

Document downloaded from:

<http://hdl.handle.net/10251/103270>

This paper must be cited as:

Marco, O.; Ródenas, J.; Albelda Vitoria, J.; Nadal, E.; Tur Valiente, M. (2017). Structural shape optimization using Cartesian grids and automatic h-adaptive mesh projection. *Structural and Multidisciplinary Optimization*. 1-21. doi:10.1007/s00158-017-1875-1



The final publication is available at

<https://doi.org/10.1007/s00158-017-1875-1>

Copyright Springer-Verlag

Additional Information

# Structural shape optimization using Cartesian grids and automatic $h$ -adaptive mesh projection

Onofre Marco · Juan José Ródenas · José Albelda · Enrique Nadal · Manuel Tur

**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.

**Keywords** Cartesian grids ·  $h$ -refinement · Shape optimization · NEFEM

---

O. Marco · J.J. Ródenas · J. Albelda · E. Nadal · M. Tur  
Centro de Investigación en Ingeniería Mecánica (CIIM), Universitat Politècnica de València,  
Valencia 46022, Spain  
E-mail: onofremarco@gmail.com

## 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:

$$\begin{aligned} & \text{minimize} && f(\mathbf{a}) \\ & \text{where} && \mathbf{a} = \{a_m\} \quad m = 1, \dots, q \\ & \text{verifying} && g_n(\mathbf{a}) \leq 0 \quad n = 1, \dots, r \\ & && \text{and} \quad h_l(\mathbf{a}) = 0 \quad l = 1, \dots, s \end{aligned} \tag{1}$$

being  $f$  the objective function,  $a_m$  are the design variables,  $g_n$  are inequality constraints and the values  $h_l$  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 (FEA).

There are different approaches and many codes to solve the optimization problem. However, some problems in this context, identified long time ago [6, 25], 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 FEA, 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 (FE) computation and its impact on the solution of the optimization problem was analyzed in [61].

Two main concepts evolved to bypass these drawbacks. On the one hand, the design update procedure can be assigned to geometry model [5, 9]. In this case, the nodal coordinates manipulation within the FE 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 (IGA) [27, 47]. However, in its finite element form, generating an analysis-suitable solid discretization is an open topic [16, 38, 72]. Some works on IGA shape optimization can be found in [10, 24, 36, 37, 55].

On the other hand, the FE model could be updated through the optimization procedure to improve the accuracy of the numerical simulation results or to enhance the element quality [32, 56, 70]. This may, for instance, take the form of adaptive mesh refinement based on error estimation in energy norm [73] or goal oriented adaptivity [23]. 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 (IB) 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 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 FE 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 IB scope. Among many other names used to describe these techniques where the mesh does not match the domain's geometry, we have the Immersed Boundary Method (IBM) [52] and the Immersed Finite Element Method (IFEM) [71]. These methods have been studied by a number of authors for very different problems including, of course, shape optimization [26, 33, 35, 46, 57].

Nevertheless, the attractive advantages of IB approaches 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 FE 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 [14, 21].

A more selective approach is employed in the Finite Cell framework [15, 51, 62]. Therein, for all intersected elements, a number of integration points are provided employing hierarchical *octree* data structures [13, 28, 42] 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 [34] or techniques used where the geometry is defined implicitly by level sets [18]. Even in the first of these approaches, where the isoparametric 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 for structural shape optimization in an embedding domain setting. The methodol-

ogy is based in the Cartesian grid FEM (cgFEM) successfully implemented in 2D [44, 45] and 3D [39, 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 [39, 63, 65] 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 [69].

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 [40]. This formulation takes into account the derivatives of the stabilized formulation implemented for the imposition of boundary conditions [69].

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 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 [7] and evolutionary [8] optimization methods, oriented to standard body-fitted FEM.

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 6. This contribution ends with the conclusions in Section 7.

## 2 Cartesian grids for optimization

This work has been developed as a logical continuation of [39, 41] 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  $\Omega$  such that an open bounded domain  $\Omega_{\text{phys}}$  fulfills  $\Omega_{\text{phys}} \subset \Omega$ . 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  $\Omega$  interacting with  $\Omega_{\text{Phys}}$ .

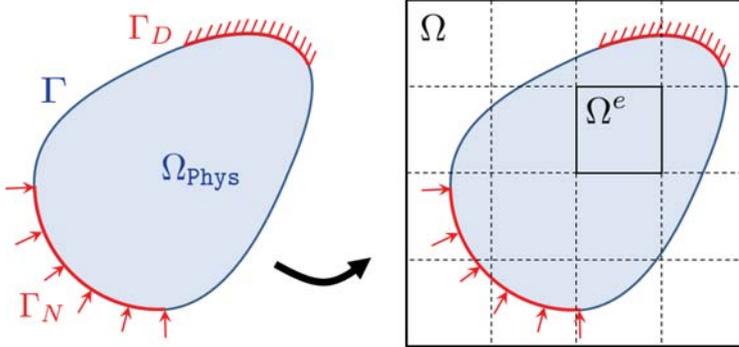


Fig. 1: Immersed Boundary Method environment. Domain  $\Omega_{\text{Phys}}$  within the embedding domain  $\Omega$ .

The discretization of the embedding domain is based on a sequence of uniformly refined Cartesian meshes to mesh the domain  $\Omega$  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 [13, 28, 42].

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) [2, 17] 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 [39, 41] we proposed a robust procedure to find intersections, between the surfaces of the boundary and the axes of the Cartesian grids, based on ray-tracing techniques [4, 29, 48, 66, 67].

Internal elements are standard FE elements whose affinity with respect to the embedding domain  $\Omega$  is exploited in order to save computational cost. Regarding the integration of intersected elements in cgFEM, we proposed a strategy in [39] 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 [63, 65] that takes into account the exact geometry defined by the CAD model.

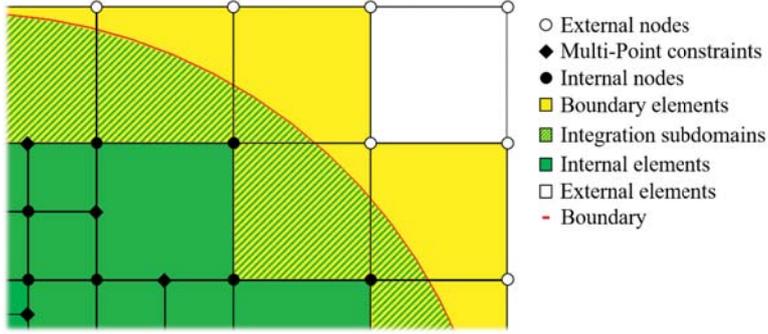
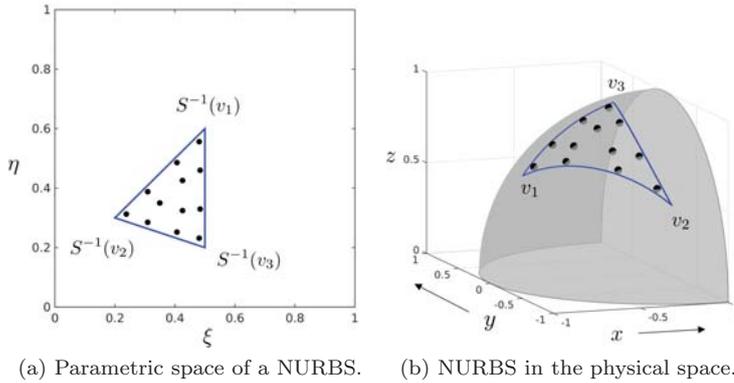


Fig. 2: Components of an immersed boundary environment.

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 FEM were demonstrated for a variety of problems, see [64].

Let  $A_{\text{phys}}$  a tetrahedral face lying on the boundary parametrized by the NURBS  $S$ , and  $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$  its three vertices, see Figure 3. A straight-sided triangle  $A_{\text{param}}$  in the parametric space of the surface is defined by the parametric coordinates of the vertices  $S^{-1}(\mathbf{v}_1), S^{-1}(\mathbf{v}_2)$ , and  $S^{-1}(\mathbf{v}_3)$ . The transformation to a curved face,  $A_{\text{phys}}$ , is defined as the image of the straight-sided triangle  $A_{\text{param}}$  by the NURBS parametrization  $S$ ,

$$A_{\text{phys}} := S(A_{\text{param}}) \quad (2)$$



(a) Parametric space of a NURBS. (b) NURBS in the physical space.

Fig. 3: Curved faces parametrization.

With this definition of curved faces, a curved tetrahedral subdomain with a face on a NURBS surface  $\mathcal{T}_{\text{Phys}}$  corresponds to a convex linear combination of the curved NURBS face and the remaining internal vertex mapped as

$$\begin{aligned} \Psi : \Lambda_{\text{Param}} \times [0, 1] &\longrightarrow \mathcal{T}_{\text{Phys}} \\ (\xi, \eta, \zeta) &\longmapsto \Psi(\xi, \eta, \zeta) := (1 - \zeta)\mathbf{S}(\xi, \eta) + \zeta\mathbf{v}_4, \end{aligned}$$

where  $\mathbf{v}_4$  is the internal vertex of  $\mathcal{T}_{\text{Phys}}$ .

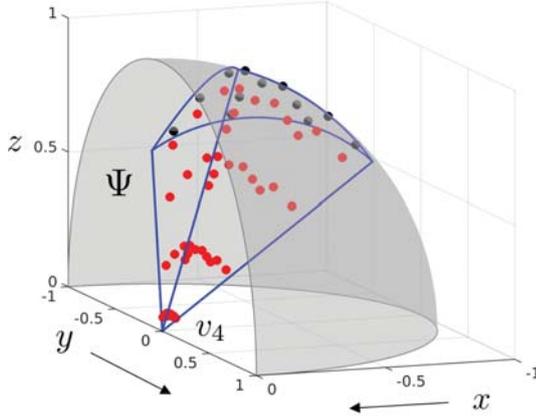


Fig. 4: Curved subdomain parametrization.

A convenient property of this strategy are the ability to decouple the directions of the surface definition,  $\Lambda_{\text{Param}}$  in the mapping  $\Psi$ , with respect to the interior direction  $\zeta$ .

Given these parametrizations, it is possible to perform the numerical integration over all the curved tetrahedral subdomains that form  $\Omega_{\text{B}}^{\text{phys}}$ . To this end, we consider tensor products of triangle quadratures for the curved faces and one-dimensional Gaussian quadratures for interior directions, see Figure 4. Analogous steps have to be made to integrate subdomains with only one edge on the boundary [39, 63].

## 2.1 Data sharing

It is worth noting that considering hierarchical relationships within the data structure suggests 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 5a shows a cross section of a model of a quarter of a cylinder, Figure 5b presents a coarse analysis mesh and Figure 5c 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.

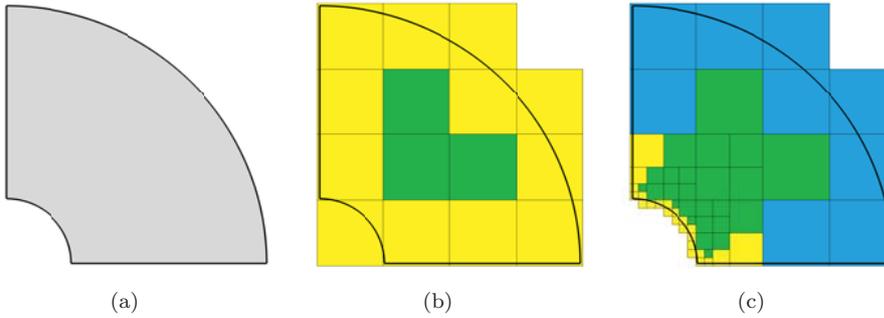


Fig. 5: *Vertical data sharing* example. (a) Model of a quarter of a cylinder. (b) Coarse mesh  $i$ . (c)  $h$ -adapted mesh  $i + 1$ .

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 5c 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 5c 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 6a).
2. Moving part with fixed intersection pattern. These surfaces or curves can be changed by the optimization algorithm, but those changes do not imply a modification of the intersection with the surrounding elements. In Figure 6a 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 6a 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 6b we can see a  $h$ -adapted mesh of a geometry,  $j + 1$ , that represents a perturbation of the original model,  $j$ , shown in Figure 5a. In Figures 6c and 6d we can verify how the blue elements used in the mesh in Figure 6b 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 6b is considerably lower than the total number of elements in the mesh.

## 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 for  $h$ -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 [58] and applied in an implementation of a FEM that used geometry-conforming meshes. Later, NDD was adapted to a Cartesian grid environment in 2D [45]. In this paper, we will

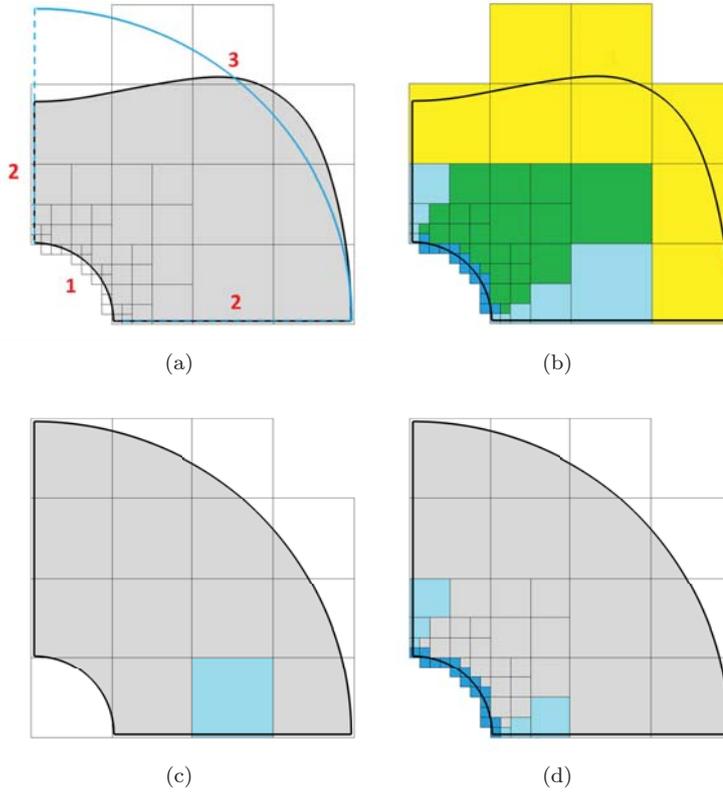


Fig. 6: *Horizontal data sharing* example. (a) Different type of entities. (b) Individual  $j + 1$ . (c) Individual  $j$ , mesh  $i$ . (d) Individual  $j$ , mesh  $i + 1$ .

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  $2 \times 2 \times 2$  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.

Figures 7, 8 and 9 graphically show the process. The embedding domain, Figure 7a, is subdivided into 8 subdomains or regions as shown in Figure 7b. 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 7c shows an example of an analysis mesh where we are going to apply the nested reordering.

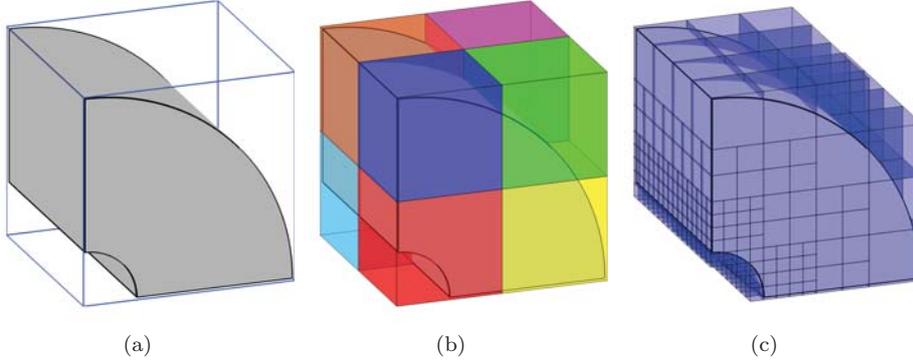


Fig. 7: Nested Domain Decomposition environment. (a) Body within the embedding domain  $\Omega_{\text{phys}} \subset \Omega$ . (b) Level 1 subdivision. (c) Example of 3D mesh to be reordered.

For the sake of clarity we will use a 2D representation of the process. Figure 8a shows the domain subdivision considering the Level-1 grid. The nodes are subdivided into 9 different categories. Only 5 of these categories are shown in the 2D representation of Figure 8. 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 9b. 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 a structure similar to the structure of the original non-reordered stiffness matrix shown in Figure 9a.

Level-2 reordering, Figure 8b, indicates that each of the Level-1 subdomains is again reordered in the same way. For instance, the red subdomain in Figure 8a is subdivided into 8 subdomains (only 4 are shown in 2D) separated by their interface, represented in black, as shown in Figure 8b. 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 Figure 8a to Figure 8c 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 9c. This nested structure could also be used to define efficient domain decomposition solvers or iterative solvers,

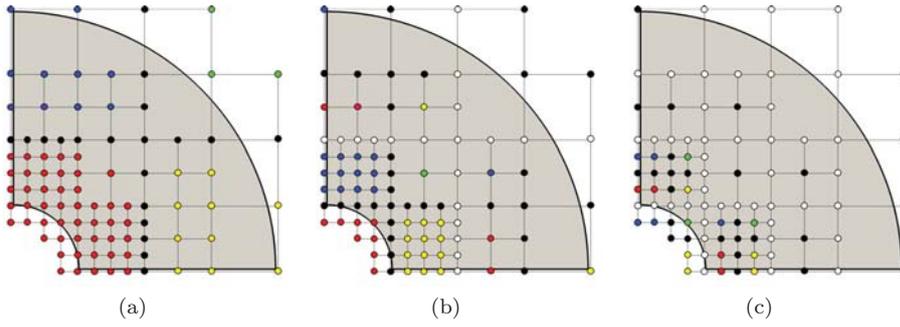


Fig. 8: Nested Domain Decomposition scheme. (a) Level-1 decomposition, (b) Level-2 decomposition and (c) Level-3 decomposition.

as in [59] 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.

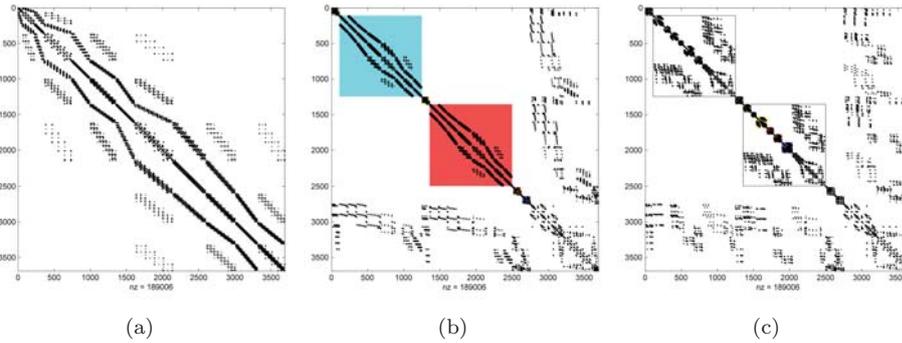


Fig. 9: Nested Domain Decomposition output. (a) Original stiffness matrix, (b) level 1 decomposition reordering and (c) last reordering (level 4).

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 analyses

Let us consider a bounded domain  $\Omega_{\text{phys}} \in \mathbb{R}^3$ . The contour can be separated into two non-overlapping parts,  $\Gamma_{\text{N}}$  and  $\Gamma_{\text{D}}$ , where Neumann and Dirichlet conditions are respectively imposed. The objective is to find the displacement field  $\mathbf{u} \in \mathcal{U}$

that satisfy the internal equilibrium equation in the domain and the Neumann and Dirichlet boundary conditions, which can be formulated as follows:

$$\begin{aligned} \nabla \boldsymbol{\sigma}(\mathbf{u}) + \mathbf{t}_v &= 0 & \text{in } \Omega_{\text{phys}} \\ \boldsymbol{\sigma}(\mathbf{u}) \mathbf{n} &= \mathbf{t}_s & \text{on } \Gamma_{\text{N}} \\ \mathbf{u} &= \mathbf{g} & \text{on } \Gamma_{\text{D}} \\ \boldsymbol{\epsilon}(\mathbf{u}) &= \mathbf{D} \boldsymbol{\sigma}(\mathbf{u}) \end{aligned} \quad (3)$$

In the above expression displacements  $\mathbf{u}$  belong to  $\mathcal{U} \equiv [H^1(\Omega)]^d$ ,  $\mathbf{t}_v \in [L^2(\Omega)]^d$  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. In linear elasticity, the strain tensor is defined from the displacement field by

$$\boldsymbol{\epsilon}(\mathbf{u}) = (\nabla \mathbf{u} + \nabla^T \mathbf{u})/2 \quad (4)$$

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

$$\boldsymbol{\epsilon} = (\boldsymbol{\sigma} - \nu (\text{tr}(\boldsymbol{\sigma}) \mathbf{I} - \boldsymbol{\sigma})) / E \quad (5)$$

The next property concerning the constitutive equation 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*

$$\|(\mathbf{u})\|^2 \leq C_E ((\mathbf{u}) : \boldsymbol{\epsilon}(\mathbf{u})) \quad \text{with} \quad C_E = \frac{E}{1 - 2\nu} \quad (6)$$

The weak form of the elastic problem allows different approaches to imposing the Dirichlet boundary conditions. The most usual procedure is to impose a constraint in the space of virtual displacement  $\mathcal{V}$ . The equilibrium between the virtual work of the elastic forces and the virtual work of the external forces applied is written as:

$$a(\mathbf{u}, \mathbf{v}) = c(\mathbf{v}) \quad \forall \mathbf{v} \in \mathcal{V} \quad (7)$$

where

$$\begin{aligned} a(\mathbf{u}, \mathbf{v}) &= \int_{\Omega_{\text{phys}}} \boldsymbol{\sigma}(\mathbf{u}) : \boldsymbol{\epsilon}(\mathbf{v}) \, d\Omega \\ c(\mathbf{v}) &= \int_{\Omega_{\text{phys}}} \mathbf{v} \cdot \mathbf{t}_v \, d\Omega + \int_{\Gamma_{\text{N}}} \mathbf{v} \cdot \mathbf{t}_s \, d\Gamma \end{aligned} \quad (8)$$

This method is very simple and effective in the context of the body-fitted FEM. However, this method is not valid for Cartesian meshes, since it is very difficult to get a null field on the Dirichlet boundary when the contour of the geometry does not match with the faces of the elements.

For this reason it seems more appropriate to use another formulation, for instance, to define the elastic problem as 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:

$$\min \left( \frac{1}{2} a(\mathbf{v}, \mathbf{v}) - c(\mathbf{v}) \right) \quad \text{with} \quad \mathbf{v} = \mathbf{g} \text{ in } \Gamma_D \quad (9)$$

One approach to solve this minimization problem is to use the Lagrange multipliers method. Besides the displacement field, a new field of Lagrange multipliers  $\boldsymbol{\lambda}$  associated with the reaction forces is added. The Lagrange multipliers belong to the Hilbert space  $\mathcal{M} = [H^{-\frac{1}{2}}(\Gamma_D)]^d$ . Formally, the problem is to find the saddle point  $[\mathbf{u}, \boldsymbol{\lambda}] \in \mathcal{U} \times \mathcal{M}$  of the following Lagrangian

$$\mathcal{L}(\mathbf{v}, \boldsymbol{\mu}) = \frac{1}{2} a(\mathbf{v}, \mathbf{v}) + b(\boldsymbol{\mu}, \mathbf{v} - \mathbf{g}) - c(\mathbf{v}) \quad \text{with} \quad b(\boldsymbol{\mu}, \mathbf{u}) = \int_{\Gamma_D} \boldsymbol{\mu} \cdot \mathbf{u} \, d\Gamma \quad (10)$$

The spaces of the finite element solution are denoted as  $\mathcal{U}^h \subset \mathcal{U}$  for displacements and  $\mathcal{M}^h \subset \mathcal{M}$  for multipliers. Substituting the FE fields in Equation (10) and optimizing the Lagrangian we obtain the following system of equations:

$$\begin{aligned} a(\mathbf{u}^h, \mathbf{v}^h) + b(\boldsymbol{\lambda}^h, \mathbf{v}^h) &= c(\mathbf{v}^h) \quad \forall \mathbf{v}^h \in \mathcal{U}^h \\ b(\boldsymbol{\mu}^h, \mathbf{u}^h) &= b(\boldsymbol{\mu}^h, \mathbf{g}) \quad \forall \boldsymbol{\mu}^h \in \mathcal{M}^h \end{aligned} \quad (11)$$

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.

In practice, the natural choices of the Lagrange multiplier field do not satisfy *InfSup* condition because they introduce too many constraints. To alleviate this situation, stabilized methods 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.

In this contribution we use an approach derived from the following Lagrangian:

$$\mathcal{L}_S(\mathbf{v}^h, \boldsymbol{\mu}^h) = \mathcal{L}(\mathbf{v}^h, \boldsymbol{\mu}^h) - \frac{1}{2} \frac{h}{k} \int_{\Gamma_D} \left\| \boldsymbol{\mu}^h + \mathbf{T}(\hat{\mathbf{u}}^h) \right\|^2 \, d\Gamma \quad (12)$$

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

In our formulation we use the recovered tractions on  $\Gamma_D$  evaluated from the recovered stress field  $\boldsymbol{\sigma}^*$  [60] to stabilize, solving the problem iteratively by updating the stress field value [68, 69],  $\boldsymbol{\sigma}^*(\hat{\mathbf{u}}^h)$  the FE recovered stress field being calculated for an FE solution from a previous iteration  $\hat{\mathbf{u}}^h$ . 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.

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

$$\begin{aligned} a(\mathbf{u}^h, \mathbf{v}^h) + \frac{k}{h} \int_{\Gamma_D} \mathbf{u}^h \cdot \mathbf{v}^h d\Gamma = \\ c(\mathbf{v}^h) + \frac{k}{h} \int_{\Gamma_D} \mathbf{g} \cdot \mathbf{v}^h + \int_{\Gamma_D} \mathbf{T}(\hat{\mathbf{u}}^h) \cdot \mathbf{v}^h d\Gamma \quad \forall \mathbf{v}^h \in \mathcal{U}^h \end{aligned} \quad (13)$$

The second term on the left hand side of (13) 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 FE solution to converge to the exact solution when the mesh is refined.

The problem (13) can be written in matrix form as:

$$\bigcup_{e=1}^{n_e} (\mathbf{k}^e + \mathbf{k}_D^e) \{\mathbf{u}^e\} = \bigcup_{e=1}^{n_e} (\mathbf{f}_q^e + \mathbf{f}_g^e + \mathbf{f}_s^e) \quad (14)$$

The stiffness matrix of each element is computed by

$$\mathbf{k}^e = \int_{\Omega_{\text{phys}}^e} \mathbf{B}^T \mathbf{D} \mathbf{B} |\mathbf{J}| d\Omega \quad (15)$$

where

- $\Omega_{\text{phys}}^e$  is the domain in local element coordinates,
- $\mathbf{B}$  is the nodal strains-displacements matrix,
- $\mathbf{D}$  is the elasticity tensor,
- $|\mathbf{J}|$  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  $(\xi, \eta, \tau)$ .

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:

$$\mathbf{f}_q^e = \int_{\Gamma_N^e} \mathbf{N}^T \mathbf{t} |\mathbf{J}| d\Gamma + \int_{\Omega^e} \mathbf{N}^t \mathbf{b} |\mathbf{J}| d\Omega \quad (16)$$

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

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:

$$\mathbf{k}_D^e = \int_{\Gamma_D^e} \frac{\kappa^*}{h} \mathbf{C}^T \mathbf{C} |\mathbf{J}| d\Gamma \quad (17)$$

where

$\Gamma_D^e$  is the portion of the Dirichlet boundary within the element,

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

$h$  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  $z$ .

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

with  $nmod$  as the number of nodes per element. Otherwise  $\mathbf{C} = \mathbf{S}\mathbf{N}$ , 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  $\delta$  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  $\Gamma_D$  and it is evaluated assembling the contribution of every element on the Dirichlet boundary:

$$\mathbf{f}_g^e = \int_{\Gamma_D^e} \frac{\kappa^*}{h} \mathbf{C}^T g |J| d\Gamma \quad (18)$$

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  $\Gamma_D$  evaluated from the recovered stress field  $\boldsymbol{\sigma}^*(\hat{\mathbf{u}}^h)$ . 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

$$\mathbf{f}_s^e = \int_{\Gamma_D^e} \mathbf{C}^T \mathbf{T}(\hat{\mathbf{u}}^h) |J| d\Gamma \quad (19)$$

Regarding the structural sensitivity analysis, the differentiation of the previous system of equations is needed, including the components due to the imposition of the boundary conditions. In this work we use a formulation presented in [40] that is based on the analytical discrete method [31, 43, 50, 53] 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 [30].

The derivative of (14) with respect to any design variable  $a_m$  allows to obtain the sensitivity of the calculation

$$\bigcup_{e=1}^{n_e} \left( \left( \frac{\partial \mathbf{k}^e}{\partial a_m} + \frac{\partial \mathbf{k}_D^e}{\partial a_m} \right) \{ \mathbf{u}^e \} + (\mathbf{k}^e + \mathbf{k}_D^e) \left\{ \frac{\partial \mathbf{u}^e}{\partial a_m} \right\} \right) = \bigcup_{e=1}^{n_e} \left( \frac{\partial \mathbf{f}_q^e}{\partial a_m} + \frac{\partial \mathbf{f}_g^e}{\partial a_m} + \frac{\partial \mathbf{f}_s^e}{\partial a_m} \right) \quad (20)$$

First, starting with  $\mathbf{K}$  and considering that the derivative of material properties matrix,  $\mathbf{D}$ , with respect to design variables is zero

$$\frac{\partial \mathbf{k}^e}{\partial a_m} = \int_{\Omega_{\text{Phys}}^e} \left[ \frac{\partial \mathbf{B}^T}{\partial a_m} \mathbf{D} \mathbf{B} + \mathbf{B}^T \mathbf{D} \frac{\partial \mathbf{B}}{\partial a_m} \right] |J| d\Omega + \int_{\Omega_{\text{Phys}}^e} \left[ \mathbf{B}^T \mathbf{D} \mathbf{B} \frac{\partial |J|}{\partial a_m} \right] d\Omega \quad (21)$$

where

$\frac{\partial |\mathbf{J}|}{\partial a_m}, \frac{\partial \mathbf{B}}{\partial a_m}$  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 [40]).

The derivatives of the terms introduced by the stabilization method will be

$$\begin{aligned} \frac{\partial \mathbf{k}_D^e}{\partial a_m} &= \int_{\Gamma_D^e} \frac{\kappa^*}{h} \mathbf{C}^T \mathbf{C} \frac{\partial |\mathbf{J}|}{\partial a_m} d\Gamma \\ \frac{\partial \mathbf{f}_g^e}{\partial a_m} &= \int_{\Gamma_D^e} \frac{\kappa^*}{h} \mathbf{C}^T g \frac{\partial |\mathbf{J}|}{\partial a_m} d\Gamma \\ \frac{\partial \mathbf{f}_s^e}{\partial a_m} &= \int_{\Gamma_D^e} \left[ \mathbf{C}^T \frac{\partial \mathbf{T}(\hat{\mathbf{u}}^h)}{\partial a_m} |\mathbf{J}| + \mathbf{C}^T \mathbf{T}(\hat{\mathbf{u}}^h) \frac{\partial |\mathbf{J}|}{\partial a_m} \right] d\Gamma \end{aligned} \quad (22)$$

As mentioned above, 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 and its derivative can be written as

$$\frac{\partial \mathbf{T}(\hat{\mathbf{u}}^h)}{\partial a_m} = \frac{\partial \boldsymbol{\sigma}^*}{\partial a_m} \mathbf{n} + \boldsymbol{\sigma}^* \frac{\partial \mathbf{n}}{\partial a_m} \quad (23)$$

### 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:

$$V = \bigcup_{e=1}^{n_e} V_e = \bigcup_{e=1}^{n_e} \int_{\Omega_{\text{Phys}}^e} d\Omega = \bigcup_{e=1}^{n_e} \int_{\Omega_{\text{Phys}}^e} |\mathbf{J}| d\xi d\eta d\tau \quad (24)$$

Applying the techniques discussed above to differentiate the components of the system of equations we obtain

$$\frac{\partial V_e}{\partial a_m} = \int_{\Omega_{\text{Phys}}^e} \frac{|\mathbf{J}|}{\partial a_m} d\xi d\eta d\tau \quad (25)$$

where  $\frac{|\mathbf{J}|}{\partial a_m}$  can be evaluated as:

$$\frac{|\mathbf{J}|}{\partial a_m} = |\mathbf{J}| \cdot \text{trace} \left( \mathbf{J}^{-1} \frac{\mathbf{J}}{\partial a_m} \right) \quad \text{with} \quad \frac{\mathbf{J}}{\partial a_m} = \sum_i^{nnod} \begin{Bmatrix} N_{i,\xi} \\ N_{i,\eta} \\ N_{i,\tau} \end{Bmatrix} \frac{\partial}{\partial a_m} \{x_i, y_i, z_i\} \quad (26)$$

In the last expression, all the components are evaluated as part of the shape sensitivity analysis. The derivatives of the nodal coordinates with respect to the design variables are the so-called velocity field whose definition will be given in Section 4.

In our study, the constraints are expressed in terms of stresses. To evaluate the stresses we consider the general expression for the calculation of stresses in continuous isoparametric elements

$$\boldsymbol{\sigma} = \mathbf{DB}\mathbf{u}_h^e \quad (27)$$

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

$$\frac{\partial \boldsymbol{\sigma}}{\partial a_m} = \mathbf{DB} \frac{\partial \mathbf{u}_h^e}{\partial a_m} + \mathbf{D} \frac{\partial \mathbf{B}}{\partial a_m} \mathbf{u}_h^e \quad (28)$$

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 [60].

### 3.2 Error estimator

The error associated with the FE discretization is evaluated in this work using the Zienkiewicz and Zhu [73] error estimator as:

$$\|\mathbf{e}_{es}\|^2 = \int_{\Omega_{\text{phys}}} (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^*)^T \mathbf{D}^{-1} (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^*) d\Omega \quad (29)$$

where  $\boldsymbol{\sigma}^*$  is a smoothed continuous stress field obtained by the recovery technique [45, 60]. The resulting expression for the sensitivity analysis of the error estimator already presented in [7]:

$$\frac{\partial \|\mathbf{e}_{es}\|^2}{\partial a_m} = \int_{\Omega_{\text{phys}}} (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^*)^T \mathbf{D}^{-1} \left( 2 \left( \frac{\partial (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^*)}{\partial a_m} \right) + \frac{(\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^*)}{|\mathbf{J}|} \frac{\partial |\mathbf{J}|}{\partial a_m} \right) |\mathbf{J}| d\Omega \quad (30)$$

where  $\frac{\partial (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^*)}{\partial a_m}$  will be approximated as follows:

$$\frac{\partial (\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^*)}{\partial a_m} = \frac{\partial \boldsymbol{\sigma}_h}{\partial a_m} - \left( \frac{\partial \boldsymbol{\sigma}}{\partial a_m} \right)^* \quad (31)$$

being  $\left( \frac{\partial \boldsymbol{\sigma}}{\partial a_m} \right)^*$  obtained through the same recovery procedure applied previously to  $\boldsymbol{\sigma}^*$ .

Equation (30) was also derived in [20] 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:

$$\|\mathbf{u}_{es}\|^2 \approx \mathbf{u}^T \mathbf{K} \mathbf{u} + \|\mathbf{e}_{es}\|^2 \quad (32)$$

$$\frac{\partial \|\mathbf{u}_{es}\|^2}{\partial a_m} \approx \frac{\partial \mathbf{u}^T}{\partial a_m} \mathbf{K} \mathbf{u} + \mathbf{u}^T \frac{\partial \mathbf{K}}{\partial a_m} \mathbf{u} + \mathbf{u}^T \mathbf{K} \frac{\partial \mathbf{u}}{\partial a_m} + \frac{\partial \|\mathbf{e}_{es}\|^2}{\partial a_m} \quad (33)$$

#### 4 Automatic $h$ -adaptive mesh projection

In this contribution we use a gradient-based algorithm [49] which uses first-order sensitivities of the objective function 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 10a, the algorithm generates the modified values of  $\mathbf{a}^j$  defining an improved design ( $\mathbf{a}^{j+1}$ ) using

$$\mathbf{a}^{j+1} = \mathbf{a}^j + \alpha \mathbf{S}(\mathbf{a})^j \quad (34)$$

where  $\mathbf{S}(\mathbf{a})^j$  is the search direction vector and  $\alpha$  is the step-length parameter.

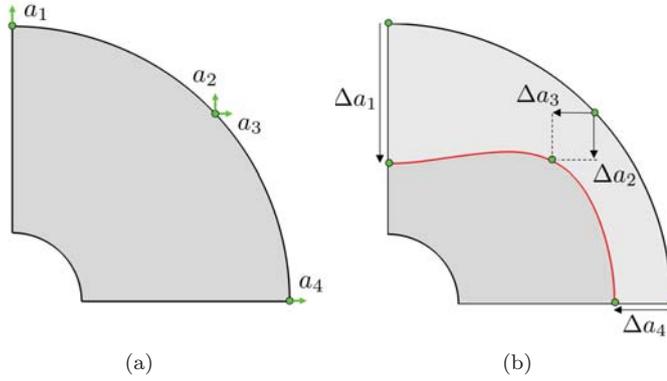


Fig. 10: Design evolution during optimization. (a) Reference design ( $j$ ). (b) Perturbed design ( $j + 1$ ).

After the definition of the  $(j + 1)$ -th geometry to be analyzed, see Figure 10b, it is necessary to construct the new analysis mesh. There have been previous developments about this using standard body-fitted FE meshes [7, 8]. In these references the information required to define a new mesh was projected from one geometry to another making use of the following expression:

$$M_{j+1} \approx M_j + \sum \left( \frac{\partial M_j}{\partial a_m} \right) \cdot \Delta a_m \quad (35)$$

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 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 [19]. In the following, we use a 3D generalization of this criterion presented in [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:

$$h_{e,n}^{n+1} \approx h_e^n \left[ \frac{1}{M^n} \right]^{1/2(p+1)} \left[ \frac{\|\mathbf{e}^{n+1}\|}{\|\mathbf{e}^n\|} \right]^{\frac{d}{2p^2+p d}} \left[ \frac{\|\mathbf{e}^{n+1}\|}{\|\mathbf{e}^n\|_e} \right]^{\frac{2}{2p+d}} \quad (36)$$

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 (36) by the projection given in Equation (39), evaluate  $\|\mathbf{e}\|_n$  as the summation of all the projected errors in elements from Equation (39), and evaluate  $\|\mathbf{e}^{n+1}\|$  as

$$\|\mathbf{e}^{n+1}\| = \frac{\gamma}{100} \|\mathbf{u}_{es}^{j+1}\| \quad (37)$$

where  $\gamma$  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 11a, using the previously computed coordinate sensitivities. The projected position  $\mathbf{r}^{j+1}$  for each node of the mesh is given by:

$$\mathbf{r}^{j+1} = \mathbf{r}^j + \sum_m \left( a_m^{j+1} - a_m^j \right) \left( \frac{\partial \mathbf{r}^j}{\partial a_m^j} \right) \quad (38)$$

*Remark 1* The velocity field at nodes,  $\frac{\partial \mathbf{r}}{\partial a_m}$ , can be defined using different strategies. As the location of the nodes of the Cartesian grid is always maintained we could simply consider that  $\frac{\partial \mathbf{r}}{\partial a_m} = 0$  in the internal nodes, thus  $\frac{\partial \mathbf{r}}{\partial a_m} \neq 0$  only on the varying portion of the boundary. However, this velocity field is not suitable for projection purposes since we need information also in the internal elements to be able to properly deform the mesh. To overcome this, we design a velocity field based on the physical approach [11] where we solve a FE problem in the entire domain imposing, as displacements, the velocity field calculated on the boundary. The resulting displacement field is then interpreted as the required velocity field. This kind of procedure is more expensive than the first one, but this computational cost can be alleviated by using a coarse FE mesh for projecting the information.

Likewise, the estimated error in energy norm and the estimated energy norm at each element required in (36) can also be estimated by projection using the expressions

$$\|\mathbf{e}_{es}\|_{e,j+1}^2 \approx \|\mathbf{e}_{es}\|_{e,j}^2 + \sum_m \left( a_m^{j+1} - a_m^j \right) \frac{\partial \|\mathbf{e}_{es}\|_e^2}{\partial a_m} \quad (39)$$

$$\|\mathbf{u}_{es}\|_{e,j+1}^2 \approx \|\mathbf{u}_{es}\|_{e,j}^2 + \sum_m \left( a_m^{j+1} - a_m^j \right) \frac{\partial \|\mathbf{u}_{es}\|_e^2}{\partial a_m} \quad (40)$$

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 11b.

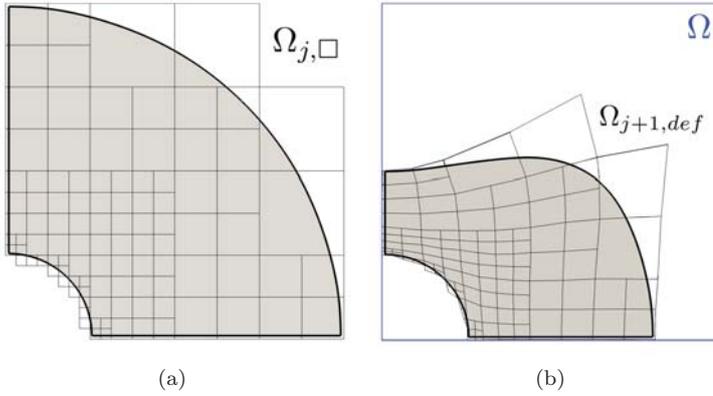


Fig. 11: Mesh projection procedure. (a) Cartesian reference analysis mesh,  $\Omega_{j,\square}$ . (b) Projected (non-Cartesian) mesh on geometry  $j+1$ .

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 11b) equivalent to the one used in the previous design  $j$  (Figure 11a). 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 (36).

Up to this point, the mesh projection presented is comparable to the strategies used for body-fitted meshes [7, 8]. As we can easily observe in Figure 11b, 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.

This strategy simply requires to project the element size, evaluated using (36) for the elements of  $\Omega_{j+1,def}$  (Figure 11b), to the embedding domain  $\Omega$ . 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  $\Omega$ . 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 Gauss points contained in the element, leading to an  $h$ -adapted Cartesian grid (see 12b).

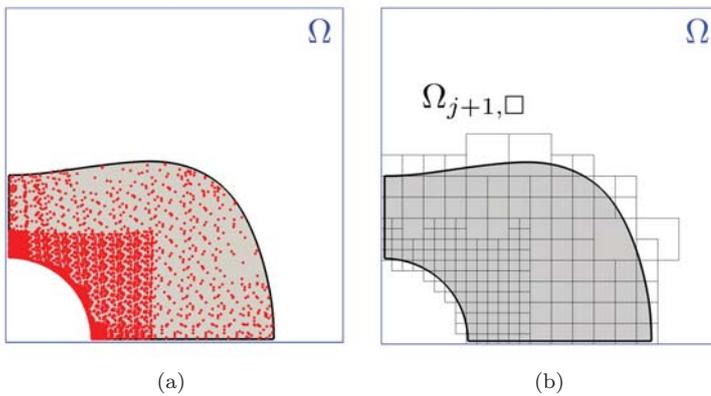


Fig. 12: Mesh projection procedure. (a) Perturbed integration points. (b) Projected Cartesian mesh,  $\Omega_{j+1,\square}$ .

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 Algorithm for constrained optimization

For the numerical examples in this contribution, we consider general problems of shape optimization as outlined in (1). We will use a sequential quadratic programming (SQP) approach [54], considered an active-set method [22], given by the MATLAB implementation [1]. SQP approaches are one of the most effective methods for non-linearly constrained optimization and generates steps by solving quadratic subproblems [49].

The formulation of a quadratic programming subproblem for the problem description in (1) is based on a quadratic approximation of the Lagrangian function

$$\mathcal{L}(\mathbf{a}, \lambda) = f(\mathbf{a}) + \sum_{n=1}^r \lambda_n \cdot g_n(\mathbf{a}) \quad (41)$$

To model this problem we now linearize the inequality constraints of (1) to obtain

$$\begin{aligned} & \text{minimize} && f(\mathbf{a}_t) + \nabla f(\mathbf{a}_t)^T \mathbf{p} \nabla_{aa}^2 \mathcal{L}_t \mathbf{p} \\ & \text{verifying} && \nabla g_n(\mathbf{a}_t) \mathbf{p} + g_n(\mathbf{a}_t) \leq 0 \quad n = 1, \dots, r \end{aligned} \quad (42)$$

The new iterate is given by  $(\mathbf{a}_t + \mathbf{p}_t, \lambda_{t+1})$  where  $\mathbf{p}_t$  and  $\lambda_{t+1}$  are the solution and the corresponding Lagrange multiplier.

In this approach the set of active constraints  $\mathcal{A}_t = \{g_n(\mathbf{a}_t) = 0\}$  at the solution of (42) constitutes our guess of the active set at the solution of the nonlinear program. If the SQP method is able to correctly identify this optimal active set (and not change its guess at a subsequent iteration) then it will act like a Newton method for equality-constrained optimization and will converge rapidly [49]. For details on the MATLAB implementation we recommend [1].

## 6 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 reorderings to the matrices. The remaining two problems will be devoted to assess the optimization methodology presented in this contribution. The last two optimization analyses will test the accuracy of the 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 13. 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:

$$u_r = \frac{P(1+\nu)}{E(k^2-1)} \left( r(1-2\nu) + \frac{R_{ext}^2}{r} \right), \quad u_y = 0 \quad (43)$$

$$\sigma_r = \frac{P}{k^2-1} \left( 1 - \frac{R_{ext}^2}{r^2} \right), \quad \sigma_\phi = \frac{P}{k^2-1} \left( 1 + \frac{R_{ext}^2}{r^2} \right), \quad \sigma_y = \nu(\sigma_r + \sigma_\phi) \quad (44)$$

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.

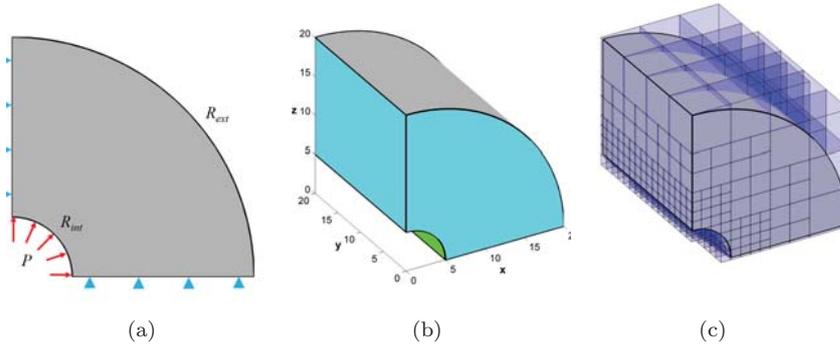


Fig. 13: Model of a cylinder under internal pressure. (a) Front view with boundary conditions. (b) 3D model representation. (c) Example of analysis mesh.

### 6.1 Performance of the direct solver

As explained in Section 2.2, the solution of the system of equations with direct solvers is a time consuming task that can be lightened using a proper reordering of the matrices involved. To solve the linear system of equations in (13), we have run the tests in MATLAB<sup>®</sup> 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 [3] generates an ordering based on how these degrees are altered during Cholesky factorization.
- *Symmetric AMD permutation (SYM-AMD) [1]*: this algorithm performs an AMD reordering taking into account the symmetry of the matrix.
- *Column AMD permutation (COL-AMD) [12]*: this algorithm returns the column approximate minimum degree permutation vector of the matrix. This is the default algorithm used by MATLAB<sup>®</sup>.

For the analysis we will study a set of uniformly refined meshes of 20-node tri-quadratic elements. The meshes used in this simulation can be seen in Figure 14.

On the left plot of Figure 15, we can observe the computational cost related with the reordering 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 solution 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

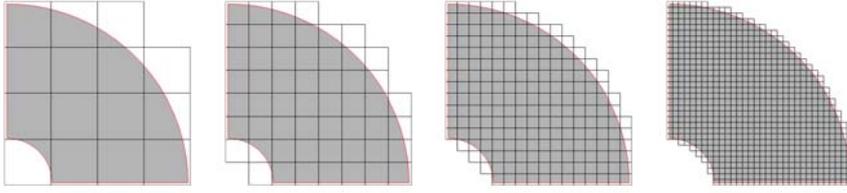


Fig. 14: 2D view of 3D uniform meshes with different element size.

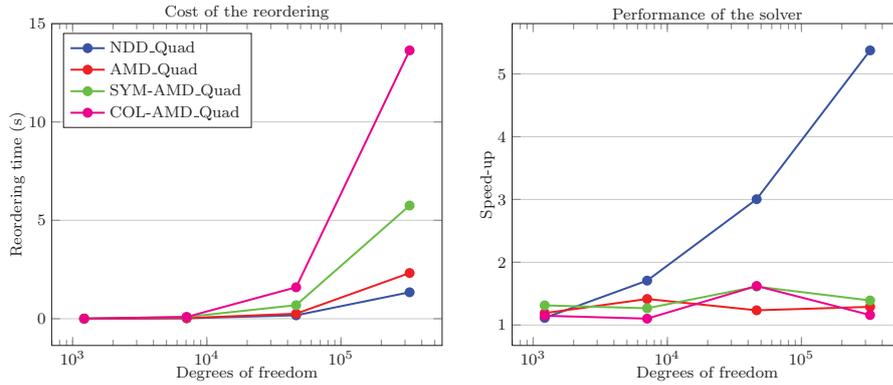


Fig. 15: Behavior of different reordering techniques. Left: reordering times. Right: speed-up in the solution of the system of equations with respect to the reference (no reordering).

represents the reduction of cost with respect with the reference calculation with no reordering.

From Figure 15 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 solution 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.

## 6.2 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 13. 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 14. By doing this, we will evaluate how varying the discretization affects the accuracy and the computational cost.

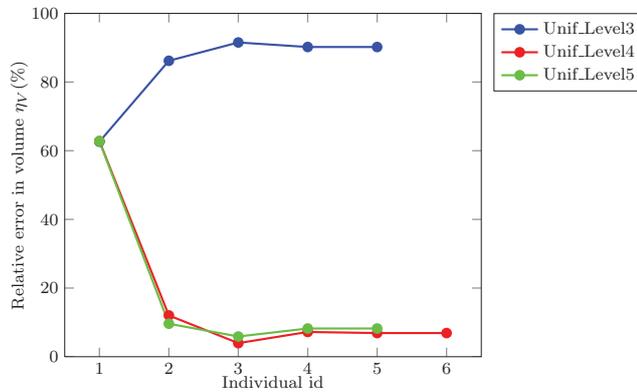


Fig. 16: Evolution of the error in the objective function (volume) with respect to the analytical solution. Uniform meshes.

In Figure 16 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 FE 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 17 we can observe the behavior of  $h$ -

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.

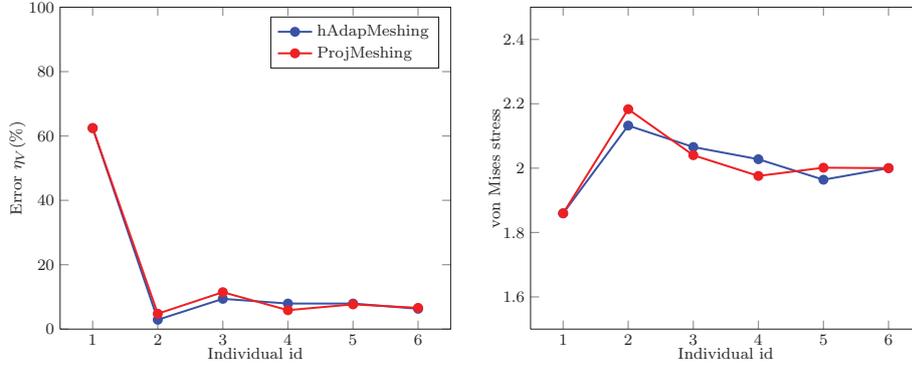


Fig. 17: Evolution of magnitudes for  $h$ -adapted and projected meshes. Left: error in the objective function (volume) with respect to the analytical solution. Right: von Mises stress.

adapted meshes with tri-quadratic elements (hAdapMeshing) and projected  $h$ -adapted meshes with the same elements (ProjMeshing).

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 variable. Computational results for  $h$ -adapted and projected meshes.

### 6.3 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 18. 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.

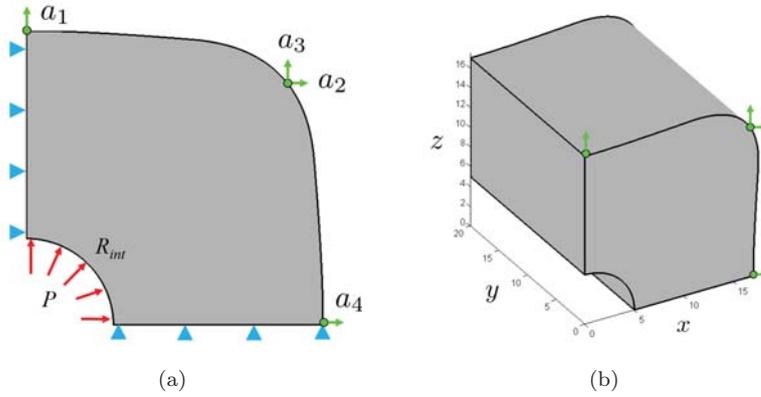


Fig. 18: Model of a cylinder under internal pressure defined by 4 design variables. (a) Front view with boundary conditions. (b) 3D model representation.

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 19 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.

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.

Figure 20 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.

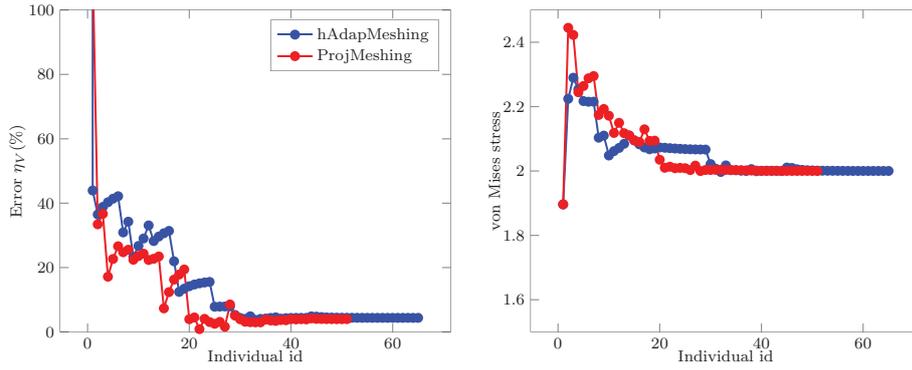


Fig. 19: Evolution of magnitudes for  $h$ -adapted and projected meshes. Left: error in the objective function (volume) with respect to the analytical solution. Right: von Mises stress.

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.

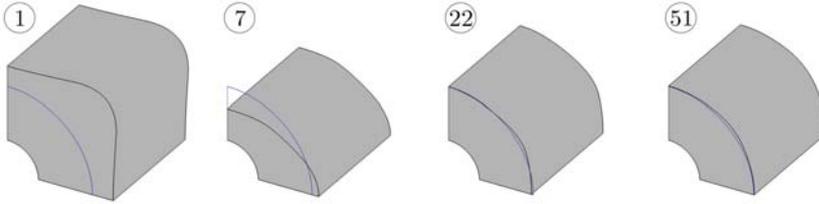


Fig. 20: Samples of individuals from the optimization procedure. The index indicates the number of model during the process.

#### 6.4 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 21. The geometry parameters are  $AB = 11$ ,  $C = 4$ ,  $AD = 20$ ,  $DE = 4$ ,  $F = 1.5$ ,  $DG = 7$ ,  $HG = 5.5$ . The Youngs modulus is  $E = 10^5$ , and Poissons ratio  $\nu = 0.333$ . The pressure is  $P = 100$  in the normal direction of the half arc as shown in Figure 21.

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 22). The allowable von Mises stress is  $\sigma_{VM} = 900$ .

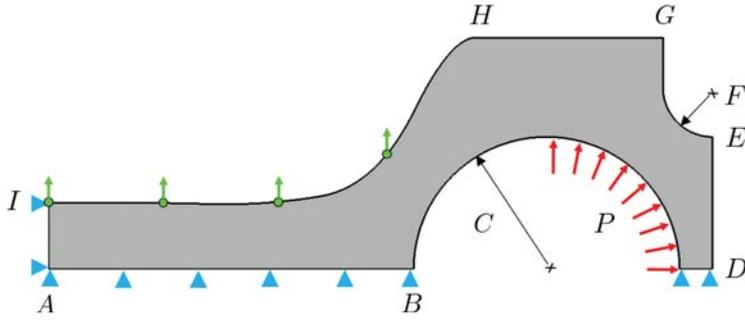


Fig. 21: Front view of the connecting rod problem with boundary conditions.

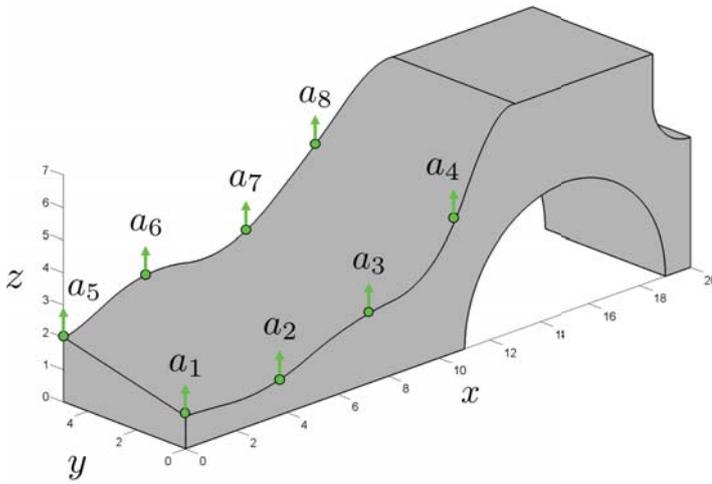


Fig. 22: 3D model representation showing the 8 design variables.

The initial values of the design variables and their allowed data range are shown in Table 6.

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.

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.

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 23 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.

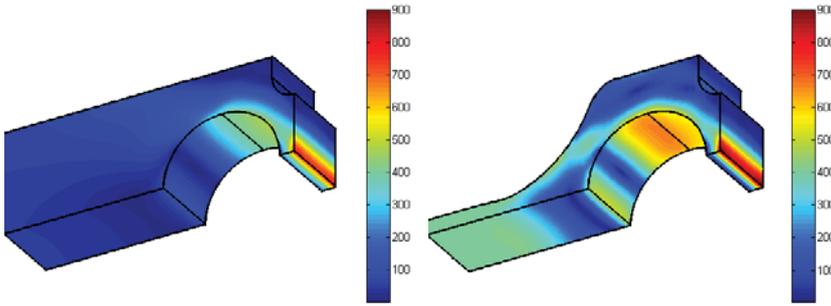


Fig. 23: Von Mises stress fields: (left) initial configuration results and (right) configuration obtained using projected meshes.

## 7 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 and tested for a 3D code. 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 demonstrated as effective, in terms of convergence, than the standard  $h$ -refined meshes, but with an important reduction of the computational cost per individual.

**Acknowledgements** 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 PROM-ETEO/2016/007.

## References

1. (2014) MATLAB version 8.3.0.532 (R2014a), Documentation. The Mathworks, Inc., Natick, Massachusetts
2. Abel JF, Shephard MS (1979) An algorithm for multipoint constraints in finite element analysis. *International Journal for Numerical Methods in Engineering* 14(3):464–467
3. Amestoy P, Davis T, Duff I (1996) An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications* 17(4):886–905
4. Barth W, Stürzlinger W (1993) Efficient ray tracing for Bezier and B-spline surfaces. *Computers & Graphics* 17(4):423–430
5. Bennett JA, Botkin ME (1985) Structural Shape Optimization with Geometric Problem Description and Adaptive Mesh Refinement. *AIAA Journal* 23(3):459–464
6. Braibant V, Fleury C (1984) Shape optimal design using b-splines. *Computer Methods in Applied Mechanics and Engineering* 44(3):247–267
7. Bugada G, Oliver J (1993) A General Methodology for Structural Shape Optimization Problems Using Automatic Adaptive Remeshing. *International Journal for Numerical Methods in Engineering* 36(18):3161–3185
8. Bugada G, Ródenas JJ, Oñate E (2008) An integration of a low cost adaptive remeshing strategy in the solution of structural shape optimization problems using evolutionary methods. *Computers & Structures* 86(13–14):1563–1578
9. Chang K, Choi KK (1992) A geometry-based parameterization method for shape design of elastic solids. *Mechanics of Structures and Machines* 20(2):215–252
10. Cho S, Ha SH (2009) Isogeometric shape design optimization: exact geometry and enhanced sensitivity. *Structural and Multidisciplinary Optimization* 38(1):53–70
11. D Belegundu YM, S Zhang, Salagame R (1991) The Natural Approach for Shape Optimization with Mesh Distortion Control. Tech. rep., Penn State University
12. Davis TA, Gilbert JR, Larimore S, Ng E (2004) An approximate column minimum degree ordering algorithm. *ACM Transactions on Mathematical Software* 30(3):353–376
13. Doctor LJ, Torborg JG (1981) Display techniques for octree-encoded objects. *IEEE Comput Graph Appl* 1(3):29–38
14. Dunning PD, Kim HA, Mullineux G (2011) Investigation and improvement of sensitivity computation using the area-fraction weighted fixed grid FEM and structural optimization. *Finite Elements in Analysis and Design* 47(8):933–941
15. Düster A, Parvizian J, Yang Z, Rank E (2008) The finite cell method for three-dimensional problems of solid mechanics. *Computer Methods in Applied Mechanics and Engineering* 197(45–48):3768–3782

16. Escobar JM, Montenegro R, Rodríguez E, Cascón JM (2014) The mecano method for isogeometric solid modeling and applications. *Engineering with Computers* 30(3):331–343
17. Farhat C, Lacour C, Rixen D (1998) Incorporation of linear multipoint constraints in substructure based iterative solvers. Part 1: a numerically scalable algorithm. *International Journal for Numerical Methods in Engineering* 43(6):997–1016
18. Fries TP, Omerović S (2016) Higher-order accurate integration of implicit geometries. *International Journal for Numerical Methods in Engineering* 106(5):323–371
19. Fuenmayor FJ, Oliver JL (1996) Criteria to achieve nearly optimal meshes in the h-adaptive finite element method. *International Journal for Numerical Methods in Engineering* 39(23):4039–4061
20. Fuenmayor FJ, Oliver JL, Ródenas JJ (1997) Extension of the Zienkiewicz-Zhu error estimator to shape sensitivity analysis. *International Journal for Numerical Methods in Engineering* 40(8):1413–1433
21. García-Ruiz MJ, Steven GP (1999) Fixed grid finite elements in elasticity problems. *Engineering Computations* 16(2):145–164
22. Gill P, Murray W, Saunders M, Wright M (1984) Procedures for optimization problems with a mixture of bounds and general linear constraints. *ACM Trans Math Software* 10:282–298
23. González-Estrada OA, Nadal E, Ródenas JJ, Kerfriden P, Bordas SPA, Fuenmayor FJ (2014) Mesh adaptivity driven by goal-oriented locally equilibrated superconvergent patch recovery. *Computational Mechanics* 53(5):957–976
24. Ha SH, Choi KK, Cho S (2010) Numerical method for shape optimization using T-spline based isogeometric method. *Structural and Multidisciplinary Optimization* 42(3):417–428
25. Haftka RT, Grandhi RV (1986) Structural shape optimization: A survey. *Computer Methods in Applied Mechanics and Engineering* 57(1):91–106
26. Haslinger J, Jedelsky D (1996) Genetic algorithms and fictitious domain based approaches in shape optimization. *Struc Optim* 12:257–264
27. Hughes TJR, Cottrell JA, Bazilevs Y (2005) Isogeometric Analysis: CAD, Finite Elements, NURBS, Exact Geometry, and Mesh Refinement. *Computer Methods in Applied Mechanics and Engineering* 194:4135–4195
28. Jackins CL, Tanimoto SL (1980) Oct-tree and their use in representing three-dimensional objects. *Computer Graphics and Image Processing* 14(3):249–270
29. Kajiya JT (1982) Ray Tracing Parametric Patches. *SIGGRAPH Comput Graph* 16(3):245–254
30. van Keulen F, Haftka RT, Kim N (2005) Review of options for structural design sensitivity analysis. Part I: linear systems. *Computer Methods in Applied Mechanics and Engineering* 194(30-33):3213–3243
31. Kibsgaard S (1992) Sensitivity analysis-the basis for optimization. *International Journal for Numerical Methods in Engineering* 34(3):901–932
32. Kikuchi N, Chung KY, Torigaki T, Taylor JE (1986) Adaptive finite element methods for shape optimization of linearly elastic structures. *Computer Methods in Applied Mechanics and Engineering* 57(1):67–89
33. Kim NH, Chang Y (2005) Eulerian shape design sensitivity analysis and optimization with a fixed grid. *Computer Methods in Applied Mechanics and Engineering* 194(30-33):3291–3314

34. Kudela L, Zander N, Kollmannsberger S, Rank E (2016) Smart octrees: Accurately integrating discontinuous functions in 3d. *Computer Methods in Applied Mechanics and Engineering* 306(1):406–426
35. Kunisch K, Peichl G (1996) Numerical gradients for shape optimization based on embedding domain techniques. *Comput Optim* 18:95–114
36. Li K, Qian X (2011) Isogeometric analysis and shape optimization via boundary integral. *Computer-Aided Design* 43(11):1427–1437
37. Lian H, Kerfriden P, Bordas SPA (2016) Implementation of regularized isogeometric boundary element methods for gradient-based shape optimization in two-dimensional linear elasticity. *International Journal for Numerical Methods in Engineering* 106(12):972–1017
38. Liu L, Zhang Y, Hughes TJR, Scott MA, Sederberg TW (2014) Volumetric T-spline Construction using Boolean Operations. *Engineering with Computers* 30(4):425–439
39. Marco O, Sevilla R, Zhang Y, Ródenas JJ, Tur M (2015) Exact 3D boundary representation in finite element analysis based on Cartesian grids independent of the geometry. *International Journal for Numerical Methods in Engineering* 103:445–468
40. Marco O, Ródenas JJ, Fuenmayor FJ, Tur M (2017) An extension of shape sensitivity analysis to an immersed boundary method based on cartesian grids. *Computational Mechanics* Submitted
41. Marco O, Ródenas JJ, Navarro-Jiménez JM, Tur M (2017) Robust h-adaptive meshing strategy for arbitrary cad geometries in a cartesian grid framework. *Computers & Structures* Submitted
42. Meagher D (1980) Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer. Tech. Rep. IPL-TR-80-11 I, Rensselaer Polytechnic Institute
43. Moita JS, Infante J, Mota CM, Mota CA (2000) Sensitivity analysis and optimal design of geometrically non-linear laminated plates and shells. *Computers & Structures* 76(1–3):407–420
44. Nadal E (2014) 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
45. Nadal E, Ródenas JJ, Albelda J, Tur M, Tarancón JE, Fuenmayor FJ (2013) Efficient Finite Element Methodology based on Cartesian Grids: Application to Structural Shape Optimization. *Abstract and Applied Analysis* 2013
46. Najafi AR, Safdari M, Tortorelli DA, Geubelle PH (2015) A gradient-based shape optimization scheme using an interface-enriched generalized FEM. *Computer Methods in Applied Mechanics and Engineering* 296:1–17
47. Nguyen VP, Anitescu C, Bordas SPA, Rabczuk T (2015) Isogeometric analysis: An overview and computer implementation aspects. *Mathematics and Computers in Simulation* 117:89–116
48. Nishita T, Sederberg TW, Kakimoto M (1990) Ray Tracing Trimmed Rational Surface Patches. *SIGGRAPH Comput Graph* 24(4):337–345
49. Nocedal J, Wright SJ (2006) *Numerical Optimization*, Second Edition. Springer-Verlag New York
50. Pandey PC, Bakshi P (1999) Analytical response sensitivity computation using hybrid finite elements. *Computers & Structures* 71(5):525–534

51. Parvizian J, Düster A, Rank E (2007) Finite Cell Method: h- and p- Extension for Embedded Domain Methods in Solid Mechanics. *Computational Mechanics* 41(1):121–133
52. Peskin CS (1977) Numerical Analysis of Blood Flow in the Heart. *Journal of Computational Physics* 25:220–252
53. Poldneff MJ, Rai IS, Arora JS (1993) Implementation of design sensitivity analysis for nonlinear structures. *AIAA Journal* 31(11):2137–2142
54. Powell M (1983) Variable metric methods for constrained optimization. In: Bachem A, Grottschel M, Korte B (eds) *Mathematical Programming: The State of the Art*, Springer, Berlin, Heidelberg, pp 288–311
55. Qian X (2010) Full analytical sensitivities in NURBS based isogeometric shape optimization. *Computer Methods in Applied Mechanics and Engineering* 199(29–32):2059–2071
56. Riehl S, Steinmann P (2014) An integrated approach to shape optimization and mesh adaptivity based on material residual forces. *Computer Methods in Applied Mechanics and Engineering* 278:640–663
57. Riehl S, Steinmann P (2016) On structural shape optimization using an embedding domain discretization technique. *International Journal for Numerical Methods in Engineering*
58. Ródenas JJ, Tarancón JE, Albelda J, Roda A, Fuenmayor FJ (2005) Hierarchical Properties in Elements Obtained by Subdivision: a Hierarchical h-adaptivity Program. In: Díez P, Wiberg NE (eds) *Adaptive Modeling and Simulation 2005*
59. Ródenas JJ, Corral C, Albelda J, Mas J, Adam C (2007) Nested domain decomposition direct and iterative solvers based on a hierarchical h-adaptive finite element code. In: Runesson K, Díez P (eds) *Adaptive Modeling and Simulation 2007*, International Center for Numerical Methods in Engineering (CIMNE), pp 206–209
60. Ródenas JJ, Tur M, Fuenmayor FJ, Vercher A (2007) Improvement of the superconvergent patch recovery technique by the use of constraint equations: the SPR-C technique. *International Journal for Numerical Methods in Engineering* 70(6):705–727
61. Ródenas JJ, Bugada G, Albelda J, Oñate E (2011) On the need for the use of error-controlled finite element analyses in structural shape optimization processes. *International Journal for Numerical Methods in Engineering* 87(11):1105–1126
62. Schillinger D, Ruess M (2015) The finite cell method: A review in the context of higher-order structural analysis of cad and image-based geometric models. *Archives of Computational Methods in Engineering* 22(3):391–455
63. Sevilla R, Fernández-Méndez S, Huerta A (2011) 3D-NURBS-enhanced Finite Element Method (NEFEM). *International Journal for Numerical Methods in Engineering* 88(2):103–125
64. Sevilla R, Fernández-Méndez S, Huerta A (2011) Comparison of High-order Curved Finite Elements. *International Journal for Numerical Methods in Engineering* 87(8):719–734
65. Sevilla R, Fernández-Méndez S, Huerta A (2011) NURBS-enhanced Finite Element Method (NEFEM): A Seamless Bridge Between CAD and FEM. *Archives of Computational Methods in Engineering* 18(4):441–484

66. Sweeney M, Bartels R (1986) Ray tracing free-form b-spline surfaces. *IEEE Computer Graphics and Applications* 6(2):41–49
67. Toth DL (1985) On Ray Tracing Parametric Surfaces. *SIGGRAPH Comput Graph* 19(3):171–179
68. Tur M, Albelda J, Nadal E, Ródenas JJ (2014) Imposing dirichlet boundary conditions in hierarchical cartesian meshes by means of stabilized lagrange multipliers. *International Journal for Numerical Methods in Engineering* 98(6):399–417
69. Tur M, Albelda J, Marco O, Ródenas JJ (2015) Stabilized Method to Impose Dirichlet Boundary Conditions using a Smooth Stress Field. *Computer Methods in Applied Mechanics and Engineering* 296:352–375
70. Yao T, Choi KK (1989) 3-d shape optimal design and automatic finite element regriding. *International Journal for Numerical Methods in Engineering* 28(2):369–384
71. Zhang L, Gerstenberger A, Wang X, Liu WK (2004) Immersed Finite Element Method. *Computer Methods in Applied Mechanics and Engineering* 293(21):2051–2067
72. Zhang Y, Wang W, Hughes TJR (2013) Conformal Solid T-spline Construction from Boundary T-spline Representations. *Computational Mechanics* 6(51):1051–1059
73. Zienkiewicz OC, Zhu JZ (1987) A Simple Error Estimator and Adaptive Procedure for Practical Engineering Analysis. *International Journal for Numerical Methods in Engineering* 24(2):337–357