methods in Applied Mechanics and Engineering* 2004; **293**(21):2051–2067. 1

[30] Liu WK, Tang S. Mathematical Foundations of the Immersed Finite Element Method. *Computational Mechanics* 2007; **39**(3):211–222. 1

[31] Haslinger J, Jedelsky D. Genetic algorithms and fictitious domain based approaches in shape optimization. *Struc. Optim.* 1996; **12**:257–264. 1

[32] Kunisch K, Peichl G. Numerical gradients for shape optimization based on embedding domain techniques. *Comput. Optim.* 1996; **18**:95–114. 1

[33] Liu WK, Liu Y, Darell D, Zhang L, Wang XS, Fukui Y, Patankar N, Zhang Y, Bajaj C, Lee J, *et al.*. Immersed Finite Element Method and its Applications to Biological Systems. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**(13):1722–1749. 1

[34] Gil AJ, Arranz-Carreño A, Bonet J, Hassan O. The Immersed Structural Potential Method for Haemodynamic Applications. *Journal of Computational Physics* 2010; **229**(22):8613–8641. 1

[35] Hughes TJR, Cottrell JA, Bazilevs Y. Isogeometric Analysis: CAD, Finite Elements, NURBS, Exact Geometry, and Mesh Refinement. *Computer Methods in Applied Mechanics and Engineering* 2005; **194**:4135–4195. 1

[36] Nguyen VP, Anitescu C, Bordas SPA, Rabczuk T. Isogeometric analysis: An overview and computer implementation aspects. *Mathematics and Computers in Simulation* 2015; **117**:89–116. 1

[37] Zhang Y, Wang W, Hughes TJR. Conformal Solid T-spline Construction from Boundary T-spline Representations. *Computational Mechanics* 2013; **6**(51):1051–1059. 1

[38] Escobar JM, Montenegro R, Rodríguez E, Cascón JM. The meccano method for isogeometric solid modeling and applications. *Engineering with Computers* 2014; **30**(3):331–343. 1

[39] Liu L, Zhang Y, Hughes TJR, Scott MA, Sederberg TW. Volumetric T-spline Construction using Boolean Operations. *Engineering with Computers* 2014; **30**(4):425–439. 1

[40] Li K, Qian X. Isogeometric analysis and shape optimization via boundary integral. *Computer-Aided Design* 2011; **43**(11):1427–1437. 1

[41] Lian H, Kerfriden P, Bordas SPA. Implementation of regularized isogeometric boundary element methods for gradient-based shape optimization in two-dimensional linear elasticity. *International Journal for Numerical Methods in Engineering* 2016; **106**(12):972–1017. 1

[42] Cho S, Ha SH. Isogeometric shape design optimization: exact geometry and enhanced sensitivity. *Structural and Multidisciplinary Optimization* 2009; **38**(1):53–70. 1, 2.1

[43] Ha SH, Choi K, Cho S. Numerical method for shape optimization using T-spline based isogeometric method. *Structural and Multidisciplinary Optimization* 2010; **42**(3):417–428. 1

[44] Qian X. Full analytical sensitivities in NURBS based isogeometric shape optimization. *Computer Methods in Applied Mechanics and Engineering* 2010; **199**(29–32):2059–2071. 1

[45] Sevilla R, Fernández-Méndez S, Huerta A. NURBS-enhanced Finite Element Method (NEFEM): A Seamless Bridge Between CAD and FEM. *Archives of Computational Methods in Engineering* 2011; **18**(4):441–484. 1

[46] Sevilla R, Fernández-Méndez S, Huerta A. 3D-NURBS-enhanced Finite Element Method (NEFEM). *International Journal for Numerical Methods in Engineering* 2011; **88**(2):103–125. 1

[47] Nadal E, Ródenas JJ, Albelda J, Tur M, Tarancón JE, Fuenmayor FJ. Efficient Finite Element Methodology based on Cartesian Grids: Application to Structural Shape Optimization. *Abstract and Applied Analysis* 2013; **2013**. 1

[48] Nadal E. *Cartesian Grid FEM (cgFEM): High Performance h-adaptive FE Analysis with Efficient Error Control. Application to Structural Shape Optimization. PhD Thesis*. Universitat Politècnica de València, 2014. 1

[49] Marco O, Sevilla R, Zhang Y, Ródenas JJ, Tur M. Exact 3D boundary representation in finite element analysis based on Cartesian grids independent of the

geometry. *International Journal for Numerical Methods in Engineering* 2015; **103**:445–468. 1, 2.1.1

[50] Choi KK, Chang KH. A Study of Design Velocity Field Computation for Shape Optimal Design. *Finite Elements in Analysis and Design* 1994; **15**(4):317–341. 2

[51] Choi K, Kim N. *Structural sensitivity analysis and optimization, mechanical engineering series, vol. 1*. Springer-Verlag: Berlin, Heidelberg, 2005. 2

[52] Ródenas JJ, Fuenmayor FJ, Tarancón J. A Numerical Methodology to Assess the Quality of the Design Velocity Field Computation Methods in Shape Sensitivity Analysis. *International Journal for Numerical Methods in Engineering* 2003; **59**(13):1725–1747. 2, 4

[53] Yang RJ, Fiedler MJ. Design Modelling for Large-Scale Three-dimensional Shape Optimization Problems. *ASME Computers in Engineering* 1987; **15**:177–182. 2

[54] Iman MH. Three-dimensional Shape Optimization. *International Journal for Numerical Methods in Engineering* 1982; **18**(5):661–673. 2

[55] Bennett JA, Botkin ME. Structural Shape Optimization with Geometric Problem Description and Adaptive Mesh Refinement. *AIAA Journal* 1985; **23**(3):459–464. 2, 2.2

[56] D Belegundu YM S Zhang, Salagame R. The Natural Approach for Shape Optimization with Mesh Distortion Control. *Technical Report*, Penn State University 1991. 2, 2.1.1, 2.2

[57] Bugeda G, Oliver J. A General Methodology for Structural Shape Optimization Problems Using Automatic Adaptive Remeshing. *International Journal for Numerical Methods in Engineering* 1993; **36**(18):3161–3185. 2

[58] Zhang WH, Beckers P. *Computer Aided Optimum Design of Structures: Recent Advances*, chap. Comparison of Different Sensitivity Analysis Approaches for Structural Shape Optimization. Springer-Verlag: Berlin, Heidelberg, 1989; 347–356. 2

[59] Ródenas JJ. Error de Discretización en el Cálculo de Sensibilidades mediante el Método de los Elementos Finitos. *PhD Thesis* 2001; . 2

[60] Piegl L, Tiller W. *The NURBS Book*. Springer-Verlag, 1995. 2.1, 2.1

[61] Rogers DF. *An Introduction to NURBS: with Historical Perspective*. Elsevier, 2001. 2.1, 2.1

[62] Kiendl J, Schmidt R, Wüchner R, Bletzinger KU. Isogeometric shape optimization of shells using semi-analytical sensitivity analysis and sensitivity weighting. *Computer Methods in Applied Mechanics and Engineering* 2014; **274**:148–167. 2.1

[63] Braibant V, Fleury C. Shape optimal design using b-splines. *Computer Methods in Applied Mechanics and Engineering* 1984; **44**(3):247–267. 2.1

[64] Schramm U, Pilkey WW. The coupling of geometric descriptions and finite elements using NURBs — a study in shape optimization. *Finite Elements in Analysis and Design* 1993; **15**(1):11–34. 2.1

[65] Silva CAC, Bittencourt ML. Velocity fields using NURBS with distortion control for structural shape optimization. *Structural and Multidisciplinary Optimization* 2007; **33**(2):147–159. 2.1

[66] Tur M, Albelda J, Marco O, Ródenas JJ. Stabilized Method to Impose Dirichlet Boundary Conditions using a Smooth Stress Field. *Computer Methods in Applied Mechanics and Engineering* 2015; **296**:352–375. 3, 3

[67] Ródenas JJ, Tur M, Fuenmayor FJ, Vercher A. Improvement of the superconvergent patch recovery technique by the use of constraint equations: the SPR-C technique. *International Journal for Numerical Methods in Engineering* 2007; **70**(6):705–727. 3, 3.1, 4.2

[68] Tur M, Albelda J, Nadal E, Ródenas JJ. Imposing dirichlet boundary conditions in hierarchical cartesian meshes by means of stabilized lagrange multipliers. *International Journal for Numerical Methods in Engineering* 2014; **98**(6):399–417. 3

[69] El-Sayed MEM, Zumwalt KW. Efficient design sensitivity derivatives for multiload case structures as an integrated part of finite element analysis. *Computers & Structures* 1991; **40**(6):1461–1467. 3.1

[70] Fuenmayor FJ, Oliver JL, Ródenas JJ. Extension of the Zienkiewicz-Zhu error estimator to shape sensitivity analysis. *International Journal for Numerical Methods in Engineering* 1997; **40**(8):1413–1433. 1, 4

[71] Fuenmayor FJ, Domínguez J, Giner E, Oliver JL. Calculation of the stress intensity factor and estimation of its error by a shape sensitivity analysis. *Fatigue & Fracture of Engineering Materials & Structures* 1997; **20**(5):813–828. 4.2

[72] Giner E, Fuenmayor FJ, Besa AJ, Tur M. An implementation of the stiffness derivative method as a discrete analytical sensitivity analysis and its application to mixed mode in LEFM. *Engineering Fracture Mechanics* 2002; **69**(18):2051–2071. 4.2

[73] Chowdhury MS, Song C, Gao W. Shape sensitivity analysis of stress intensity factors by the scaled boundary finite element method. *Engineering Fracture Mechanics* 2014; **116**:13–30. 4.2

[74] Choi MJ, Cho S. Isogeometric shape design sensitivity analysis of stress intensity factors for curved crack problems. *Computer Methods in Applied Mechanics and Engineering* 2014; **279**:469–496. 4.2

[75] Kanninen MF, Popelar CH. *Advanced Fracture Mechanics, Oxford Engineering Science Series*. Oxford University Press, 1985. 4.2

[76] Zienkiewicz OC, Zhu JZ. A Simple Error Estimator and Adaptive Procedure for Practical Engineering Analysis. *International Journal for Numerical Methods in Engineering* 1987; **24**(2):337–357. 4.2

[77] Rice JR. A Path Independent Integral and the Approximate Analysis of Strain Concentration by Notches and Cracks. *Journal of Applied Mechanics* 1968; **35**(2):379–386. 4.2

[78] Li FZ, Shih CF, Needleman A. A comparison of methods for calculating energy release rates. *Engineering Fracture Mechanics* 1985; **21**(2):405–421. 4.2

[79] Shivakumar KN, Raju IS. An equivalent domain integral method for three-dimensional mixed-mode fracture problems. *Engineering Fracture Mechanics* 1992; **42**(6):935–959. 4.2

PAPER E

Structural shape optimization using Cartesian grids and automatic h-adaptive mesh projection

O. Marco, J. J. Ródenas, J. Albelda, E. Nadal and M. Tur

Preprint submitted to Structural and Multidisciplinary Optimization

Abstract

We present a novel approach to 3D structural shape optimization that leans on an Immersed Boundary Method. A boundary tracking strategy based on evaluating the intersections between a fixed Cartesian grid and the evolving geometry sorts elements as internal, external and intersected. The integration procedure used by the NURBS-Enhanced Finite Element Method accurately accounts for the nonconformity between the fixed embedding discretization and the evolving structural shape, avoiding the creation of a boundary fitted mesh for each design iteration, yielding in very efficient mesh generation process. A Cartesian hierarchical data structure improves the efficiency of the analyses, allowing for trivial data sharing between similar entities or for an optimal reordering of the matrices for the solution of the system of equations, among other benefits. Shape optimization requires the sufficiently accurate structural analysis of a large number of different designs, presenting the computational cost for each design as a critical issue. The information required to create 3D Cartesian h-adapted mesh for new geometries is projected from previously analyzed geometries using shape sensitivity results. Then, the refinement criterion permits one to directly build h-adapted mesh on the new designs with a specified and controlled error level. Several examples are presented to show how the techniques here proposed considerably improve the computational efficiency of the optimization process.

Key words

Cartesian grids; h-refinement; Shape optimization; NEFEM

Contents

1 Introduction **269**

2 Cartesian grids for optimization **272**
2.1 Data sharing 274
2.2 Nested domain reordering 276

3 Structural and shape sensitivity analysis **279**
3.1 Objective function and constraints 282
3.2 Error estimator 282

4 Automatic *h*-adaptive mesh projection **283**

5 Numerical examples **286**
5.1 Performance of the direct solver 288
5.2 Thick-wall cylinder defined by 1 design variable 290
5.3 Thick-wall cylinder defined by 4 design variables 292
5.4 Connecting rod defined by 8 design variables 293

6 Conclusions **296**

Bibliography **297**

1. Introduction

The structural shape optimization problem can be tackled as the minimization of a real function f, which depends on some variables and is subjected to several constraints. The generic form of such problem is:

$$\begin{array}{cll} \text{minimize} & f(\mathbf{a}) & \\ \text{where} & \mathbf{a} = \{a_i\} & i = 1, \ldots, n \\ \text{verifying} & \mathbf{g}(\mathbf{a}) \leq 0 & j = 1, \ldots, m \\ \text{and} & \mathbf{h}(\mathbf{a}) = 0 & k = 1, \ldots, l \end{array} \tag{1}$$

being f the objective function, a_i are the design variables, g_j are inequality constraints and the values h_k define equality constraints. The vector $\mathbf{a}$ defines a specific structural shape and the task consists in finding the $\mathbf{a}$ values which define the optimum design.

The algorithms for the solution of (1) are, normally, iterative. Among the optimization algorithms we will mainly focus in this paper on the gradient-based algorithms because of their fast convergence to the optimal solution. These methods require the computation of the objective function, the constraints and their derivatives (sensitivities) with respect to the design variables for each geometry considered during the process. In addition, in every step of that process, it is necessary to evaluate the values, and their sensitivities, for f and $\mathbf{g}$. In this work, these calculations are done through Finite Element Analysis.

There are different approaches and many codes to solve the optimization problem. However, some problems in this context, identified long time ago[1, 2], still remain unsolved, like the incorporation of robust parametrization techniques for the definition of each design and the unwanted variations in the structural response due to mesh-dependency effects. Also, in many optimization processes based on the use of finite element analyses, there is no control on the accuracy of the numerical solution. As a result, when the process comes to an end there is no guarantee on the practicability of the final outcome; sometimes analyses with higher accuracy levels would expose that the final design is unfeasible, contravening one or more of the constraints imposed. The control of the error related with the finite element computation and its impact on the solution of the optimization problem was analyzed in [3].

Two main concepts evolved to bypass these drawbacks. On the one hand, the design update procedure can be assigned to geometry model[4, 5]. In this case, the nodal coordinates manipulation within the finite element discretization is avoided, thus removing impracticable patterns that cannot be defined by any combination of design variables. Another idea would be to improve the geometrical accuracy of the models by integrating CAD representations with the FEM solvers, as in the case of Isogeometric Analysis [6, 7]. However, in its finite element form, generating an

analysis-suitable solid discretization is an open topic[8, 9, 10]. Some works on IGA shape optimization can be found in [11, 12, 13, 14, 15].

On the other hand, the finite element model could be updated through the optimization procedure to improve the accuracy of the numerical simulation results or to enhance the element quality[16, 17, 18]. This may, for instance, take the form of adaptive mesh refinement based on error estimation in energy norm[19] or goal oriented adaptivity[20]. However, when it comes to complicated geometries or to large shape changes, these strategies may still necessitate computationally expensive re-meshing algorithms.

From this perspective, so-called Immersed Boundary discretization techniques seem the most appropriate choice for structural shape optimization. The main notion behind these methods is to extend the structural analysis problem to a easy-to-mesh approximation domain that encloses the physical domain boundary. Then, it suffices to generate a discretization based on the approximation domain subdivision, rather than a geometry-conforming finite element mesh. Moreover, when the structural component is allowed to evolve, the physical points move through the fixed discretization created from the embedding domain where there will be no mesh distortion. There are plenty of alternatives within the IBM scope. Among many other names used to describe these FE techniques where the mesh does not match the domain's geometry, we have the Immersed Boundary Method (IBM) [21] and the Immersed Finite Element Method (IFEM) [22]. These methods have been studied by a number of authors for very different problems including, of course, shape optimization [23, 24, 25, 26, 27].

Nevertheless, the attractive advantages of IBM come together with numerical challenges. Basically, the computational effort has moved from the use of expensive meshing algorithms towards the use of, for example, elaborated numerical integration schemes to be able to capture the mismatch between the geometrical domain boundary and the embedding finite element mesh. All intersected elements have to be integrated properly in order to account for the volume fractions interior to the physical domain. The domain integration for these methods have been investigated in the literature.

The first intuition, is to homogenize the material properties within intersected elements based on the actual volume fraction of these elements covered by the domain. This approach is straight forward and could be computationally efficient but it provides low accuracy for the structural analysis[28, 29].

A more selective approach is employed in the finite cell framework[30, 31, 32]. Therein, for all intersected elements, a number of integration points are provided employing hierarchical *octree* data structures[33, 34, 35] for their distribution. Then, only the integration points interior to the physical domain are considered in the respective integral contribution. However, regardless of the number of integration points, the exact representation of the geometry is not possible.

Still, the highest level of accuracy and optimal convergence rate, for the embedding domain structural analysis, is obtained only when proper integration schemes are used

along the intersected elements. Very recently, several methodologies to perform high-order integration in embedded methods have emerged, such as the so-called 'smart octrees' tailored for Finite Cell approaches[36] or techniques used where the geometry is defined implicitly by level sets[37]. Even in the first of these approaches, where the isoparamentric reoriented elements only provide an approximated FE description of the boundary, the exact geometry is not taken into account at the integration stage.

A step further to improve accuracy and retain the optimal convergence rate of the FE solution is to consider the exact geometry when integrating. In the embedded domain framework this can be realized by the use of a separate element-wise tetrahedralization that is used for integration purposes only. The present contribution is concerned with the formulation and implementation of this last approach. The methodology is based in the Cartesian grid FEM (cgFEM) succesfully implemented in 2D [38, 39] and 3D [40, 41] problems. The Cartesian grid FEM relies on an explicit geometry description using parametric surfaces (i.e. NURBS or T-spline) and includes NURBS-enhanced integration techniques [40, 42, 43] in order to consider exactly the boundary description of the embedded domain. Stabilization terms are added at the elements cut by the Dirichlet and Neumann boundaries to ensure the appropriate satisfaction of these boundary conditions to maintain the optimal convergence rate of the FE solution and to provide control of the variation of the solution in the vicinity of the boundary [44].

In order to calculate the sensitivities of the magnitudes that take part in the shape optimization analysis, a formulation for the derivation of design sensitivities in the discrete setting is used[45]. This formulation takes into account the derivatives of the stabilized formulation implemented for the imposition of boundary conditions[44].

In this paper we propose a structural shape optimization method that will benefit from the accuracy of cgFEM but also from the computational efficiency due to the a data structure that allows sharing information between the different geometries analyzed during the procedure. Last, this work presents a strategy that provides an h-adapted mesh for every design without performing a complete adaptive procedure. It is based on the calculation of the sensitivities of the magnitudes present during the refinement step with respect to the design variables. This sensitivity analysis can be performed only once on a reference geometry, and then utilized to project the results of the analysis to other designs just before being analyzed. This procedure is useful for moderate shape modifications during the whole optimization process. Alternatively, the shape sensitivity analysis can be performed also for other geometries obtained during the optimization process if required. The projection of information allows to generate appropriate h-adapted meshes in one preprocess step which, compared to standard re-meshing operations, significantly reduces the computational cost of mesh generation. This method is inspired by a similar strategy that was developed and used in the context of gradient-based[46] and evolutionary[47] optimization methods, based on the standard, body-fitted, Finite Element Method.

The paper is organized as follows: A brief review of basic features of the cgFEM methodology will be shown in Section 2, including how to take advantage of them in shape optimization. Section 3 will present the formulation used for the structural and for the sensitivity analysis. Section 4 will be devoted to explain our strategy for h-adaptive mesh projection. 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. Cartesian grids for optimization

This work has been developed as a logical continuation of [40] where a new FEA methodology called cgFEM was presented. This methodology was implemented in a FE code for the analysis of structural 3D components called FEAVox, where the main novelty was the ability to perform accurate numerical integration in non-conforming meshes independent of the geometry.

The foundations of mesh generation in cgFEM consists in defining an embedding domain Ω such that an open 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. In Figure 1 we can appreciate the embedding domain Ω interacting with $\Omega_{\texttt{Phys}}$.

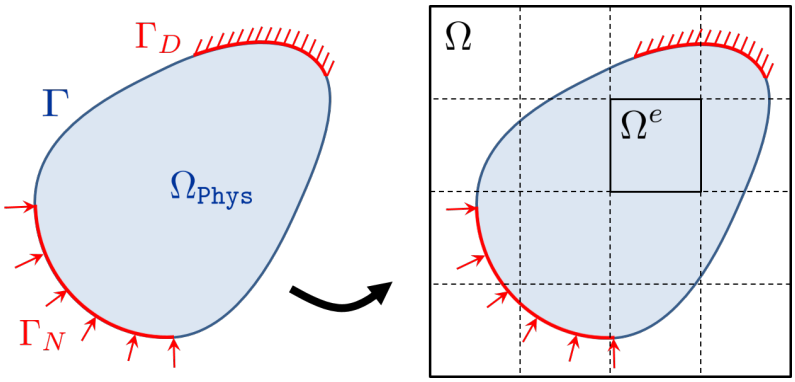


Figure 1: Immersed Boundary Method environment. Domain $\Omega_{\texttt{Phys}}$ within the embedding domain Ω.

The discretization of the embedding domain is based on a sequence of uniformly refined Cartesian meshes to mesh the domain Ω where the different levels of the

Cartesian meshes are connected by predefined hierarchical relations. There are plenty of techniques that follow these principles, all of them based, in one way or another, on the *octree* concept [33, 34, 35].

The first step of the analysis consists of creating the analysis mesh taking a set of non-overlapped elements of different sizes from the different levels of the Cartesian grid pile (see Figure 2). In order to enforce C^0 continuity between adjacent elements of different levels multipoint constraints (MPCs)[48, 49] are used.

During the creation of the FE analysis mesh used to solve the boundary value problem we have to classify the elements of the Cartesian grid in three groups: boundary, internal and external elements. In order to do that, we need to know the relative position of the domain of interest with respect to the embedding domain. In [40] we proposed a robust procedure to find intersections, between the surfaces of the boundary and the axes of the Cartesian grids, based on the ray-tracing techniques[50, 51, 52, 53, 54].

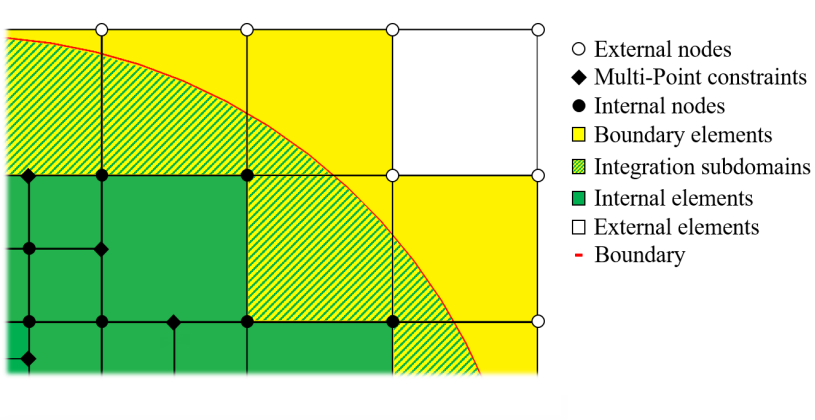


Figure 2: Components of an immersed boundary environment.

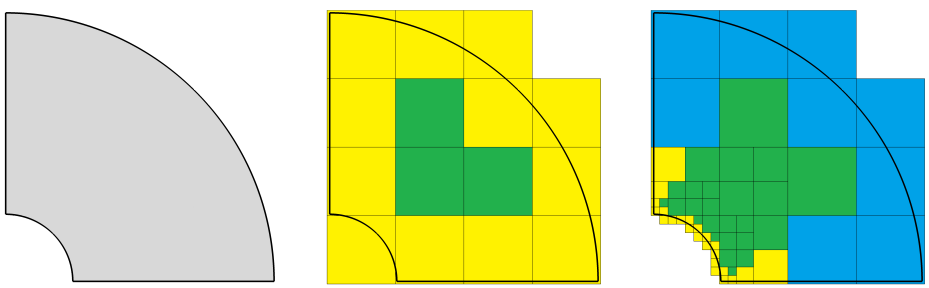
Internal elements are standard FE elements whose affinity with respect to the embedding domain Ω is exploited in order to save the computational cost. Regarding the integration of intersected elements in cgFEM, we proposed a strategy in [40] to perform the integration over the internal part of these elements. In order to achieve this, we generate a tetrahedralization into each boundary element. This submesh of tetrahedrons will be used only during the integration step. Numerical integration over the intersected elements is then accomplished by integrating over each subdomain of the tetrahedralization. In order to perform the integration over the subdomains, we proposed a NURBS-Enhanced integration strategy [42, 43] that takes into account the exact geometry defined by the CAD model.

In cgFEM, Dirichlet boundary conditions are imposed using stabilized Lagrange multipliers. More precisely, the procedure chosen to impose the constraints follows the technique proposed by [44]. This method is suitable for h-refinement in the context of hierarchical Cartesian grids.

2.1. Data sharing

It is worth noting that the hierarchical relationships considered in the data structure suggest an automatic improvement of the mesh refinement thus positively affecting the efficiency of the FE implementation. Nodal coordinates, mesh topology, hierarchical relations, neighborhood patterns, and other geometric information are algorithmically evaluated when required.

In addition, all internal elements share the same stiffness matrix, for constant material properties and linear elasticity problems, which is only calculated once on a reference element. Then, a scale factor related to the mesh level is used to adapt the stiffness to the actual element size. As we can imagine, this implies a major increase in efficiency of the generation of the numerical model. Figure 3a shows a cross section of a model of a quarter of a cylinder, Figure 3b presents a coarse analysis mesh and Figure 3c shows the mesh obtained after its h-adaptive refinement. For both meshes we only have to evaluate one element for the domain colored in green, which represents all the internal elements.



(a) Model of a quarter of a cylinder.

(b) Coarse mesh i.

(c) h-adapted mesh $i+1$.

Figure 3: *Vertical data sharing* example.

On the other hand, each boundary element is trimmed differently, so each of these elements will require a particular evaluation of the element matrices. Contrasting with many h-adaptive FE codes, where the previous meshes are discarded and new ones are created, the use of Cartesian grids together with the hierarchical data structure allows recycling calculations performed in previous meshes.

In this context, the so-called *vertical data sharing* by means of which the matrices of elements present in different meshes of the same h-adaptive process will not be re-evaluated. Figure 3c represents the resulting mesh of a h-adaptive process where the blue colored elements represent elements evaluated in previous meshes that do not require to be re-evaluated. Hence, the only element matrices to be evaluated for the analysis of the mesh shown in Figure 3c correspond to the yellow elements.

Within the context of the traditional FEM, each geometry requires a different body-fitted mesh, therefore, the elements of different geometries are, generally, completely different and unrelated. This situation makes very difficult to enable an efficient exchange of information between different geometries. However, cgFEM provides a framework to define an operation, called *horizontal data sharing*, to further improve the computational efficiency of the optimization process by allowing the transfer of information between elements of different geometries. This data sharing just requires all geometries to be defined using the same embedding domain to ensure that the Cartesian grid pile is the same for all geometries, making the inter-geometries data transfer possible.

The different components of a parametric definition of the boundary of the models to be analyzed can be subdivided into three types:

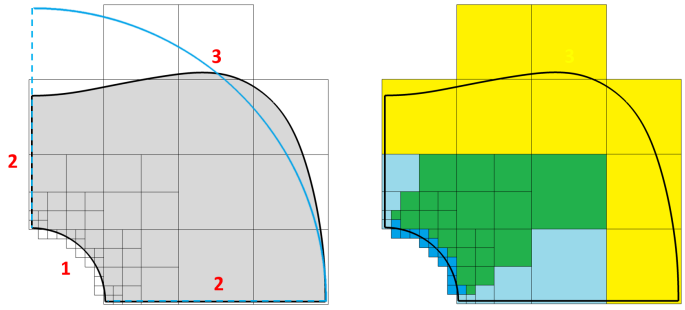
1. Fixed part. This is the part of the boundary that remains fixed in all the geometries (such as the internal curve of the cylinder represented in Figure 4a).

2. Moving part with fixed intersection pattern. This surfaces or curves can be changed by the optimization algorithm, but those changes do not affect the intersection with the surrounding elements in the same manner. In Figure 4a we can see two curves type 2 corresponding with planes of symmetry of the model.

3. Free moving part. This part of the geometry could freely change during the optimization analysis without a predictable pattern. The outer curve in Figure 4a is a good example of this kind of entity.

The *horizontal data sharing* consists of reusing, on the one hand, the computations attached to the elements intersected by the fixed part of the boundary in the different geometries analyzed during the optimization process. For instance, in Figure 4b we can see a h-adapted mesh of a geometry, $j+1$, that represents a perturbation of the original model, j, shown in Figure 3a. In Figures 4c and 4d we can verify how the blue elements used in the mesh in Figure 4b are inherited from separated meshes of a different individual evaluated previously. Dark blue represents the elements intersected by entities type 1 and light blue the elements intersected by type 2 entities. As in the *vertical sharing*, the green elements will be evaluated as explained before and the yellow elements will be the only elements evaluated for this individual. Observe that the *horizontal data sharing* implies a significant reduction of calculations. For example, the number of elements that need integration (yellow elements) for the mesh

shown in Figure 4b is considerably lower than the total number of elements in the mesh.



(a) Different type of entities.

(b) Individual $j+1$.

(c) Individual j, mesh i.

(d) Individual j, mesh $i+1$.

Figure 4: *Horizontal data sharing* example.

2.2. Nested domain reordering

Solving large sparse linear systems is the most time-consuming computation in shape optimization using FEM.

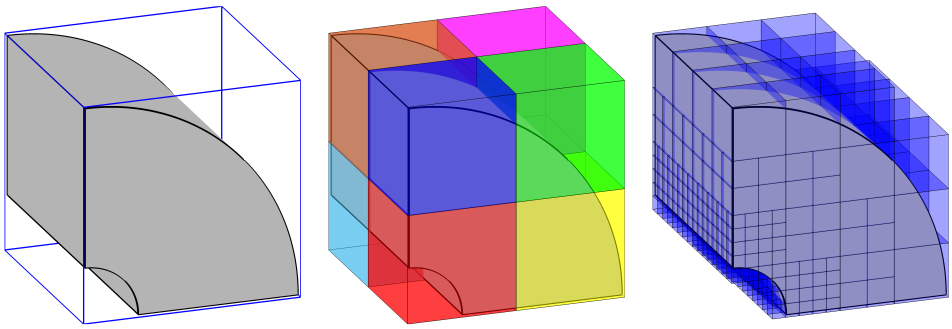
In this contribution we are interested in direct solvers. Matrix reordering plays an important role on the performance of these solvers. In fact, it is common to reorder the system matrix before proceeding to its factorization as it can increase the sparsity of the factorization, making the overall process faster and reducing the storage cost. Finding the optimal ordering is usually not possible although heuristic methods can be used to obtain good reorderings at a reasonable computational cost.

This section is intended to show how the hierarchical data structure inherent to the Cartesian grids, thus directly related to the mesh topology, can be used to directly obtain a reordering of the system matrix that speeds up the Cholesky factorization process. The Nested Domain Decomposition (NDD) technique is a domain decomposition technique specially tailored to h-adaptive FE analysis codes with refinement based on element subdivision. The technique simply consists in recursively subdividing the domain of the problem using the hierarchical structure of the mesh. This technique was first described in [55] and applied in an implementation of a FEM that used geometry-conforming meshes. Later, NDD was adapted to a Cartesian grid environment in 2D[38]. In this paper, we will use a 3D generalization of NDD. The technique consist in subdividing the domain of the problem considering that each element of a uniform grid of the lowest levels of the Cartesian grid pile (normally the Level-1 grid, with 2x2x2 elements). Then, the degrees of freedom of the nodes of the mesh to be analyzed falling into a subdomain will be allocated together in the stiffness matrix. The nodes falling on the interface of the subdomains will not be reordered and will simply be moved to the end of the matrix producing the typical arrowhead type structure of the domain decomposition techniques. This idea is then recursively applied into each original subdomain producing a nested arrowhead type structure. This reordering will provide a considerable reduction of the computational cost associated to the resolution of the system of equations with a minimum computational cost.

Figures 5, 6 and 7 graphically show the process. The embedding domain, Figure 5a, is subdivides into 8 subdomains or regions as shown in Figure 5b. Each subdomain is represented in a different color. We can easily identify those subdomains with the elements of the first refinement level. Thus, the nested reordering in cgFEM will be made up by grouping the nodes according to the corresponding element in the hierarchical structure. Figure 5c shows an example of an analysis mesh where we are going to apply the nested reordering.

For the sake of clarity we will use a 2D representation of the process. Figure 6a shows the domain subdivision considering the Level-1 grid. The nodes are subdivided into 9 different categories. Only 5 of theses categories are shown in the 2D representation of Figure 6. The colored ones indicate the nodes falling into each one of the elements of the Level-1 grid. Black nodes are those falling on the interface between the Level-1 elements. The stiffness matrix will be reordered, grouping all nodes of the same color, as shown in Figure 7b. This grouping creates an arrowhead-type structure made up of blocks. It can be noticed that the blocks on the diagonal (two of them clearly shown shadowed in blue and red) show an structure similar to the structure of the original non-reordered stiffness matrix shown in Figure 7a.

Level-2 reordering, Figure 6b, indicates that each of the Level-1 subdomains is again reordered in the same way. For instance, the red subdomain in Figure 6a is subdivided into 8 subdomains (only 4 are shown in 2D) separated by their interface, represented in black, as shown in Figure 6b. Interfaces of previous levels are repre-



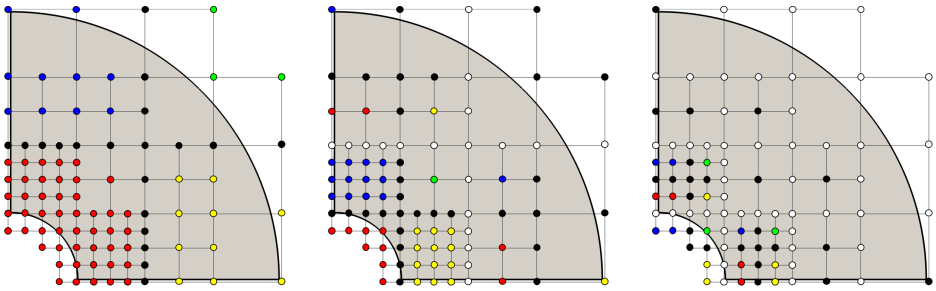
(a) Body within the embedding domain $\Omega_{\text{Phys}} \subset \Omega$.

(b) Level 1 subdivision.

(c) Example of 3D mesh to be reordered.

Figure 5: Nested Domain Decomposition environment.

sented by white nodes. The same process is followed for the next levels, using the elements of the corresponding level of the hierarchical structure.



(a) Level-1 decomposition.

(b) Level-2 decomposition.

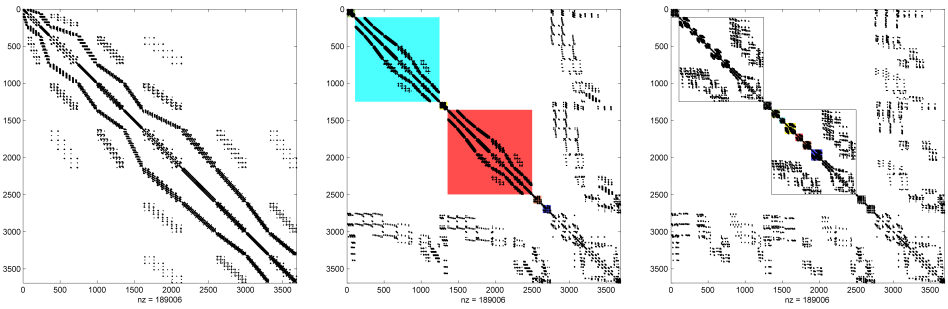
(c) Level-3 decomposition.

Figure 6: Nested Domain Decomposition scheme.

In the process, each node of the mesh is given a code with as many digits as levels of the Cartesian grid pile used. The i-th digit of the code contains the subdomain number (1 to 8) of the node considering the Level-i grid, or 9 if the node is on the interface between the Level-i subdomains, as in Figure 6a to Figure 6c for levels 1 to 3. Once the code of each node has been obtained a simple 'alphabetical' reordering of the codes provides the NDD reordering of the nodes. The degrees of freedom of the matrix will then be reordered considering nodal reordering.

The result of the NDD reordering generates the nested arrowhead type structure of the stiffness matrix represented in Figure 7c. This nested arrowhead type structure

obtained could also be used to define efficient nested domain decomposition solvers or iterative solvers, as in [56] where we showed initial implementations of these types of solvers. However, in this contribution we have just used this technique to reorder the system of equations to improve the performance of the Cholesky factorization.



(a) Original stiffness matrix. (b) Level-1 decomposition reordering. (c) Last reordering (Level-4).

Figure 7: Nested Domain Decomposition output.

In the section devoted to numerical examples we will show the performance of this method in comparison with other common procedures.

3. Structural and shape sensitivity analysis

In this paper we use a method to compute the shape sensitivities that was presented in [45]. Here we recall the main features for clarity of the presentation. We will only consider the solution of structural problems governed by the standard elliptic equations (see, for instance [57]), where introducing appropriate boundary conditions and discretizating using FEM we obtain the standard linear system of equations:

$$\mathbf{K}\mathbf{u} = \mathbf{f} \tag{2}$$

This formulation, valid for standard FEM, will need to be adapted to our immersed boundary framework of our approach. Imposing Dirichlet boundary conditions in Cartesian grids is difficult as the nodes do not necessarily lie on the Dirichlet boundary, thus making the direct enforcement of the essential boundary conditions, in general,

not possible. In cgFEM we use an stabilized method[44] similar to Nitche's method, that modifies $\mathbf{K}$ and $\mathbf{f}$ leading to a new system of equations:

$$\underbrace{(\mathbf{K}+\mathbf{K}_D)}_{\text{Stiffness matrix}}\ \mathbf{u} = \underbrace{\mathbf{f}_q+\mathbf{f}_g+\mathbf{f}_s}_{\text{Equivalent force vector}} \tag{3}$$

where the global stiffness matrix is obtained by the contribution of the classical stiffness matrix $\mathbf{K}$ and a stabilization term $\mathbf{K}_D$ for all the boundary elements containing the Dirichlet boundary. On the other side of the equation, the equivalent force vector is evaluated by adding the contribution of the standard FE vector of equivalent forces on nodes $\mathbf{f}_q$, the point loads applied on nodes, the stabilization term of the Dirichlet boundary $\mathbf{f}_g$ and the stabilizing component $\mathbf{f}_s$. For details on these components check [44].

Regarding the structural sensitivity analysis, the differentiation of the previous system of equations is needed, including the components due to the imposition boundary conditions. In this work we use a formulation presented in [45] that is based on the analytical discrete method[58, 59, 60, 61] consisting in obtaining analytical expressions of the sensitivities of the external forces and stiffness matrix. A detailed review and comparison of the different sensitivity analysis approaches can be found in[62].

The derivative of (3) with respect to any design variable a_m allows to obtain the sensitivity of the calculation

$$\left(\frac{\partial \mathbf{K}}{\partial a_m}+\frac{\partial \mathbf{K_D}}{\partial a_m}\right)\mathbf{u}+(\mathbf{K}+\mathbf{K}_D)\,\frac{\partial \mathbf{u}}{\partial a_m}=\frac{\partial \mathbf{f}_q}{\partial a_m}+\frac{\partial \mathbf{f}_g}{\partial a_m}+\frac{\partial \mathbf{f}_s}{\partial a_m} \tag{4}$$

First, starting with $\mathbf{K}$ and considering that the derivative of material properties matrix, $\mathbf{D}$, with respect to design variables is zero

$$\frac{\partial \mathbf{K}}{\partial a_m}=\int_\Omega\left[\frac{\partial \mathbf{B}^T}{\partial a_m}\mathbf{D}\mathbf{B}+\mathbf{B}^T\mathbf{D}\frac{\partial \mathbf{B}}{\partial a_m}\right]|\mathbf{J}|\mathrm{d}\Omega+\int_\Omega\left[\mathbf{B}^T\mathbf{D}\mathbf{B}\frac{\partial|\mathbf{J}|}{\partial a_m}\right]\mathrm{d}\Omega \tag{5}$$

where

- $\mathbf{B}$ is the nodal strains-displacements matrix,
- $\mathbf{D}$ is the stiffness matrix that relates stresses with strains. In this work we consider linear elasticity where, under isotropic behavior, this matrix depends only on E, the Young modulus, and ν, the Poisson ratio of the material,
- $|\mathbf{J}|$ is the determinant of the matrix $\mathbf{J}$, representing $\mathbf{J}$ the Jacobian matrix of transformation of the global coordinates (x, y, z) to the local element coordinates (ξ, η, τ).

$\frac{\partial|\mathbf{J}|}{\partial a_m}, \frac{\partial \mathbf{B}}{\partial a_m}$ are the sensitivities of $|\mathbf{J}|$ and $\mathbf{B}$ with respect to the design variable a_m, which are functions of the velocity field, $\mathbf{V}_m$, that represents the partial derivatives of the location of material points, $\mathbf{P}$, with respect to the design variables: $\mathbf{V}_m = \frac{\partial \mathbf{P}}{\partial a_m}$ (see [45]).

The derivatives of the terms introduced by the stabilization method will be

$$\begin{aligned}
\frac{\partial \mathbf{K}_D}{\partial a_m} &= \int_{\Gamma_D} \frac{\kappa^*}{h} \mathbf{C}^T \mathbf{C} \frac{\partial |\mathbf{J}|}{\partial a_m} \mathrm{d}\Gamma \\
\frac{\partial \mathbf{f}_g}{\partial a_m} &= \int_{\Gamma_D} \frac{\kappa^*}{h} \mathbf{C}^T g \frac{\partial |\mathbf{J}|}{\partial a_m} \mathrm{d}\Gamma \\
\frac{\partial \mathbf{f}_s}{\partial a_m} &= \int_{\Gamma_D} \left[\mathbf{C}^T \frac{\partial \mathbf{T}(\hat{\mathbf{u}}^h)}{\partial a_m} |\mathbf{J}| + \mathbf{C}^T \mathbf{T}(\hat{\mathbf{u}}^h) \frac{\partial |\mathbf{J}|}{\partial a_m} \right] \mathrm{d}\Gamma
\end{aligned} \tag{6}$$

where

Γ_D is the portion of the boundary where Dirichlet conditions are imposed,

κ^* is the penalty constant, being $\kappa^* = \kappa \cdot E$,

h is the element size,

$\mathbf{C}$ is the matrix of finite element interpolation if Dirichlet conditions are applied on the three displacement components x, y and z.

$$\mathbf{C} = \mathbf{N} = \begin{bmatrix}
N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 & \dots & N_{nnod} & 0 & 0 \\
0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & \dots & 0 & N_{nnod} & 0 \\
0 & 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & \dots & 0 & 0 & N_{nnod}
\end{bmatrix}$$

with $nnod$ being the number of nodes per element. Otherwise $\mathbf{C} = \mathbf{SN}$, where $\mathbf{S}_{ii} = \sum_d \delta_{id}$ would be a diagonal matrix and d is the direction where Dirichlet boundary conditions are applied.

In our approach we use the recovered tractions on Γ_D evaluated from the recovered stress field $\boldsymbol{\sigma}^*$[63] to stabilize, solving the problem iteratively updating the stress field value [44, 64], $\boldsymbol{\sigma}^*(\hat{\mathbf{u}}^h)$ being the FE recovered stress field calculated for an FE solution from a previous iteration (or mesh) $\hat{\mathbf{u}}^h$. The traction on the boundary is defined as $\mathbf{T}(\hat{\mathbf{u}}^h) = \boldsymbol{\sigma}^*(\hat{\mathbf{u}}^h) \cdot \mathbf{n}$ where $\mathbf{n}$ is the unit vector normal to the boundary, then its derivative can be written as

$$\frac{\partial \mathbf{T}(\hat{\mathbf{u}}^h)}{\partial a_m} = \frac{\partial \boldsymbol{\sigma}^*}{\partial a_m} \mathbf{n} + \boldsymbol{\sigma}^* \frac{\partial \mathbf{n}}{\partial a_m} \tag{7}$$

3.1. Objective function and constraints

Although we will consider that the objective function is the volume, other magnitudes could also be considered. This volume can be obtained adding the volume of each finite element present in the mesh, computed as:

$$V = \sum_e V_e = \sum_e \int_{\Omega^e} \mathrm{d}\Omega^e = \sum_e \int_{\Omega^e} |\mathbf{J}| \, \mathrm{d}\xi \mathrm{d}\eta \mathrm{d}\tau \tag{8}$$

The techniques discussed above to differentiate the components of the system of equation can be applied to evaluate the sensitivity of equation 8.

In our study, the constraints are expressed in terms stresses. To evaluate the stresses we consider the general expression for the calculation of stresses in continuous isoparametric elements

$$\boldsymbol{\sigma} = \mathbf{D}\mathbf{B}\mathbf{u}_h^e \tag{9}$$

$\mathbf{u}_h^e$ being the vector of nodal displacements of element e. Taking the derivative with respect to the design variable a_m, it yields

$$\frac{\partial \boldsymbol{\sigma}}{\partial a_m} = \mathbf{D}\mathbf{B}\frac{\partial \underline{\mathbf{u}}^e}{\partial a_m} + \mathbf{D}\frac{\partial \mathbf{B}}{\partial a_m}\underline{\mathbf{u}}^e \tag{10}$$

where all terms on the right can be evaluated using the development of the preceding sections. Once we have evaluated both $\boldsymbol{\sigma}$ and $\frac{\partial \boldsymbol{\sigma}}{\partial a_m}$ we can apply the construction of the smoothing field based on a recovery technique shown in [63].

3.2. Error estimator

The error associated with the finite element discretization is evaluated in this work using the Zienkiewicz and Zhu[19] error estimator as:

$$\|\mathbf{e}_{es}\|^2 = \int_\Omega (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^*)^T \mathbf{D}^{-1} (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^*) \mathrm{d}\Omega \tag{11}$$

where $\boldsymbol{\sigma}^*$ is a smoothed continuous stress field obtained by the Recovery technique[38, 63]. The resulting expression for the sensitivity analysis of the error estimator already presented in [46]:

$$\frac{\partial \|\mathbf{e}_{es}\|^2}{\partial a_m} = \sum_e \int_{\Omega^e} (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^*)^T \mathbf{D}^{-1} \left(2 \left(\frac{\partial (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^*)}{\partial a_m} \right) + \frac{(\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^*)}{|\mathbf{J}|} \frac{\partial |\mathbf{J}|}{\partial a_m} \right) |\mathbf{J}| \, \mathrm{d}\Omega^e \tag{12}$$

where $\frac{\partial(\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^*)}{\partial a_m}$ will be approximated as follows:

$$\frac{\partial(\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^*)}{\partial a_m} = \frac{\partial \boldsymbol{\sigma}_h}{\partial a_m} - \left(\frac{\partial \boldsymbol{\sigma}}{\partial a_m}\right)^* \tag{13}$$

being $\left(\frac{\partial \boldsymbol{\sigma}}{\partial a_m}\right)^*$ obtained through the same recovery procedure applied previously to $\boldsymbol{\sigma}^*$.

Equation (12) was also derived in [65] for the definition of an estimator for the discretization error in shape sensitivity analysis. In order to use an h-refinement strategy, it will also be necessary to compute the energy norm and its sensitivity with respect to each design variable. This can be evaluated considering::

$$\|\mathbf{u}_{es}\|^2 \approx \mathbf{u}^T \mathbf{K} \mathbf{u} + \|\mathbf{e}_{es}\|^2 \tag{14}$$

$$\frac{\partial \|\mathbf{u}_{es}\|^2}{\partial a_m} \approx \frac{\partial \mathbf{u}^T}{\partial a_m} \mathbf{K} \mathbf{u} + \mathbf{u}^T \frac{\partial \mathbf{K}}{\partial a_m} \mathbf{u} + \mathbf{u}^T \mathbf{K} \frac{\partial \mathbf{u}}{\partial a_m} + \frac{\partial \|\mathbf{e}_{es}\|^2}{\partial a_m} \tag{15}$$

4. Automatic *h*-adaptive mesh projection

In this contribution we use a gradient-based algorithm[66] which uses first-order sensitivities of the objective functions and constraints to evaluate the solution of (1). Using this information and the values of the design variables for the j-th geometry obtained during the iterative process ($\mathbf{a}^j$), see Figure 8a, the algorithm generates the modified values of $\mathbf{a}^j$ defining an improved design ($\mathbf{a}^{j+1}$) using

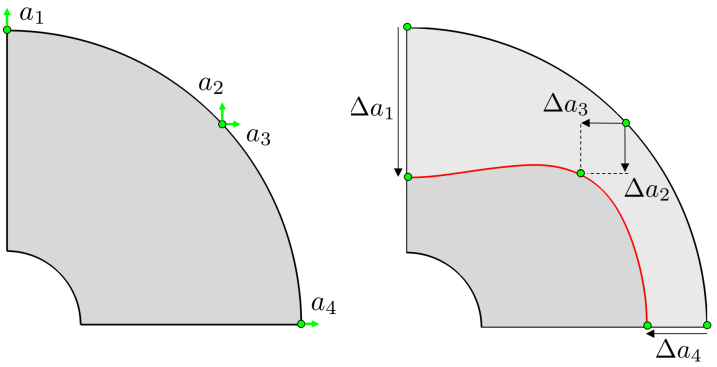
$$\mathbf{a}^{j+1} = \mathbf{a}^j + \alpha \mathbf{S}(\mathbf{a})^j \tag{16}$$

where $\mathbf{S}(\mathbf{a})^j$ is the search direction vector and α is a parameter related to the step size.

After the definition of the $(j+1)$-th geometry to be analyzed, see Figure 8b, it is necessary to construct the new analysis mesh. There have been previous developments about this using standard body-fitted FE meshes [46, 47]. In these references the information required to define a new mesh was projected from one geometry to another making use of the following expression:

$$M_{j+1} \approx M_j + \sum \left(\frac{\partial M_j}{\partial a_m}\right) \cdot \Delta a_m \tag{17}$$

where M represents any magnitude that has to be projected from geometry j to geometry $j+1$. The generation of an h-adapted mesh used in these references was



(a) Reference design (j). (b) Perturbed design ($j+1$).

Figure 8: Design evolution during optimization.

based on the use of a mesh optimality criterion, in these cases the criterion used was the minimization of the number of elements in the mesh to be created that would produce the prescribed estimated error in energy norm. This criterion is equivalent to the equidistribution of the error in energy norm on the elements of the new mesh[67]. In the following, we use a 3D generalization of this criterion presented [41].

Let's assume that $\Omega_{j,def}$ is mesh n of an h-adaptive analysis that corresponds to the geometry $j+1$ and we want to evaluate mesh $n+1$ (the new mesh) of the h-adaptive sequence, then:

$$h_{e,n}^{n+1} \approx h_e^n \left[\frac{1}{M^n}\right]^{1/2(p+1)} \left[\frac{\|\mathbf{e}^{n+1}\|}{\|\mathbf{e}^n\|}\right]^{\frac{d}{2p^2+pd}} \left[\frac{\|\mathbf{e}^{n+1}\|}{\|\mathbf{e}^n\|_e}\right]^{\frac{2}{2p+d}} \tag{18}$$

where

h_e^n is the size of the element e of the mesh n,

$h_{e,n}^{n+1}$ is the new element size of the mesh $n+1$ obtained by the subdivision of element e in the mesh n,

M^n is the number of elements in the mesh n,

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

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

$\|\mathbf{e}^n\|_e$ is the error of the element e of the mesh n,

p is the polynomial degree of the shape functions used,

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

To use this expression we have to replace $\|\mathbf{e}^n\|_e$ in (18) by the projection given in Equation (21), evaluate $\|\mathbf{e}\|_n$ as the summation of all the projected errors in elements from Equation (21), and evaluate $\|\mathbf{e}^{n+1}\|$ as

$$\|\mathbf{e}^{n+1}\| = \frac{\gamma}{100}\|\mathbf{u}_{es}^{j+1}\| \tag{19}$$

where γ is the prescribed percentage of relative error in energy norm and $\|\mathbf{u}_{es}^{j+1}\|$ is the global projected energy norm.

Hence, once a new design has been defined, the projection starts with the previous analysis mesh, defined as $\Omega_{j,\square}$ in Figure 9a, using the previously computed coordinate sensitivities. The projected position $\mathbf{r}^{j+1}$ for each node of the mesh is given by:

$$\mathbf{r}^{j+1} = \mathbf{r}^j + \sum_i^m \left(a_i^{j+1} - a_i^j\right)\left(\frac{\partial \mathbf{r}^j}{\partial a_i^j}\right) \tag{20}$$

Likewise, the estimated error in energy norm and the estimated energy norm at each element required in (18) can also be estimated by projection using the expressions

$$\|\mathbf{e}_{es}\|_{e,j+1}^2 \approx \|\mathbf{e}_{es}\|_{e,j}^2 + \sum_i^m \left(a_i^{j+1} - a_i^j\right)\frac{\partial \|\mathbf{e}_{es}\|_e^2}{\partial a_i} \tag{21}$$

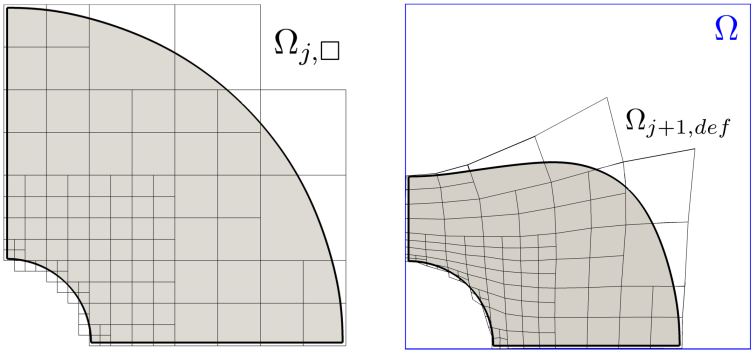
$$\|\mathbf{u}_{es}\|_{e,j+1}^2 \approx \|\mathbf{u}_{es}\|_{e,j}^2 + \sum_i^m \left(a_i^{j+1} - a_i^j\right)\frac{\partial \|\mathbf{u}_{es}\|_e^2}{\partial a_i} \tag{22}$$

These projections give an approximation to the values of the estimated error in energy norm and the energy norm that would be obtained if the next design were computed with the previous Cartesian mesh $\Omega_{j,\square}$ projected to the new geometry, represented as $\Omega_{j+1,def}$ in Figure 9b.

As in a standard remeshing procedure, we have an h-adapted mesh for geometry $j+1$ and, thanks to the extrapolation procedure, the values of energy norm and its estimated error at each element. Hence, without any further computation on geometry $j+1$, the projected estimated error and energy norm allow us to estimate the quality of the results that would be obtained through the FE analysis of geometry $j+1$ with a mesh (Figure 9b) equivalent to the one used in the previous design j (Figure 9a). If the target error prescribed for the FE analysis is lower than the projected error of the $(j+1)$-th geometry, the mesh must be h-refined using (18).

Up to this point, the mesh projection presented is comparable to the strategies used for standard body-fitted meshes[46, 47]. As we can easily observe in Figure 9b, this kind of projection yields in a discretization that is not compatible with the hierarchical Cartesian structure of cgFEM, thus losing most of the advantages related to its use.

In this paper we propose a projection strategy that will allow to generate an h-adapted analysis mesh of the new design $j+1$ keeping the Cartesian structure intact.



(a) Cartesian reference analysis mesh, $\Omega_{j,\square}$.

(b) Projected (non-Cartesian) mesh on geometry $j+1$.

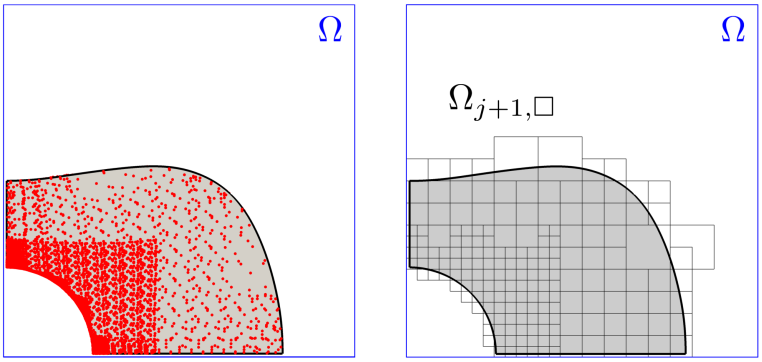
Figure 9: Mesh projection procedure.

This strategy simply requires to project the element size evaluated using (18) for the elements of $\Omega_{j+1,def}$ (Figure 9b) to the embedding domain Ω. To do this we assign this element size to the Gauss points of each element and project all the integration points of $\Omega_{j+1,def}$ to Ω. These projected integration points containing element size information can be trivially located into the elements of a uniform Cartesian grid of the prescribed level. Then these Cartesian elements are recursively refined until the size of each element is smaller than the minimum element sizes defined by the Gausss points contained in the element, leading to an h-adapted Cartesian grid (see 10b)

From this perspective, projection, through sensitivity analysis, can transform a posteriori error estimation into a preprocess tool able to generate an h-adapted mesh for the new design, recycling calculations obtained on previous stages of the optimization process.

5. Numerical examples

In this Section we will show three numerical analyses. The first one will be used to show the performance of the direct solver used to evaluate solution of the systems of equations when applying different reordinations to the matrices. The remaining two problems will be devoted to asses the optimization methodology presented in this contribution. The last two optimization analyses will test the accuracy of the



(a) Perturbed integration points.

(b) Projected Cartesian mesh, $\Omega_{j+1,\square}$.

Figure 10: Mesh projection procedure.

cgFEM implementation coupled with the optimization algorithm using an academical problem with different number of design variables.

The model proposed for this study is a thick-wall infinite cylinder loaded with internal pressure. The geometrical model for this problem is represented in Figure 11. A linear-elasticity 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 symmetries. The internal and external surfaces are of radius r and R, with $R_{int} = 5$ and $R_{ext} = 20$. Young's modulus is $E = 1000$, Poisson's ratio is $\nu = 0.3$ and the applied load is $P = 1$.

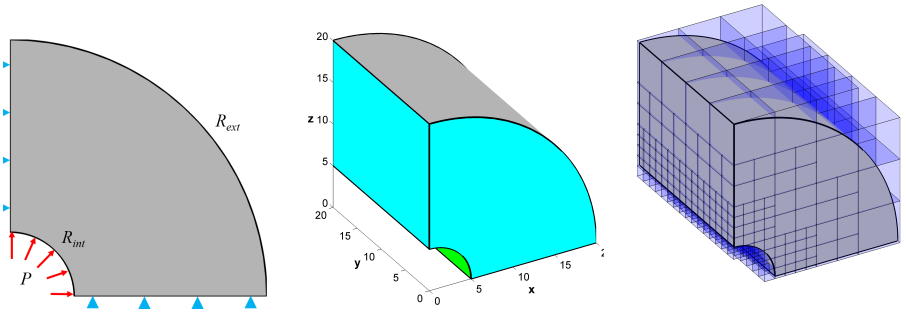
The exact solution for displacements and stresses is given by:

$$u_r = \frac{P(1+\nu)}{E(k^2-1)}\left(r\,(1-2\nu) + \frac{R_{ext}^2}{r}\right), \qquad u_y = 0 \tag{23}$$

$$\sigma_r = \frac{P}{k^2-1}\left(1 - \frac{R_{ext}^2}{r^2}\right), \qquad \sigma_\phi = \frac{P}{k^2-1}\left(1 + \frac{R_{ext}^2}{r^2}\right), \qquad \sigma_y = \nu\,(\sigma_r + \sigma_\phi) \tag{24}$$

where $k = R_{ext}/R_{int}$, $r = \sqrt{x^2+z^2}$.

For the optimization analyses we will substitute the constant R_{ext} for a unique design variable or we will define a set of design variables to define arbitrary external surfaces.



(a) Front view with boundary conditions. (b) 3D model representation. (c) Example of analysis mesh.

Figure 11: Model of a cylinder under internal pressure.

5.1. Performance of the direct solver

As explained in Section 2.2, the resolution of the system of equations with direct solvers is a time consuming task that can be lighten using a proper reordering of the matrices involved. To solve the linear system of equations in (3), we have run the tests in MATLAB® 2014a, using the standard *backslash* solver provided in this compilation. In this example, we will compare four different reordering strategies:

- *Nested Domain Decomposition (NDD)*: in this case, we use the NDD reordering presented in Section 2.2.
- *Reference*: this strategy consists in solving the system without any previous reordering.
- *Approximate Minimum Degree (AMD) permutation*: if the degree of a node in a graph is the number of connections to that node, the AMD algorithm[68] generates an ordering based on how these degrees are altered during Cholesky factorization.
- *Symmetric AMD permutation (SYM-AMD)[69]*: this algorithm performs an AMD reordination taking into account the symmetry of the matrix.
- *Column AMD permutation (COL-AMD)[70]*: this algorithm returns the column approximate minimum degree permutation vector of the matrix. This is the default algorithm used by MATLAB®.

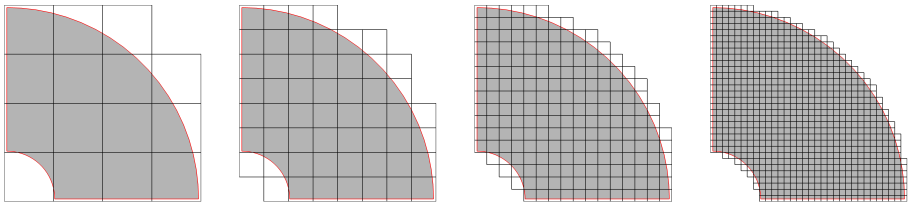


Figure 12: 2D view of 3D uniform meshes with different element size.

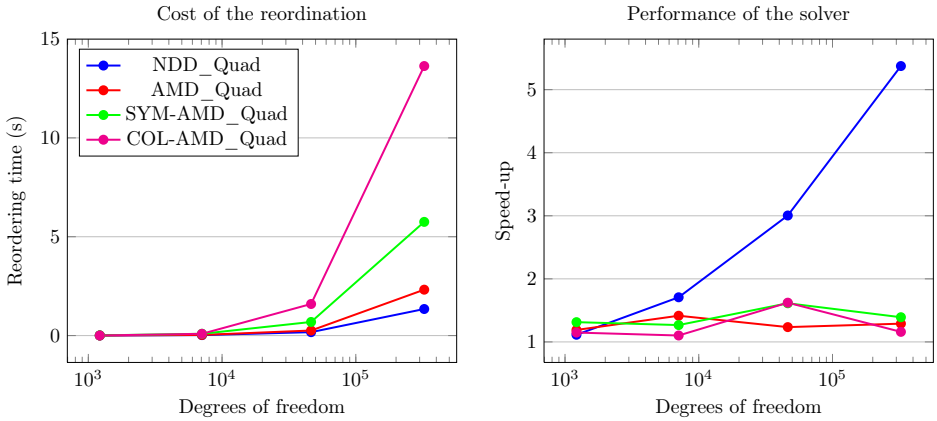


Figure 13: Behavior of different reordering techniques. Left: reordering times. Right: speed-up in the resolution of the system of equations with respect to the reference (no reordering).

For the analysis we will study a set of uniformly refined meshes of 20-node triquadratic elements. The meshes used in this simulation can be seen in Figure 12.

On the left plot of Figure 13, we can observe the computational cost related with the reordering function of the degrees of freedom present in the meshes. This computational cost takes into account both, finding the reordered indexes and the reordering process. The right plot shows the computational cost related to the resolution of the system of equations in terms of the speed-up achieved with respect to the reference, i.e., with no reordering. This means that a value larger than 1 represents the reduction of cost with respect with the reference calculation with no reordering.

From Figure 13 we can extract the several conclusions. We can notice how, for small problems (two first meshes), the differences in computational cost between the different alternatives are not significant. However, for larger problems we can clearly observe how the computational cost related to NDD reordering is clearly superior to the alternatives studied.

So, when using NDD, the time devoted to reorder the system of equations and to solve it is reduced, allowing for the resolution of larger systems of equations with the same resources. The reason behind this positive performance of the proposed reordering technique can be that the NDD reordering could represent an optimal reordering, as it takes into account the topology of the mesh.

5.2. Thick-wall infinite cylinder loaded with internal pressure defined by 1 design variable

Let us consider R_{ext} as the design variable that defines the cylinder presented in Figure 11. Our objective in this problem is to minimize the volume of the model under internal pressure P applied on the circular internal surface, with unknown external surface, where the Von Mises stresses must be below the yield stress S_y. For the parameters defined above and for $S_y = 2$, the optimal analytical solution corresponds to $b = 13.681300358237177$ and the corresponding volume is $V = 2547.485744735241$.

Design variable	Initial value	Data range
R_{ext}	17	$[9 - 20]$

Table 1: Thick-wall infinite cylinder defined by 1 design variable. Design variable data.

The first analysis consist of using sets of uniform meshes of 20-node tri-quadratic elements with different element size. We will use meshes of levels 3, 4 and 5 that correspond with the three last levels of refinement represented in Figure 12. By doing this, we will evaluate how varying the discretization affects the accuracy and the computational cost.

In Figure 14 we can observe the evolution of the relative error in volume evaluated as $\eta_V(\%) = \frac{|V_h - V|}{V} \cdot 100$ where V_h is the volume integrated with the finite element mesh and V is the exact volume of the model. The plot shows the convergence of the optimization process to a clearly suboptimal solution when using coarse meshes. In order to get closer to the theoretical optimal solution finer meshes have to be used, however this decision will involve an increase of the computational cost.

In Table 2 we can see the average discretization estimated error in energy norm per individual and the average computational cost per individual. We observe how, in order to reduce the discretization error, the computational cost of each individual increases significantly. This conclusion justifies the use of h-adaptive meshes.

We repeat the analysis but using h-adapted meshes and the projection technique presented in Section 4. In Figure 15 we can observe the behavior of h-adapted meshes with tri-quadratic elements (hAdapMeshing) and projected h-adapted meshes with the same elements (ProjMeshing).

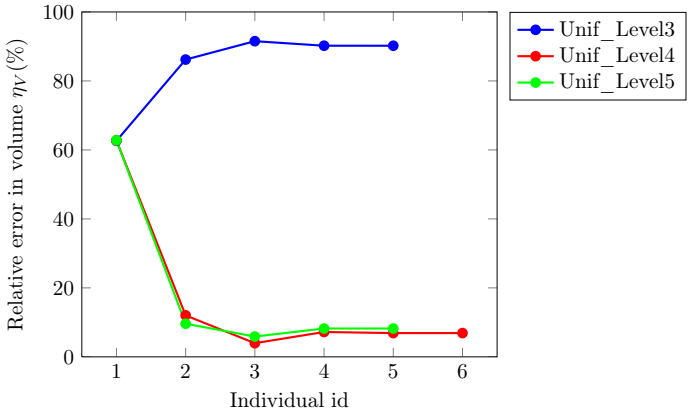


Figure 14: Evolution of the error in the objective function (volume) with respect to the analytical solution. Uniform meshes.

Type of mesh	Computational cost (s)	Estimated discretization error
Unif_Level3	9.05	7.99%
Unif_Level4	25.13	2.41%
Unif_Level5	229.81	0.67%

Table 2: Computational results for uniform meshes. Average values of computational cost and estimated discretization error in energy norm.

Table 3 shows the details of the analyses in terms of average computational cost and estimated discretization error of the meshes. In this case, the h-adapted meshes achieve a level of accuracy similar to the accuracy obtained with the level 5 uniform mesh, but in a fraction of the time. In addition, the projected meshes cut the computational cost of the h-adaptive process in around 25%.

Type of mesh	Computational cost (s)	Estimated discretization error
hAdapMeshing	52.23	0.87%
ProjMeshing	39.53	0.99%

Table 3: Thick-wall cylinder defined by 1 design variables. Computational results for h-adapted and projected meshes.

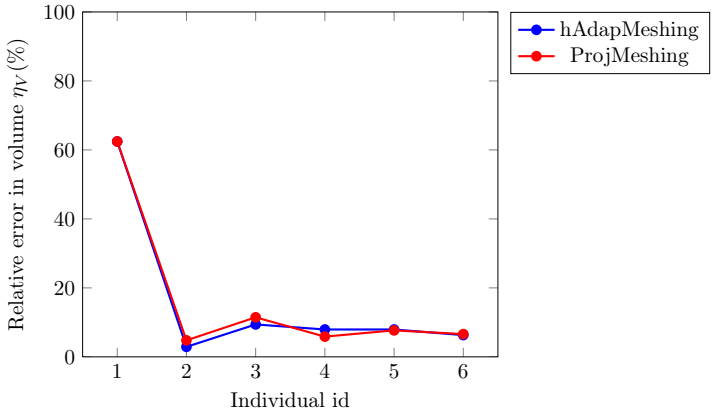


Figure 15: Evolution of the error in the objective function (volume) with respect to the analytical solution. h-adapted and projected meshes.

5.3. Thick-wall infinite cylinder loaded with internal pressure defined by 4 design variables

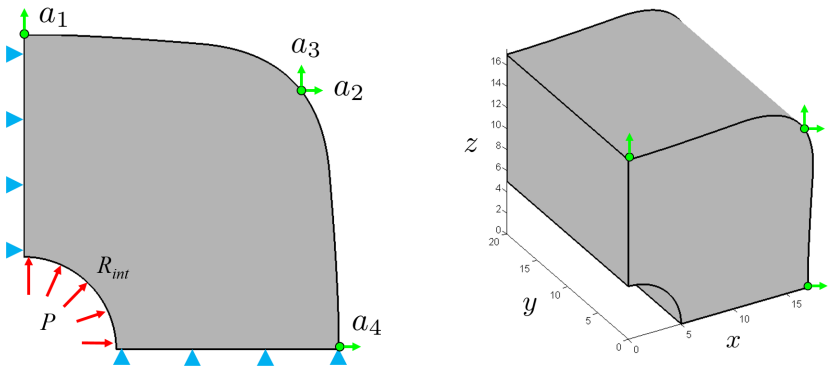
In this example we modify the previous model introducing several design variables. The initial shape is shown in Figure 16. The shape optimization problem consists of finding the best shape for the external boundary defined by four design variables, corresponding to coordinates of the points used to define the external boundary.

The mechanical properties for this problem correspond to those exposed at the beginning of the section. The initial values of the design variables and their allowed data range and constraints are shown in Table 4.

Design variable	Initial value	Data range	Constraints on the design variables
a_1	17	$[10-20]$	None
a_2	16	$[8-17]$	$a_2 \leq a_4 - 1$
a_3	16	$[8-17]$	$a_3 \leq a_1 - 1$
a_4	17	$[10-20]$	None

Table 4: Thick-wall infinite cylinder defined by 4 design variable. Design variables data.

Figure 17 shows the evolution of the relative error in volume for an optimization process performed using standard h-refined meshes and another carried out using projected meshes. We can observe a common convergence path regardless of the different discretizations used.



(a) Front view with boundary conditions.

(b) 3D model representation.

Figure 16: Model of a cylinder under internal pressure defined by 4 design variables.

In Table 5 we can see the average discretization estimated error in energy norm per individual and the computational cost per individual. The computational costs include the simulations performed to evaluate the sensitivities. We observe that for a comparable level of discretization error we save close to 20% of time when using mesh projection.

Type of mesh	Computational cost (s)	Estimated discretization error
hAdapMeshing	151.81	1.22%
ProjMeshing	124.90	1.46%

Table 5: Thick-wall cylinder defined by 4 design variables. Computational results for h-adapted and projected meshes.

Figure 18 shows several of the individuals analyzed during the process including the first and the last one (51). In addition, the theoretical optimal solution has been drawn to clarify the evolution of the procedure.

5.4. Connecting rod defined by 8 design variables

The objective of this problem is to minimize the volume of a connecting rod without violating the given maximum Von Mises stress. Because of the symmetry, only a fourth of the component is modeled. The geometry of the initial design and the boundary conditions are shown in Figure 19. The geometry parameters are $AB = 11$,

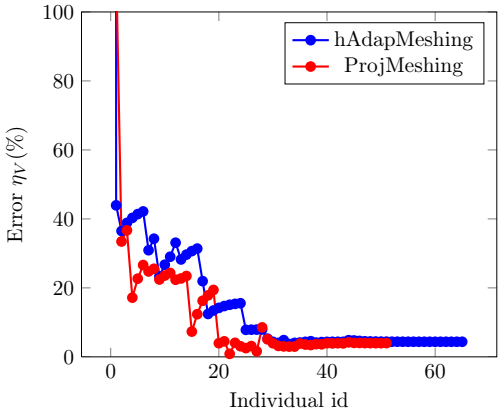


Figure 17: Evolution of the error in the objective function (volume) with respect to the analytical solution. h-adapted and projected meshes.

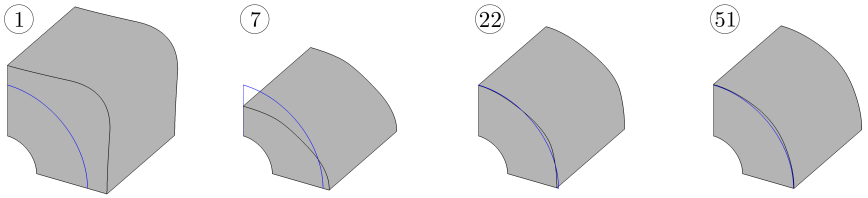


Figure 18: Samples of individuals from the optimization procedure. The index indicate the number of model during the process.

$C = 4$, $AD = 20$, $DE = 4$, $F = 1.5$, $DG = 7$, $HG = 5.5$. The Young's modulus is $E = 10^5$, and Poisson's ratio $\nu = 0.333$. The pressure is $P = 100$ in the normal direction of the half arc as shown in Figure 19.

The design boundary is the surface HG. The end point H is fixed while eight points are used to interpolate HG. The vertical positions of the eight interpolation points on the design surface are set as design variables (see Figure 20). The allowable von Mises stress is $\sigma_{VM} = 900$.

The initial values of the design variables and their allowed data range are shown in Table 4.

Table 7 shows the average discretization estimated error in energy norm per individual and the computational cost per individual. We observe for this problem how the optimization procedure based in mesh projection cuts slightly more of a 20% of the time per individual.

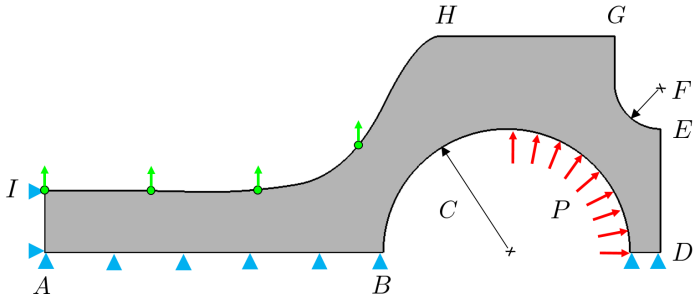


Figure 19: Front view of the connecting rod problem with boundary conditions.

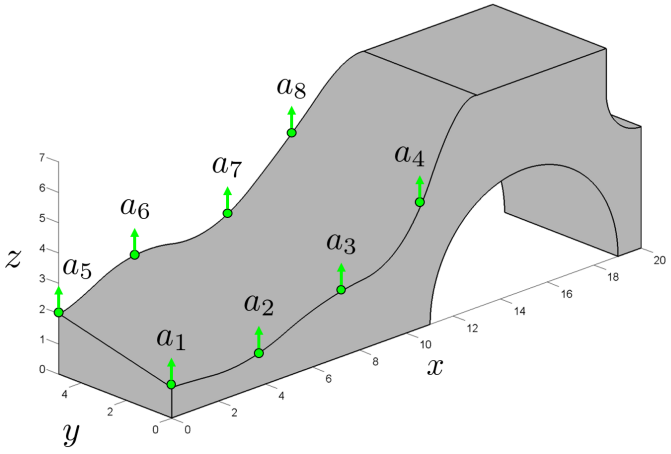


Figure 20: 3D model representation showing the 8 design variables.

Design variable	Initial value	Data range
a_1, a_5	7	$[1-7]$
a_2, a_6	7	$[1-7]$
a_3, a_7	7	$[1.2-7]$
a_4, a_8	7	$[2-7]$

Table 6: Connecting rod defined by 8 design variable. Design variables data.

Figure 21 shows the Von Mises stress fields for the initial configuration of the model opposed to the field obtained for the optimal solution provided by the shape optimization algorithm.

Type of mesh	Computational cost (s)	Estimated discretization error
hAdapMeshing	607.84	2.83%
ProjMeshing	471.02	2.62%

Table 7: Connecting rod defined by 8 design variable. Computational results for h-adapted and projected meshes.

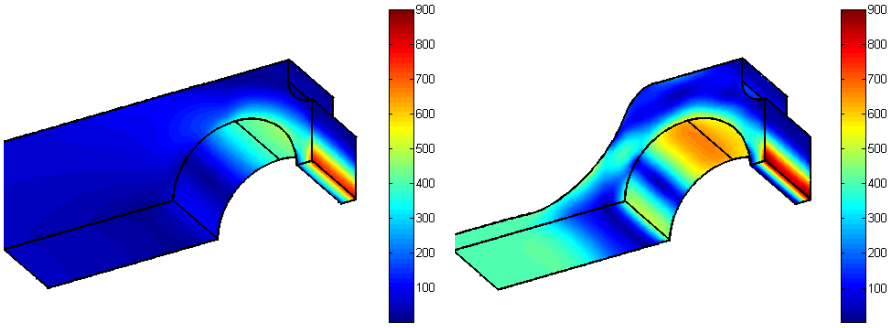


Figure 21: Von Mises stress fields: (left) initial configuration results and (right) configuration obtained using projected meshes.

6. Conclusions

Several tools to make gradient-based optimization procedures have been proposed. First, information sharing procedures that can be easily applied reducing the number of calculations needed. Also, the Nested Domain Decomposition reordering technique has been developed for a 3D code and tested. The NDD provides an optimal reordering of the global system of equations with minimum computational cost in comparison with other techniques. In addition, the speed-up shown during the resolution of the systems of equations is significant, allowing the efficient usage of the computational resources. Finally, an h-adaptive mesh projection strategy has been adapted to the immersed boundary environment. The projection avoids the need to generate a suitable discretization after following a full refinement process. The discretizations generated with this procedure has been demostrated as effective, in terms of convergence, than the standard h-refined meshes, but with an important reduction of the computational cost per individual.

Acknowledgments

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

References

[1] Braibant V, Fleury C. Shape optimal design using b-splines. *Computer Methods in Applied Mechanics and Engineering* 1984; **44**(3):247–267. 1

[2] Haftka RT, Grandhi RV. Structural shape optimization: A survey. *Computer Methods in Applied Mechanics and Engineering* 1986; **57**(1):91–106. 1

[3] Ródenas JJ, Bugeda G, Albelda J, Oñate E. On the need for the use of error-controlled finite element analyses in structural shape optimization processes. *International Journal for Numerical Methods in Engineering* 2011; **87**(11):1105–1126. 1

[4] Bennett JA, Botkin ME. Structural Shape Optimization with Geometric Problem Description and Adaptive Mesh Refinement. *AIAA Journal* 1985; **23**(3):459–464. 1

[5] Chang K, Choi KK. A geometry-based parameterization method for shape design of elastic solids. *Mechanics of Structures and Machines* 1992; **20**(2):215–252. 1

[6] Hughes TJR, Cottrell JA, Bazilevs Y. Isogeometric Analysis: CAD, Finite Elements, NURBS, Exact Geometry, and Mesh Refinement. *Computer Methods in Applied Mechanics and Engineering* 2005; **194**:4135–4195. 1

[7] Nguyen VP, Anitescu C, Bordas SPA, Rabczuk T. Isogeometric analysis: An overview and computer implementation aspects. *Mathematics and Computers in Simulation* 2015; **117**:89–116. 1

[8] Zhang Y, Wang W, Hughes TJR. Conformal Solid T-spline Construction from Boundary T-spline Representations. *Computational Mechanics* 2013; **6**(51):1051–1059. 1

[9] Escobar JM, Montenegro R, Rodríguez E, Cascón JM. The meccano method for isogeometric solid modeling and applications. *Engineering with Computers* 2014; **30**(3):331–343. 1

[10] Liu L, Zhang Y, Hughes TJR, Scott MA, Sederberg TW. Volumetric T-spline Construction using Boolean Operations. *Engineering with Computers* 2014; **30**(4):425–439. 1

[11] Cho S, Ha SH. Isogeometric shape design optimization: exact geometry and enhanced sensitivity. *Structural and Multidisciplinary Optimization* 2009; **38**(1):53–70. 1

[12] Ha SH, Choi K, Cho S. Numerical method for shape optimization using T-spline based isogeometric method. *Structural and Multidisciplinary Optimization* 2010; **42**(3):417–428. 1

[13] Qian X. Full analytical sensitivities in NURBS based isogeometric shape optimization. *Computer Methods in Applied Mechanics and Engineering* 2010; **199**(29–32):2059–2071. 1

[14] Li K, Qian X. Isogeometric analysis and shape optimization via boundary integral. *Computer-Aided Design* 2011; **43**(11):1427–1437. 1

[15] Lian H, Kerfriden P, Bordas SPA. Implementation of regularized isogeometric boundary element methods for gradient-based shape optimization in two-dimensional linear elasticity. *International Journal for Numerical Methods in Engineering* 2016; **106**(12):972–1017. 1

[16] Kikuchi N, Chung KY, Torigaki T, Taylor JE. Adaptive finite element methods for shape optimization of linearly elastic structures. *Computer Methods in Applied Mechanics and Engineering* 1986; **57**(1):67–89. 1

[17] Yao T, Choi KK. 3-d shape optimal design and automatic finite element regridding. *International Journal for Numerical Methods in Engineering* 1989; **28**(2):369–384. 1

[18] Riehl S, Steinman P. An integrated approach to shape optimization and mesh adaptivity based on material residual forces. *Computer Methods in Applied Mechanics and Engineering* 2014; **278**:640–663. 1

[19] Zienkiewicz OC, Zhu JZ. A Simple Error Estimator and Adaptive Procedure for Practical Engineering Analysis. *International Journal for Numerical Methods in Engineering* 1987; **24**(2):337–357. 1, 3.2

[20] González-Estrada OA, Nadal E, Ródenas JJ, Kerfriden P, Bordas SPA, Fuenmayor FJ. Mesh adaptivity driven by goal-oriented locally equilibrated superconvergent patch recovery. *Computational Mechanics* 2014; **53**(5):957–976. 1

[21] Peskin CS. Numerical Analysis of Blood Flow in the Heart. *Journal of Computational Physics* 1977; **25**:220–252. 1

[22] Zhang L, Gerstenberger A, Wang X, Liu WK. Immersed Finite Element Method. *Computer Methods in Applied Mechanics and Engineering* 2004; **293**(21):2051–2067. 1

[23] Haslinger J, Jedelsky D. Genetic algorithms and fictitious domain based approaches in shape optimization. *Struc. Optim.* 1996; **12**:257–264. 1

[24] Kunisch K, Peichl G. Numerical gradients for shape optimization based on embedding domain techniques. *Comput. Optim.* 1996; **18**:95–114. 1

[25] Kim NH, Chang Y. Eulerian shape design sensitivity analysis and optimization with a fixed grid. *Computer Methods in Applied Mechanics and Engineering* 2005; **194**(30–33):3291–3314. 1

[26] Najafi AR, Safdari M, Tortorelli DA, Geubelle PH. A gradient-based shape optimization scheme using an interface-enriched generalized FEM. *Computer Methods in Applied Mechanics and Engineering* 2015; **296**:1–17. 1

[27] Riehl S, Steinmann P. On structural shape optimization using an embedding domain discretization technique. *International Journal for Numerical Methods in Engineering* 2016; . 1

[28] García-Ruíz MJ, Steven GP. Fixed grid finite elements in elasticity problems. *Engineering Computations* 1999; **16**(2):145–164. 1

[29] Dunning PD, Kim HA, Mullineux G. Investigation and improvement of sensitivity computation using the area-fraction weighted fixed grid FEM and structural optimization. *Finite Elements in Analysis and Design* 2011; **47**(8):933–941. 1

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

[31] Düster A, Parvizian J, Yang Z, Rank E. The finite cell method for three-dimensional problems of solid mechanics. *Computer Methods in Applied Mechanics and Engineering* 2008; **197**(45-48):3768–3782. 1

[32] Schillinger D, Ruess M. 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* 2015; **22**(3):391–455. 1

[33] Meagher D. Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer. *Technical Report IPL-TR-80-11 I*, Rensselaer Polytechnic Institute 1980. 1, 2

[34] Jackins CL, Tanimoto SL. Oct-tree and their use in representing three-dimensional objects. *Computer Graphics and Image Processing* 1980; **14**(3):249–270. 1, 2

[35] Doctor LJ, Torborg JG. Display techniques for octree-encoded objects. *IEEE Comput. Graph. Appl.* 1981; **1**(3):29–38. 1, 2

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

[37] Fries TP, Omerović S. Higher-order accurate integration of implicit geometries. *International Journal for Numerical Methods in Engineering* 2016; **106**(5):323–371. 1

[38] Nadal E, Ródenas JJ, Albelda J, Tur M, Tarancón JE, Fuenmayor FJ. Efficient Finite Element Methodology based on Cartesian Grids: Application to Structural Shape Optimization. *Abstract and Applied Analysis* 2013; **2013**. 1, 2.2, 3.2

[39] Nadal E. *Cartesian Grid FEM (cgFEM): High Performance h-adaptive FE Analysis with Efficient Error Control. Application to Structural Shape Optimization. PhD Thesis*. Universitat Politècnica de València, 2014. 1

[40] Marco O, Sevilla R, Zhang Y, Ródenas JJ, Tur M. Exact 3D boundary representation in finite element analysis based on Cartesian grids independent of the geometry. *International Journal for Numerical Methods in Engineering* 2015; **103**:445–468. 1, 2, 2, 2

[41] Marco O, Ródenas JJ, Navarro-Jiménez JM, Tur M. Robust h-adaptive meshing strategy for arbitrary cad geometries in a cartesian grid framework. *Computers & Structures* 2017; **Submitted**. 1, 4

[42] Sevilla R, Fernández-Méndez S, Huerta A. NURBS-enhanced Finite Element Method (NEFEM): A Seamless Bridge Between CAD and FEM. *Archives of Computational Methods in Engineering* 2011; **18**(4):441–484. 1, 2

[43] Sevilla R, Fernández-Méndez S, Huerta A. 3D-NURBS-enhanced Finite Element Method (NEFEM). *International Journal for Numerical Methods in Engineering* 2011; **88**(2):103–125. 1, 2

[44] Tur M, Albelda J, Marco O, Ródenas JJ. Stabilized Method to Impose Dirichlet Boundary Conditions using a Smooth Stress Field. *Computer Methods in Applied Mechanics and Engineering* 2015; **296**:352–375. 1, 2, 3, 3, 3

[45] Marco O, Ródenas JJ, Fuenmayor FJ, Tur M. An extension of shape sensitivity analysis to an immersed boundary method based on cartesian grids. *Computational Mechanics* 2017; **Submitted**. 1, 3, 3, 3

[46] Bugeda G, Oliver J. A General Methodology for Structural Shape Optimization Problems Using Automatic Adaptive Remeshing. *International Journal for Numerical Methods in Engineering* 1993; **36**(18):3161–3185. 1, 3.2, 4, 4

[47] Bugeda G, Ródenas JJ, Oñate E. An integration of a low cost adaptive remeshing strategy in the solution of structural shape optimization problems using evolutionary methods. *Computers & Structures* 2008; **86**(13–14):1563–1578. 1, 4, 4

[48] Abel JF, Shephard MS. An algorithm for multipoint constraints in finite element analysis. *International Journal for Numerical Methods in Engineering* 1979; **14**(3):464–467. 2

[49] Farhat C, Lacour C, Rixen D. Incorporation of linear multipoint constraints in substructure based iterative solvers. Part 1: a numerically scalable algorithm. *International Journal for Numerical Methods in Engineering* 1998; **43**(6):997–1016. 2

[50] Kajiya JT. Ray Tracing Parametric Patches. *SIGGRAPH Comput. Graph.* 1982; **16**(3):245–254. 2

[51] Toth DL. On Ray Tracing Parametric Surfaces. *SIGGRAPH Comput. Graph.* 1985; **19**(3):171–179. 2

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

[53] Nishita T, Sederberg TW, Kakimoto M. Ray Tracing Trimmed Rational Surface Patches. *SIGGRAPH Comput. Graph.* 1990; **24**(4):337–345. 2

[54] Barth W, Stürzlinger W. Efficient ray tracing for Bezier and B-spline surfaces. *Computers & Graphics* 1993; **17**(4):423–430. 2

[55] Ródenas JJ, Tarancón JE, Albelda J, Roda A, Fuenmayor FJ. Hierarchical Properties in Elements Obtained by Subdivision: a Hierarquical h-adaptivity Program. *Adaptive Modeling and Simulation 2005*, Díez P, Wiberg NE (eds.), 2005. 2.2

[56] Ródenas JJ, Corral C, Albelda J, Mas J, Adam C. Nested domain decomposition direct and iterative solvers based on a hierarchical h-adaptive finite element code. *Adaptive Modeling and Simulation 2007*, Runesson K, Díez P (eds.), Internacional Center for Numerical Methods in Engineering (CIMNE), 2007; 206–209. 2.2

[57] Zienkiewicz OC, Taylor RL, Zhu JZ ((eds.)). *The Finite Element Method: Its Basis and Fundamentals*. Butterworth-Heinemann: Oxford, 2013. 3

[58] Kibsgaard S. Sensitivity analysis-the basis for optimization. *International Journal for Numerical Methods in Engineering* 1992; **34**(3):901–932. 3

[59] Poldneff MJ, Rai IS, Arora JS. Implementation of design sensitivity analysis for nonlinear structures. *AIAA Journal* 1993; **31**(11):2137–2142. 3

[60] Pandey PC, Bakshi P. Analytical response sensitivity computation using hybrid finite elements. *Computers & Structures* 1999; **71**(5):525–534. 3

[61] Moita JS, Infanta J, Mota CM. Sensitivity analysis and optimal design of geometrically non-linear laminated plates and shells. *Computers & Structures* 2000; **76**(1–3):407–420. 3

[62] van Keulen F, Haftka R, Kim N. Review of options for structural design sensitivity analysis. Part I: linear systems. *Computer Methods in Applied Mechanics and Engineering* 2005; **194**(30-33):3213–3243. 3

[63] Ródenas JJ, Tur M, Fuenmayor FJ, Vercher A. Improvement of the superconvergent patch recovery technique by the use of constraint equations: the SPR-C technique. *International Journal for Numerical Methods in Engineering* 2007; **70**(6):705–727. 3, 3.1, 3.2

[64] Tur M, Albelda J, Nadal E, Ródenas JJ. Imposing dirichlet boundary conditions in hierarchical cartesian meshes by means of stabilized lagrange multipliers. *International Journal for Numerical Methods in Engineering* 2014; **98**(6):399–417. 3

[65] Fuenmayor FJ, Oliver JL, Ródenas JJ. Extension of the Zienkiewicz-Zhu error estimator to shape sensitivity analysis. *International Journal for Numerical Methods in Engineering* 1997; **40**(8):1413–1433. 3.2

[66] Nocedal J, Wright SJ. *Numerical Optimization, Second Edition*. Springer-Verlag New York, 2006. 4

[67] Fuenmayor FJ, Oliver JL. Criteria to achieve nearly optimal meshes in the h-adaptive finite element mehod. *International Journal for Numerical Methods in Engineering* 1996; **39**(23):4039–4061. 4

[68] Amestoy P, Davis T, Duff I. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications* 1996; **17**(4):886–905. 5.1

[69] The Mathworks, Inc., Natick, Massachusetts. *MATLAB version 8.3.0.532 (R2014a), Documentation* 2014. 5.1

[70] Davis TA, Gilbert JR, Larimore S, Ng E. An approximate column minimum degree ordering algorithm. *ACM Transactions on Mathematical Software* 2004; **30**(3):353–376. 5.1