A hierarchical h adaptivity methodology based on element subdivision

Metodología jerárquica h adaptativa basada en subdivisión de elementos

Juan Ródenas[1], José Albelda[2], Manuel Tur[3], Francisco Fuenmayor[4]

[1] Centro de Investigación en Ingeniería Mecánica, Universitat Politècnica de València, Camino de Vera s/n, 46022-Valencia, Spain. Email: jjrodena@mcm.upv.es
[2]Centro de Investigación en Ingeniería Mecánica, Universitat Politècnica de València, Camino de Vera s/n, 46022-Valencia, Spain. Email: jalbelda@mcm.upv.es
[3]Centro de Investigación en Ingeniería Mecánica, Universitat Politècnica de València, Camino de Vera s/n, 46022-Valencia, Spain. Email: matuva@mcm.upv.es
[4]Centro de Investigación en Ingeniería Mecánica, Universitat Politècnica de València, Camino de Vera s/n, 46022-Valencia, Spain. Email: ffuenmay@mcm.upv.es

RECEIVED: December 20, 2016. ACCEPTED: February 27, 2017. FINAL VERSION: May 15, 2017.

ABSTRACT

This paper presents a hierarchical h adaptive methodology for Finite Element Analysis based on the hierarchical relations between parent and child elements that come out if these elements are geometrically similar. Under this similarity condition the terms involved in the evaluation of element stiffness matrices of parent and child elements are related by a constant which is a function of the element sizes ratio (scaling factor). These relations have been the basis for the development of a hierarchical *h* adaptivity methodology based on element subdivision and the use of multi-point-constraints to ensure C^0 continuity. The use of a hierarchical data structure significantly reduces the amount of calculations required for the mesh refinement, the evaluation of the global stiffness matrix, element stresses and element error estimation. The data structure also produces a natural reordering of the global stiffness matrix that improves the behaviour of the Cholesky factorization.

KEYWORDS: Adaptive Modelling, Hierarchical properties, Mesh Enrichment, Mesh Generation.

RESUMEN

En este artículo se presenta una metodología h adaptativa para el Análisis por Elementos Finitos basada en las relaciones jerárquicas entre elementos padre e hijo que surgen si estos elementos son geométricamente similares. Bajo esta condición de similitud, los términos resultantes de la evaluación de las matrices de rigidez de elementos padre e hijo están relacionados por una constante que es una función de la relación de tamaños de elemento (factor de escala). Estas relaciones han sido la base para el desarrollo de una metodología jerárquica *h* adaptativa basada en la subdivisión de elementos y el uso de restricciones multipunto para asegurar la continuidad C^0. El uso de una estructura de datos jerárquica reduce significativamente la cantidad de cálculos requeridos para el refinamiento de la malla, la evaluación de la matriz de rigidez global, las tensiones de los elementos y la estimación del error del elemento. La estructura de datos también produce un reordenamiento natural de la matriz de rigidez global que mejora el comportamiento de la factorización de Cholesky.

PALABRAS CLAVE: Modelado Adaptativo, Propiedades jerárquicas, Enriquecimiento de malla, Mallado.

1. INTRODUCTION

The context of the developments presented in this paper is that of finding a methodology based on the use of the h-version of the Finite Element Method (FEM) for the accurate analysis of mechanical components with the lowest possible computational cost.

The generation of optimal meshes for FE analysis has been a particular area of development for many years [1-3]. It is well known that the cheapest FE mesh to produce a solution with a fixed quality at minimum cost is an adapted one. As a consequence of this, *h*-adaptive techniques based on the estimation of the FE discretization error are commonly used to speed up FE analysis. The *h*-adaptive analysis techniques for the control of the discretization error of FE analysis can be classified into two groups: the *mesh regeneration* techniques [4, 5], based on the *full remeshing* of the domain, and those based on *element splitting*, where the mesh is *enriched* (or *refined*) at a local level. Both of them are appropriate techniques to solve the problem although, in general terms, the first one provides a slightly more accurate solution than the second for a given number of degrees of freedom. However, the computational cost associated to the first one can be higher because, in most cases, the reduction of the error level involves the global regeneration of the mesh instead of a local modification. The evaluation of element matrices is not usually the bottleneck of the FE analysis, but it must be taken into account that the full remeshing involves the evaluation of the matrices for each element of the new mesh, whereas the use of element splitting techniques can (totally or partially) avoid these computations. A first comparative study between *mesh enrichment* and *mesh regeneration* techniques can be found in Zhu *et al* [6].

Element splitting techniques can be applied both, on structured meshes, based on a certain hierarchical structure [7], where the topological and geometrical relations between the elements are known *a*-priori; and on non-structured meshes [8-10] with elements of general shape by using one of the subdivision schemes described in the literature (bisection techniques, 4-T algorithm of Rivara, etc). The use of a structured element splitting, as in the quadtree and octree methods [8, 11] is rather efficient as it allows improving the storage of the mesh related information. However, other splitting techniques, that could be termed non-structured element splitting techniques, are more flexible as they can be applied on any mesh.

One of the most important factors to be taken into account when the element splitting techniques are used is the geometrical quality of the elements created during the refinement process. The structured element splitting allows for a good control of the newly created elements. Depending on the non-structured element splitting strategy used to refine the mesh, one can obtain more distorted elements or elements of an increasing geometrical quality [12, 13].

Another important factor to be considered is that certain element splitting techniques provide conforming refined meshes, whereas other strategies provide non-conforming meshes. In the first case [14], the refinement strategy allows for lower control of the element shape. The techniques that produce non-conforming meshes require additional techniques to restore C^0 continuity between contiguous elements, such as the addition of new elements [9, 15] or the use of Multi Point Constraints (MPCs) [16-19].
A hierarchical data structure, with *parent-child* relations between existing elements and those obtained during the mesh refinement process, can be easily defined if the mesh refinement process is based on element splitting. This kind of data structure has been the basis to define a new *h*-adaptive methodology for efficient FE analyses. A 2D *h*-adaptive FE code with hierarchical features for the resolution of the linear elasticity problem based on this structure has been created. This code has been used as a framework to explore the advantages of using hierarchical relations in a FE code with *h*-adaptive analyses capabilities. The data structure has also been used to create a natural reordering of the FE system of equations that speeds up its resolution with respect to the use of other reordering schemes. Therefore, the hierarchical *h*-adaptivity code allows for the use of an analysis methodology that improves the efficiency of the main parts of the process: mesh refinement, generation of the system of equations and its resolution, and any post-processing that involves volume integrals.

The remaining of the paper is as follows. Section 2 presents the hierarchical properties between geometrically similar elements. Section 3 will describe the element splitting technique used to provide geometrically similar elements, and then, Section 4 will present the main characteristics of the hierarchical *h*-adaptive program that uses this element splitting technique. Section 5 will show the natural reordering scheme of the linear system of equations provided by the hierarchical relations. The advantages of using the hierarchical relations will be demonstrated in Section 6, devoted to the numerical examples.

2. PROBLEM DEFINITION AND FEM SOLUTION

We will consider the 2D linear elasticity problem on a bounded domain $\Omega \subset \mathbf{R}^2$. The unknown displacement field **u** is the solution of the boundary value problem

$$\begin{aligned} \nabla\cdot\boldsymbol{\sigma}(\mathbf{u})+\mathbf{b}&=\mathbf{0} && \text{in } \Omega \\ \boldsymbol{\sigma}(\mathbf{u})\cdot\mathbf{n}&=\mathbf{t} && \text{on } \Gamma_N \\ \mathbf{u}&=\tilde{\mathbf{u}} && \text{on } \Gamma_D \end{aligned} \quad (1)$$

where G_N and G_D are the Neumann and Dirichlet boundaries, with $\partial\Omega=\Gamma_N \cup \Gamma_D$, $\Gamma_N \cap \Gamma_D=\emptyset$. **b** are body loads per unit volume, **t** are tractions applied on G_N (with **n** the normal vector to the boundary) and $\tilde{\mathbf{u}}$ are the prescribed displacements on G_D. In the weak form the problem reads:
Find $\mathbf{u}\in V$ such that:

$$\forall \mathbf{v}\in V \qquad a(\mathbf{u},\mathbf{v})=l(\mathbf{v}) \quad (2)$$

being V the standard test space for elasticity problems and

$$\begin{aligned} a(\mathbf{u},\mathbf{v}) &:= \int_\Omega \boldsymbol{\sigma}(\mathbf{u}):\boldsymbol{\varepsilon}(\mathbf{v})d\Omega \\ l(\mathbf{v}) &:= \int_\Omega \mathbf{b}\cdot\mathbf{v}d\Omega+\int_{\Gamma_N}\mathbf{t}\cdot\mathbf{v}d\Omega \end{aligned} \quad (3)$$

where **s** and **e** represent the stresses and strains.

Let $\mathbf{u}^h$ be a finite element approximation to **u** that lies in a functional space $V^h \subset V$ associated with a mesh of isoparametric finite elements of characteristic size h, such that

$$\forall \mathbf{v}^h \in V^h,\ a(\mathbf{u}^h,\mathbf{v}^h)=l(\mathbf{v}^h) \quad (4)$$

Using a variational formulation of the problem in (1) and a finite element approximation $\mathbf{u}^h=\mathbf{N}\mathbf{u}^e$, where **N** are the shape functions matrix and $\mathbf{u}^e$ are the nodal displacements, the following system of linear equations to evaluate $\mathbf{u}^e$ is obtained

$$\mathbf{K}\mathbf{U}=\mathbf{f} \quad (5)$$

where **K** and **f** are obtained after the assembly of the stiffness matrices $\mathbf{k}_e$ and equivalent force vectors $\mathbf{f}_e$ of each element e given by:

$$\mathbf{k}_e=\int_{\Omega_l}\mathbf{B}^t\mathbf{D}\mathbf{B}|\mathbf{J}|d\Omega_l \quad (6)$$

$$\mathbf{f}_e=\int_{\Omega_l}\mathbf{N}^t\mathbf{b}|\mathbf{J}|d\Omega_l+\int_{\Gamma_e\ \Gamma}\mathbf{N}^t\mathbf{t}d\Gamma \quad (7)$$

where **, D** is the elasticity matrix that defines the stresses as **s** = **De**, **B** are derivatives of the shape functions **N**, W_l is the element domain in local coordinates and **J** is the Jacobian matrix. The following section will show the relations between the terms used to evaluate $\mathbf{k}_e$ of geometrically similar elements.

3. HIERARCHICAL PROPERTIES BETWEEN GEOMETRICALLY SIMILAR ELEMENTS

To the authors' knowledge, there is only a reduced number of references related to this topic. Tabarraei and Sukumar [11] demonstrated that, for the special case of quadtree meshes, for the Poisson equation and for the elasticity problem, the stiffness matrix of a subelement is the same as the stiffness matrix of the parent. Suzuki and Tabata [20] also showed the importance of reusing previous calculations studying the structure of the finite element mass and stiffness matrices of congruent subdomains (each of them being an image of a reference subdomain by an affine transformation, see [20] for further details). As a result, Suzuki and Tabata were able to express the global matrices as functions of the sub-matrices in the reference subdomain. This reduced the amount of memory requirements for the storage of the matrices and allowed for the use of a domain decomposition solver.

If a finite element W_i can be obtained by translation and/or scaling of an original element W_0, see Figure 1, a hierarchical relationship related to the element geometry arises between the terms involved in the evaluation of the element stiffness matrices of both elements.

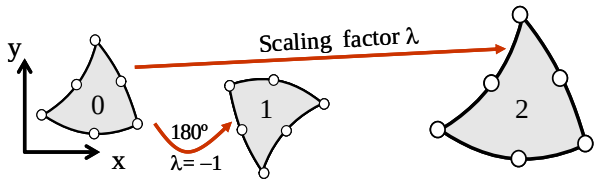


Figure 1. Elements obtained by translation and/or scaling from an original element.

Therefore, the evaluation of the stiffness matrix corresponding to an element that keeps a geometrical similarity with respect to an original element requires no additional computations if the matrices associated to the original element are available. These findings where previously exposed in [21].

Proposition. *Let* W_0 *be an isoparametric finite element defined by its nodal coordinates* $\mathbf{P}_{0n}=(x_{0n},y_{0n},z_{0n})$, $n=1,\ldots,nnpe$, *being 'nnpe' the number of nodes per element.*

Let W_1 *be another isoparametric finite element, geometrically similar to* W_0, *such that its nodal coordinates,* $\mathbf{P}_{1n}$, *can be obtained using a linear transformation by scaling the nodal coordinates of* W_0 *by an scaling factor* l *and by imposing a translation by means of a vector* $\mathbf{T}=(x_t, y_t, z_t)$:

$$\mathbf{P}_{1n} = (x_{1n}, y_{1n}, z_{1n}) = (\lambda x_{0n} + x_t, \lambda y_{0n} + y_t, \lambda z_{0n} + z_t)$$
$$= \lambda(x_{0n}, y_{0n}, z_{0n}) + (x_t, y_t, z_t) = \lambda \mathbf{P}_{0n} + \mathbf{T} \quad (8)$$
$$n = 1,..., nnpe$$

The following relations are obtained:

Relation 1: $\mathbf{J}_1(\xi,\eta,\tau) = \lambda \mathbf{J}_0(\xi,\eta,\tau)$

Relation 2: $inv(\mathbf{J}_1(\xi,\eta,\tau)) = \frac{1}{\lambda} inv(\mathbf{J}_0(\xi,\eta,\tau))$

Relation 3: $|\mathbf{J}_1(\xi,\eta,\tau)| = \lambda^d |\mathbf{J}_0(\xi,\eta,\tau)|$, *where d is the problem dimension (2 for 2D,…)*

Relation 4: $\mathbf{B}_1(\xi,\eta,\tau) = \frac{1}{\lambda}\mathbf{B}_0(\xi,\eta,\tau)$

Relation 5: *if* **D=constant** $\mathbf{k}_1 = \lambda^{d-2}\mathbf{k}_0$

Proof of Relation 1: The Jacobian matrix associated to the isoparametric coordinates transformation of a point in W_0 of local and global coordinates (x,h,t) and (x_0,y_0,z_0) is calculated as:

$$\mathbf{J}_0(\xi,\eta,\tau) = \begin{bmatrix} \frac{\partial x_0}{\partial \xi} & \frac{\partial y_0}{\partial \xi} & \frac{\partial z_0}{\partial \xi} \\ \frac{\partial x_0}{\partial \eta} & \frac{\partial y_0}{\partial \eta} & \frac{\partial z_0}{\partial \eta} \\ \frac{\partial x_0}{\partial \tau} & \frac{\partial y_0}{\partial \tau} & \frac{\partial z_0}{\partial \tau} \end{bmatrix} \quad (9)$$

Considering the shape functions N_n(x,h,t) in the interpolation of global coordinates, the first term in $\mathbf{J}_0$(x,h,t), and similarly any other term, will be given by:

$$\frac{\partial x_0}{\partial \xi} = \sum_n^{nnpe} \frac{\partial N_n(\xi,\eta,\tau)}{\partial \xi} x_{0n} \quad (10)$$

Consider now the following relations:

$$x_{1n} = \lambda x_{0n} + x_t \ , \ \sum_n^{nnpe} N_n = 1 \quad \Rightarrow \quad \sum_n^{nnpe} \frac{\partial N_n}{\partial \xi} = 0$$

The first term of the Jacobian matrix $\mathbf{J}_1$(x,h,t) associated to element W_1 can be evaluated, considering (3) and the previous relations, as

$$\frac{\partial x_1}{\partial \xi} = \sum_n^{nnpe} \frac{\partial N_n}{\partial \xi} x_{1n} = \sum_n^{nnpe} \frac{\partial N_n}{\partial \xi}(\lambda x_{0n} + x_t)$$
$$= \lambda \left(\sum_n^{nnpe} \frac{\partial N_n}{\partial \xi} x_{0n} \right) + \left(\sum_n^{nnpe} \frac{\partial N_n}{\partial \xi} \right) x_t = \lambda \frac{\partial x_0}{\partial \xi} \quad (11)$$

Similar relations would be found for all terms in $\mathbf{J}_1$(x,h,t). Therefore, the following relation is obtained

$$\mathbf{J}_1(\xi,\eta,\tau) = \lambda \mathbf{J}_0(\xi,\eta,\tau) \quad \blacksquare$$

Proof of Relations 2 and 3: These relations are immediately derived from Relation 1 ■.

Proof of Relation 4: All terms in the $\mathbf{B}(x,h,t)$ matrix are first partial derivatives of shape functions with respect to global coordinates, which, for element W_0, can be evaluated as:

$$\begin{Bmatrix} \frac{\partial N_{0n}}{\partial x} \\ \frac{\partial N_{0n}}{\partial y} \\ \frac{\partial N_{0n}}{\partial z} \end{Bmatrix} = \mathbf{J}_0^{-1} \begin{Bmatrix} \frac{\partial N_n}{\partial \xi} \\ \frac{\partial N_n}{\partial \eta} \\ \frac{\partial N_n}{\partial \tau} \end{Bmatrix} \quad (12)$$

Considering Relation 2, the following relation is obtained for element W_1:

$$\begin{Bmatrix} \frac{\partial N_{1n}}{\partial x} \\ \frac{\partial N_{1n}}{\partial y} \\ \frac{\partial N_{1n}}{\partial z} \end{Bmatrix} = \mathbf{J}_1^{-1} \begin{Bmatrix} \frac{\partial N_n}{\partial \xi} \\ \frac{\partial N_n}{\partial \eta} \\ \frac{\partial N_n}{\partial \tau} \end{Bmatrix} = \frac{\mathbf{J}_0^{-1}}{\lambda} \begin{Bmatrix} \frac{\partial N_n}{\partial \xi} \\ \frac{\partial N_n}{\partial \eta} \\ \frac{\partial N_n}{\partial \tau} \end{Bmatrix} = \frac{1}{\lambda} \begin{Bmatrix} \frac{\partial N_{0n}}{\partial x} \\ \frac{\partial N_{0n}}{\partial y} \\ \frac{\partial N_{0n}}{\partial z} \end{Bmatrix} \quad (13)$$

All terms in matrices $\mathbf{B}$(x,h,t) for W_0 and W_1 are related by 1/l. Therefore, we finally obtain

$$\mathbf{B}_1(\xi,\eta,\tau) = \frac{1}{\lambda}\mathbf{B}_0(\xi,\eta,\tau) \quad \blacksquare$$

Proof of Relation 5: Assuming that **D** = **constant** (**s** = **De)**, the stiffness matrices for elements W_0 and W_1 will be evaluated as:

$$\mathbf{k}_0 = \int_{\Omega_l} \mathbf{B}_0^t \mathbf{D} \mathbf{B}_0 |\mathbf{J}_0| d\Omega_l \, , \, \mathbf{k}_1 = \int_{\Omega_l} \mathbf{B}_1^t \mathbf{D} \mathbf{B}_1 |\mathbf{J}_1| d\Omega_l \quad (14)$$

Taking into account Relations 2 and 3 in the expression corresponding to $\mathbf{k}_1$:

$$\mathbf{k}_1 = \int_{\Omega_l} \frac{\mathbf{B}_0^t}{\lambda} \mathbf{D} \frac{\mathbf{B}_0}{\lambda} \lambda^d |\mathbf{J}_0| d\Omega_l = \int_{\Omega_l} \lambda^{d-2} \mathbf{B}_0^t \mathbf{D} \mathbf{B}_0 |\mathbf{J}_0| d\Omega_l \quad \blacksquare$$
$$= \lambda^{d-2} \mathbf{k}_0$$

Consequently, for the 2D case where $d = 2$, $\mathbf{k}_0$ and $\mathbf{k}_1$ are exactly the same matrices. For the 1D case these matrices will be related by the constant factor 1/l, and by l in the 3D case.

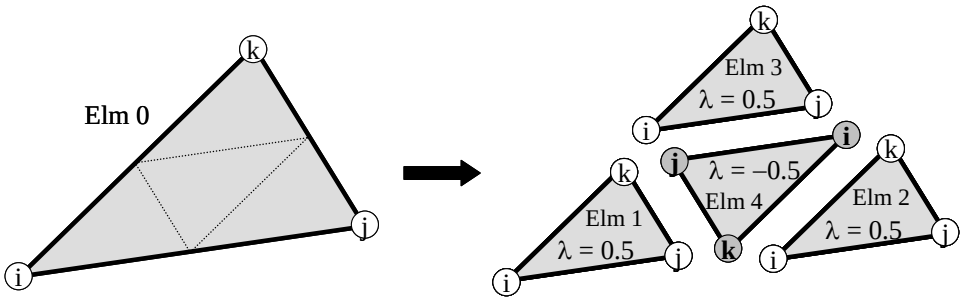
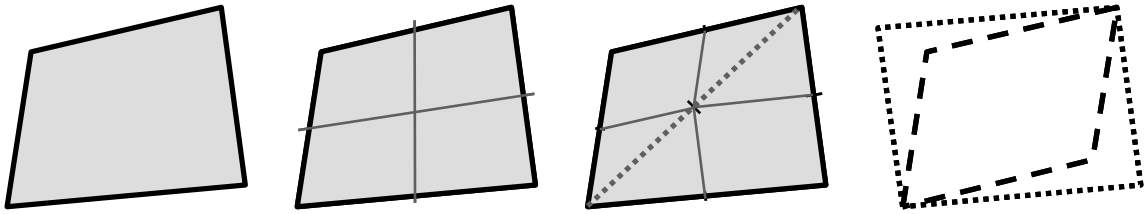


Figure 2. Subdivision of *parent* triangular element into 4 *child* elements.



a) *Parent* element b) Standard subdivision c) Proposed subdivision d) Dummy *parent* elements

Figure 3. Subdivision of bi-linear *parent* element into 4 *child* elements.

4. SIMILARITY RELATIONS IN REFINEMENT BY MESH SPLITTING IN 2D

4.1. Linear triangular elements

Figure 2 shows a *parent* linear triangular element that has been subdivided into 4 *child* elements placing new nodes at the mid-side point of each element side.

The figure shows the scaling factor l that relates *parent* and *child* elements. With the numbering pattern adopted for the new elements, a geometrical similarity relation with respect to *Elm 0* can be obtained even for *Elm 4*, whose associated scaling factor value is l = -0.5. The scaling factor corresponding to *child* elements 1, 2 and 3 is l = 0.5.

4.2. Bi-linear quadrilaterals

Figure **3** shows the subdivision process of a bi-linear element into 4 new elements. The original (*parent*) element is represented in
Figure **3**.a).
Figure **3**.b) shows the standard splitting procedure. In this procedure the new elements are obtained by using 2 straight lines that join the mid-side points of opposite sides of the element. This technique will provide *child* elements geometrically similar to the *parent* element only in the case where the *parent* element is a parallelogram. This picture shows that, in general, the *child* elements created using this technique are not geometrically similar to the *parent* element.

For this linear quadrilateral elements, the element subdivision procedure proposed in this paper and represented in
Figure **3**.c), consists of joining the mid-side points of each side of the element with the mid-side point of any of the diagonals of the quadrilateral (the longest diagonal has been considered in the implementation). As shown in Figure **3**.c, the *child* elements located over the selected diagonal will be geometrically similar to the *parent* element, whereas the other two *child* elements created, which are not similar to the *parent* element, will be parallelograms. Therefore, if any of the *child* elements is further subdivided, the new elements created will always be geometrically similar to either the original *parent* element (
Figure **3**.a)) or the *dummy parallelogram parent* elements represented in
Figure **3**.d).

Figure 4 represents a sequence of successive subdivisions of the element represented in
Figure **3**.a). The same colour has always been used to represent all the geometrically similar elements. It can be clearly observed that, for any subdivision level, only 3 different kinds of geometrically similar quadrilaterals will appear.

4.3. Elements over curved boundaries

Two different types of *parent* elements are generated during the mesh refinement based on subdivision of elements. *Type A* elements are defined as *parent* elements whose *child* elements are geometrically similar which will, therefore, inherit the element calculations. On the other side, *type B* elements are defined as *parent* elements

with at least one *child* element not geometrically similar. The *child* elements created from a *type B* element will not inherit element calculations.

As represented in Figure 5, when subdividing one element, if any of its sides lies over a curved boundary, the new nodes to be generated over this boundary will not be located over the straight line that defines the side of the original element. Therefore, there will not be a geometrical similarity relation between *parent* and *child* elements. According to the previous paragraph's definitions, this kind of *parent* elements will be of *type B*. Let W_p be a *parent* element with none of its sides located over a curved boundary. As previously explained, under these circumstances, the element can be subdivided in such a way that the geometrical similarity relations between *parent* and *child* elements hold. Therefore, element W_p is of *type A*.

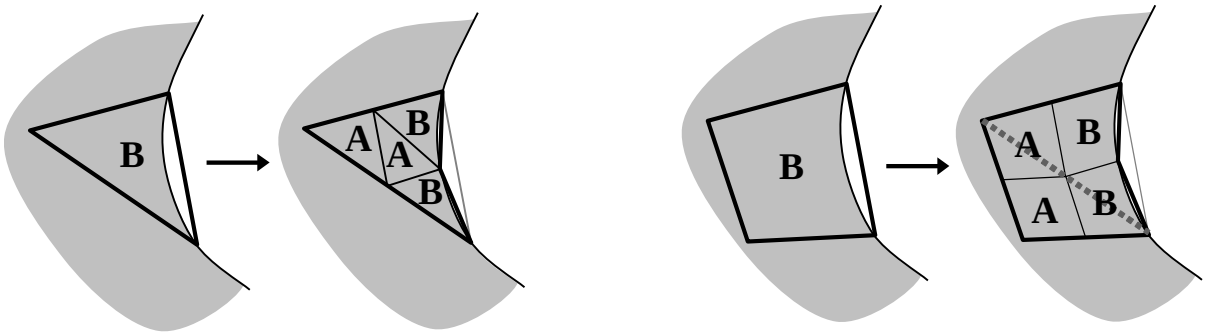


Figure 5. Subdivision of elements over curved boundaries. Type A and type B elements.

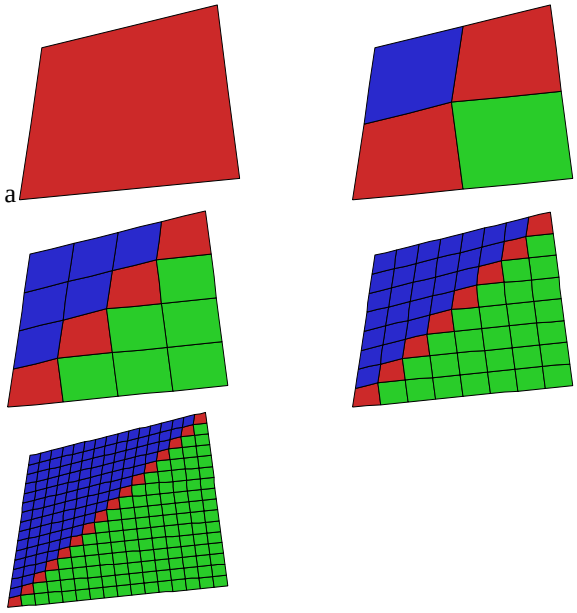


Figure 4. Quadrilateral: Subdivision sequence.

4.4. Higher order 2D elements

Let's consider quadratic triangles and serendipity quadrilaterals with mid-side nodes located at the mid-side point of the vertex nodes at each side of the element in the global reference system. The local to global mapping functions $x(x,h)$ and $y(x,h)$ of these elements and their linear version are exactly identical. Thus, the subdivision procedure described for linear triangles and bi-linear quadrilaterals will also provide *child* elements geometrically similar to their *parent* elements for these elements. If quadratic triangles or serendipity quadrilaterals are not defined by straight line segments with mid-side nodes located over the mid-side point of each side of the element, then, they will be *type B* elements.

4.5. Refinement level

The *refinement level* r_e for an element *e* is defined as the number of subdivision steps required to obtain element *e* from its *ancestor* in the original mesh. According to this definition, the refinement level for every element in the original mesh will be $r_e = 0$, the refinement level for elements directly obtained from subdivision of elements with $r_e = 0$ will be $r_e = 1$, $r_e = 2$ for elements obtained by subdivision of elements with $r_e = 1$, and so on.

Let W_p be a Type A *parent element*. The scaling factor with respect to W_p (or, in the case of quadrilateral elements, with respect to the dummy parent elements associated to W_p, see
Figure **3**.d) corresponding to the elements obtained by *s* successive subdivisions of W_p will simply be a function of the difference between refinement levels, *s*. Thus, the scaling factor will be $|l| = 0.5^s$.

5. A HIERARCHICAL *H*-ADAPTIVE CODE: DATA STRUCTURE

A code for the resolution of the 2D linear elasticity problem, which can be described as a Finite Element hierarchical *h*-adaptive program, has been developed

using Matlab® [22]. The code, which requires an initial conforming mesh of linear triangles or quadrilaterals, uses a hierarchical data structure to drive the *h*-refinement process, mainly based on the following objects, whose description can be found in appendix A:

- **Node**: stores data associated to each node.
- **Element**: stores data associated to each element.
- **KMatrix**: stores data associated to each stiffness matrix. Geometrically similar elements will be related to a single KMatrix object.

The idea behind the development of this code was to create a framework to test the benefits of the use of hierarchical relations in *h*-adaptivity. Apart from the hierarchical relations between *parent* and *child* elements described in previous sections, the program also uses the following relations.

Neighbourhood relations.

- *Element to element neighbourhood relations*. The following information is stored at each side of each element: neighbour element and neighbour element's side number.
- *Element - boundary relations*. For sides of elements located over the boundary of the domain the boundary identification code is also stored.

These two relations (element-element, element-boundary) can be inherited by the new elements created during the mesh refinement. The use of this information simplifies and accelerates the refinement process.

Nodal hierarchical relations.

During the splitting process of each element, new nodes are created. The location of these new nodes can always be expressed as a function of the location of the nodes of the parent element. To do this, the code stores the values of the parent element shape functions evaluated at the location of the new node. Therefore, each new node will keep information about its *parent* nodes (the nodes of the parent element) and the influence of each of these *parents* over the node (value of the element shape functions calculated at the location of the new node).
This hierarchical *parents-child* relationship between nodes has two main uses in the program:

- *MPC's equations.* These *parents-child* relations can be used to impose the multi-point constraint equations used to ensure C^0 continuity between adjacent elements with different subdivision levels.
- *Data interpolation-extrapolation between different meshes.* Nodal values evaluated in one mesh can be easily interpolated to more refined meshes or extrapolated to coarser meshes by using the *parents-child* relations between nodes.

5.1. Further advantages of the hierarchical data structure

The hierarchical data structure has some further advantages:

- *Stress evaluation at Gauss points.* These values can be easily evaluated by using $\mathbf{s}_G = \mathbf{DBu}^e$ ($\mathbf{u}^e$ displacement vector at nodes of element *e*, **D** Hook's matrix corresponding to the stress-strain relation $\mathbf{s} = \mathbf{De}$) because the values of the **B** matrix evaluated at Gauss points are available for the elements used to create each of the *KMatrix* objects.

- *Determinant of the Jacobian matrix at Gauss points.* The evaluation of any result involving element integrals (equivalent load vector corresponding to body loads, strain energy, energy norm, error estimation in energy norm,…) requires the evaluation of the determinant of the Jacobian matrix $|\mathbf{J}|$ at Gauss Points. These values are available for the elements used to create each of the *KMatrix* objects.

6. STIFFNESS MATRIX REORDERING

Matrix reordering plays an important role on the performance of the direct solver. Reordering the columns of a matrix can often make its LU or QR factors sparser. Reordering the rows and columns can often make its Cholesky factorization sparser. This allows for a reduction of the time required to obtain the solution of the problem. Finding the optimal ordering is usually not possible, but finding a good ordering is. Matlab® [22], which has been used to develop our FE code, incorporates a number of reordering algorithms, some of which involve the use of an iterative process to obtain the reordering.

This section is intended to show how the hierarchical data structure of the program can be used to directly obtain a reordering of the system matrix that speeds up the Cholesky factorization process.

6.1. Reordering based on a nested domain decomposition (NDD)

The mesh splitting technique used produces a natural decomposition of the domain. The elements contained in the initial mesh can be considered as the subdomains into which the original domain is divided. For the following meshes, the elements to be considered in each subdomain are those obtained by the subdivision of the original elements. The subdomains defined by the elements

included in the first mesh of the analysis can be termed 0-*Level* subdomains. This idea can be recursively applied into each of the original subdomains. Thus, as represented in Figure 6, new sub-subdomains (1-*Level* subdomains) could be defined into the 0-*Level* subdomains, and so forth.

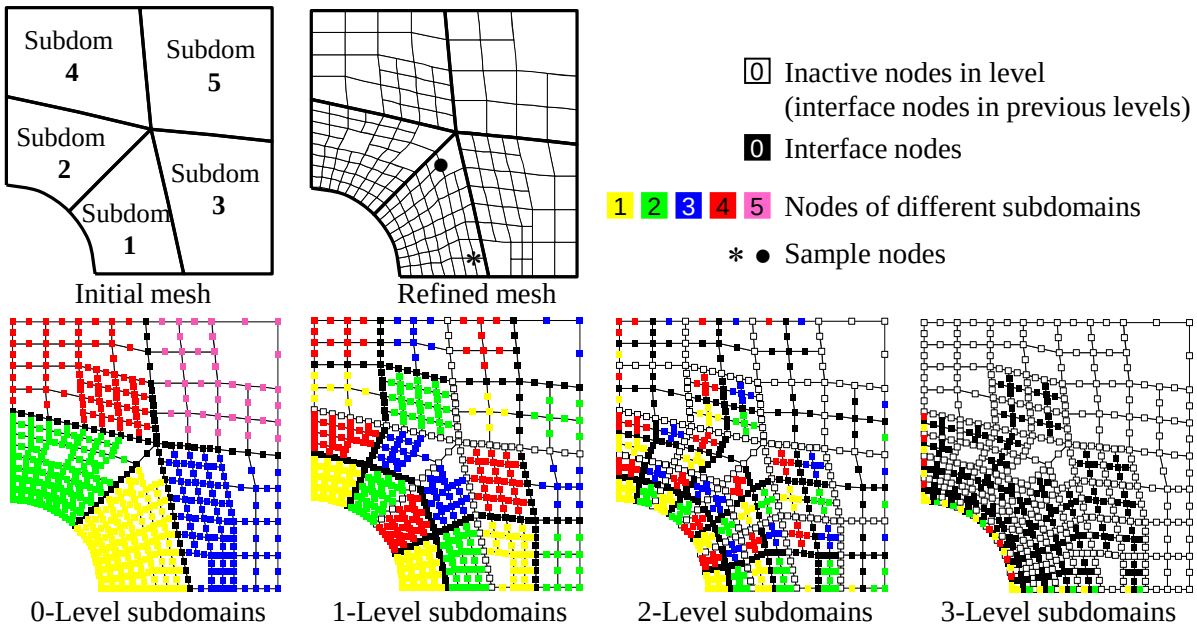


Figure 6. FE model. Subdomains in different levels.

The value of a new nodal property called .*Subdom*, which stores a code that indicates the subdomain number in which each node is created at each subdivision level, is easily obtained by using the hierarchical data structure during the mesh generation process. The value 0 will be assigned to those nodes created over the interfaces between the subdomains. Taking this into account the sample nodes highlighted in **Figure 6** would have the codes shown in **Figure 7**:

		0-Level	1-Level	2-Level	3-Level
Sample node ∗	Node.Subdom =	1	2	2	0
Sample node •	Node.Subdom =	1	3	0	0

Figure 7. Codes for sampling nodes shown in Figure 6.

The value of this property allows for a simple reordering of the linear system of equations using a dictionary-type reordering (*sortrows* command in Matlab® [22]). The reordering thus obtained will be denoted as *NDD reordering*. Figure 8.a) shows the original structure of a stiffness matrix corresponding to a problem with 3 elements in the original mesh which has been uniformly refined. The arrowhead-like structure represented in Figure 8.b) is obtained if the system of equations is reordered taking into account the value of the .*Subdom* property for the 0-*Level*. Observe that the degrees of freedom (*dofs*) placed in the first 0-*Level* subdomain (element number 1 of the original mesh) are located first; then, those in the second 0-*Level* subdomain, and so forth. The last *dofs* of the reordered matrix correspond to the *dofs* of the interfaces between the 0-*Level* subdomains. Finally, the structure represented in Figure 8.c) is obtained if the complete nested subdomains structure is used to reorder the system of equations.

The numerical results presented in Section 7 will show the advantages obtained by using the natural reordering provided by the hierarchical *h*-adaptivity code.

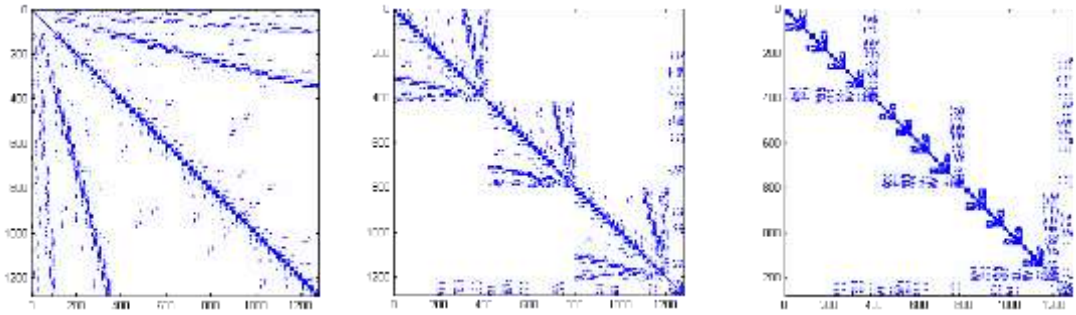
7. NUMERICAL EXAMPLES

7.1. Domain with straight boundaries

When evaluating element stiffness matrices, the biggest advantages of the program emerge when the boundary of the component can be represented by straight-line segments. Under this situation, all the elements in the mesh will be *Type A elements*. Then, the only *KMatrix* objects to be evaluated will correspond to the elements in the original mesh and their dummy elements (see Figure **3**), if the elements are non-parallelograms quadrilaterals.

As an example of this kind of domains, the plate with a crack represented in Figure 9 has been studied. Due to the

problem's symmetry, only the right hand side of the domain has been considered in the analyses.



a) Original b) Reordered matrix considering 0-Level subdomains c) Reordered matrix considering full nested subdomains structure

Figure 8. Stiffness matrix for a problem with 3 subdomains in the original mesh.

The problem has been studied using a *h*-adaptive analysis based on the estimation of the discretization error in energy norm. In order to evaluate an estimate $\|\mathbf{e}_{es}\|$ of the exact value of the discretization error in energy norm $\|\mathbf{e}_{ex}\|$, Zienkiewicz-Zhu [23] developed the ZZ estimator proposing the use of the following expression

$$\|\mathbf{e}_{es}\|^2 = \int_\Omega \left(\boldsymbol{\sigma}^* - \boldsymbol{\sigma}^h\right)^T \mathbf{D}^{-1} \left(\boldsymbol{\sigma}^* - \boldsymbol{\sigma}^h\right) d\Omega \qquad (15)$$

where domain W can refer to either the whole domain or a local (element) subdomain, $\boldsymbol{\sigma}^h$ represents the stresses evaluated using the Finite Element Method, $\boldsymbol{\sigma}^*$ is the so-called smoothed or recovered stress field, that is a better approximation of the exact solution than $\boldsymbol{\sigma}^h$.

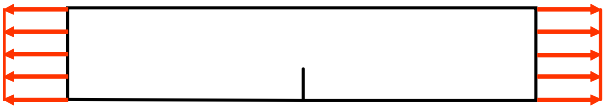


Figure 9. Plate with crack under traction.

Equation (15) is rewritten as follows in terms of local coordinates for the evaluation of the error in energy norm at element *e*:

$$\|\mathbf{e}_{es}\|_e^2 = \int_{\Omega_l} \left(\boldsymbol{\sigma}^* - \boldsymbol{\sigma}^h\right)^T \mathbf{D}^{-1} \left(\boldsymbol{\sigma}^* - \boldsymbol{\sigma}^h\right) |\mathbf{J}| \, d\Omega_l \quad (16)$$

We have used *Relation 3* to reduce the computational cost of the evaluation of |**J**|. Observe that the computational cost of any domain integral can be reduced by means of the use of *Relation 3*.

An enhanced version [24] of the Superconvergent Patch Recovery technique (SPR) [25] has been employed in the error estimation process. This version of the SPR technique provides a very accurate recovered stress field as it uses constraint equations to impose the exact local satisfaction of the equilibrium and compatibility equations in the patch of elements surrounding each vertex node.

The criterion used to define the size of the elements in new meshes is based on the criterion of *minimization of the number of elements in the new mesh* described by Ladeveze *et al.* [26, 27] and Coorevits *et al* [28].

Figure 10 shows two sequences of *h*-adapted meshes, obtained with quadratic triangles and quadrilaterals, used to analyse the problem represented in Figure 9. Note that some of the quadrilateral elements used in the first mesh of the sequence were deliberately distorted to illustrate the use of quadrilateral elements of arbitrary shapes. In Figure 10 all the elements associated to the same *KMatrix* object have been represented with the same colour. The statistics of the mesh sequence have been shown in

.

clearly shows the advantages obtained when the hierarchical relations between elements are used. For example, when triangular elements have been used, the last mesh in the sequence contains 9135 elements, some of them with a refinement level r = 11, and has required the creation of 12168 elements. However, only 36 *KMatrix* objects have been evaluated to create the problem stiffness matrix for this mesh with 19592 nodes.

7.2. Domain with curved boundaries

Parent elements with one or more of their sides lying over curved boundaries will not be geometrically similar to their *child* elements. Therefore, when the mesh is refined, new *KMatrix* objects will be required to be created. In any case, it must be taken into account that this will only happen along curved boundaries. It can be intuitively observed that, whereas the number of

elements, in the 2D case, will grow as a function of the area of the domain, the number of *KMatrix* objects to be created will be a function of the length of the curved boundaries, i.e. one less dimension.

A *Goal Oriented h-Adaptive* process has been used in this case. The adaptive process is based on the estimation of the error in the magnitude of interest $Q(\mathbf{e}_{es})$ using the

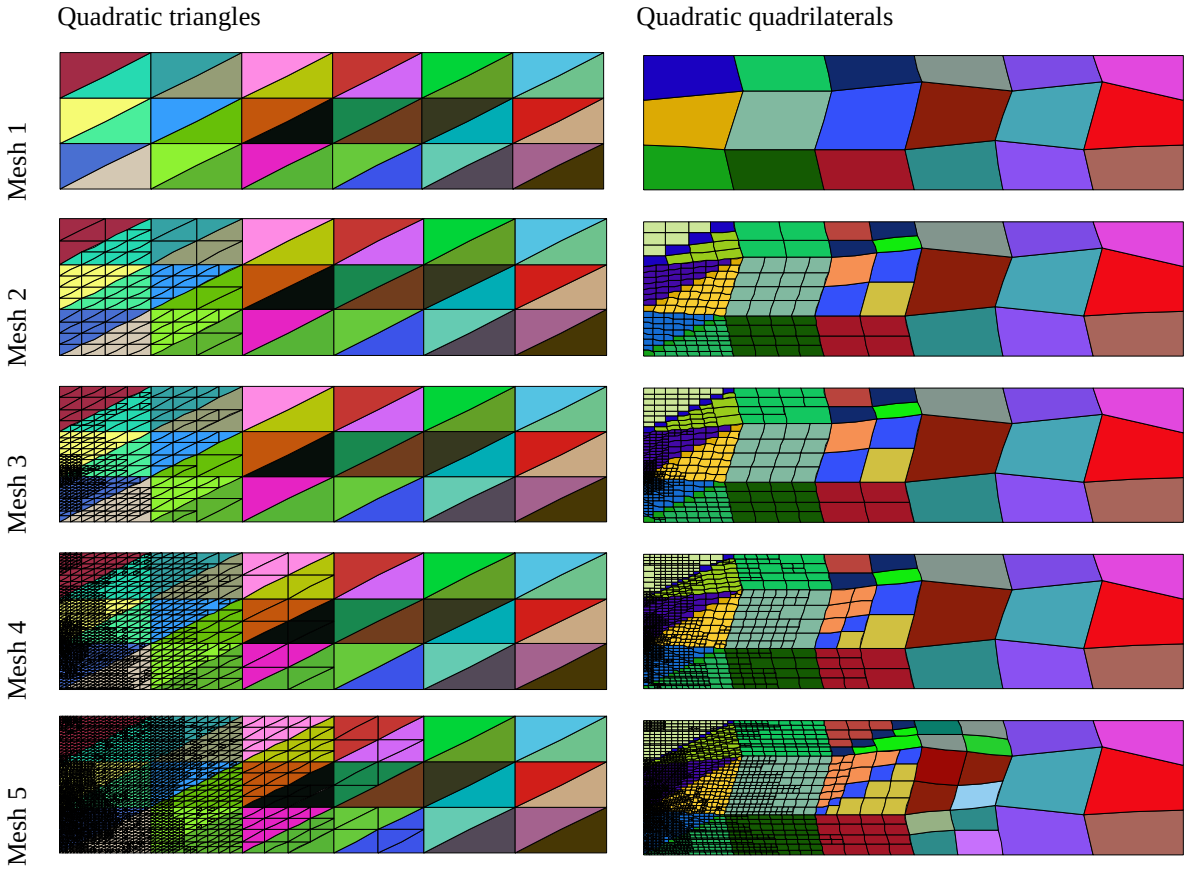


Figure 10. Plate with crack under traction. Sequence of *h*-adapted meshes. Representation of elements with the same *KMatrix* object.

Table 1. Plate with crack under traction. *H*-adaptive refinement data.

	Triangular elements				**Quadrilateral elements**			
Mesh	Nodes	Created Elements	Active Elements	*KMatrix* Obj. evaluated	Nodes	Created Elements	Active Elements	*KMatrix* Obj. evaluated
1	91	36	36	36	73	18	18	46
2	359	196	156	36	678	262	201	46
3	1496	896	681	36	1881	746	564	46
4	5781	3572	2688	36	5479	2218	1668	46
5	19592	12168	9135	36	15594	6446	4839	46

The example presented in this section corresponds to a gravity dam. The initial mesh used in this problem has been represented in Figure 11. The objective of the analysis is the evaluation of the mean value of the von-Mises stress in an area of interest which has been defined by the highlighted elements. Quadratic triangular elements have been used in this problem.

recovery type error estimator given in the following expression for element *e*, as proposed for example in [29] and [30]:

$$Q(\mathbf{e}_{es})_e = \int_{\Omega_l} \left(\boldsymbol{\sigma}_p^* - \boldsymbol{\sigma}_p^h\right)^T \mathbf{D}^{-1} \left(\boldsymbol{\sigma}_d^* - \boldsymbol{\sigma}_d^h\right) |\mathbf{J}| \, d\Omega_l \quad (17)$$

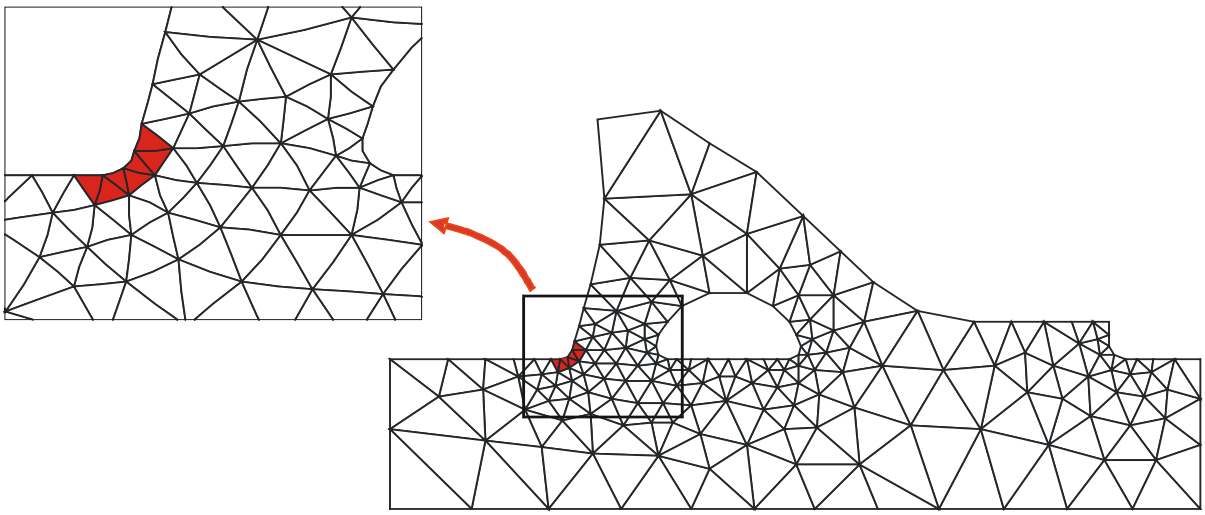


Figure 11. Gravity dam. Mesh 1. Quadratic triangular elements.

where $\boldsymbol{\sigma}_p^h$ and $\boldsymbol{\sigma}_p^*$ represent the FE and recovered stress fields corresponding to the primal problem and $\boldsymbol{\sigma}_d^h$ and $\boldsymbol{\sigma}_d^*$ represent those corresponding to the dual problem used to extract the magnitude of interest. The standard SPR technique was used in this case to obtain the recovered stress fields $\boldsymbol{\sigma}_p^*$ and $\boldsymbol{\sigma}_d^*$. As in the previous example, *Relation 3* was used in the evaluation of (17) to reduce the computational cost asociated to the evaluation of $|\mathbf{J}|$.

The hierarchical data structure has been used to define the area of interest in more refined meshes, by simply taking into account that when a *parent* element into the area of interest is subdivided into four *children* elements, the *children elements* will also be part of the area of interest. A detail of mesh 4 of the *h*-adaptive mesh sequence around the area of interest is represented in Figure 12.

Figure 13 shows mesh 4 entirely. Elements with the same *KMatrix* object have been represented with the same colour in these two figures. A graphical comparison between the number of *KMatrix* objects evaluated and the number of elements used in each mesh is represented in Figure 14. This figure clearly shows that the number of elements used in each mesh, which is a function of the area, grows faster than the number of *KMatrix* objects evaluated, which is a function of the length of curved contours.

7.3. Solver improvement

Different direct solvers strategies that make use of the Cholesky factorization of the stiffness matrix (symmetrical and positive definite, once the displacement constrains have been imposed) have been considered in this section. The factorization time, evaluated for a sequence of *h*-adapted meshes, used by the *chol* MatLab® command [22] has been considered as the main parameter for the comparison. The MatLab® profiling tool has been used to evaluate the CPU time employed by the *chol(K(p,p))*command, where *K(p,p)* represents a permutation **p** of the system matrix **K**. Each of the five strategies under comparison corresponds to a different permutation **p**. The first four strategies, denoted by *colperm*, *symrcm*, *symamd* and *amd*, respectively correspond to the permutations obtained by using the *colperm*, *symrcm*, *symamd* and *amd* commands available in MatLab®. A description of the algorithms used by each of these commands can be found in the MatLab® 2009b Help [22] and the references therein. The last strategy, denoted by *ndd*, makes use of the natural reordering directly provided by the program's hierarchical structure.

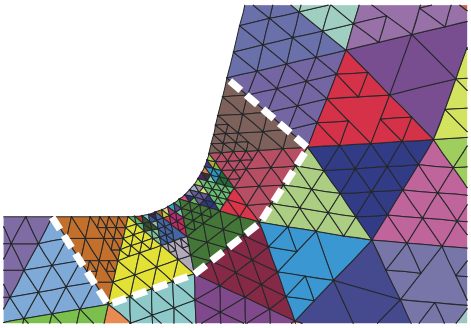
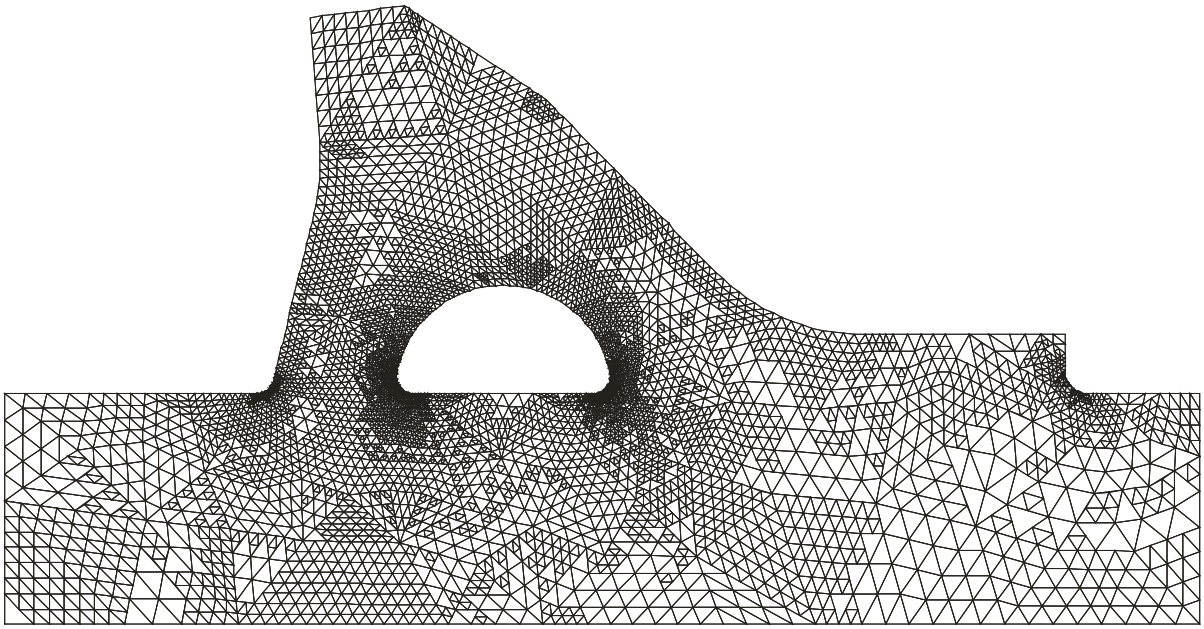
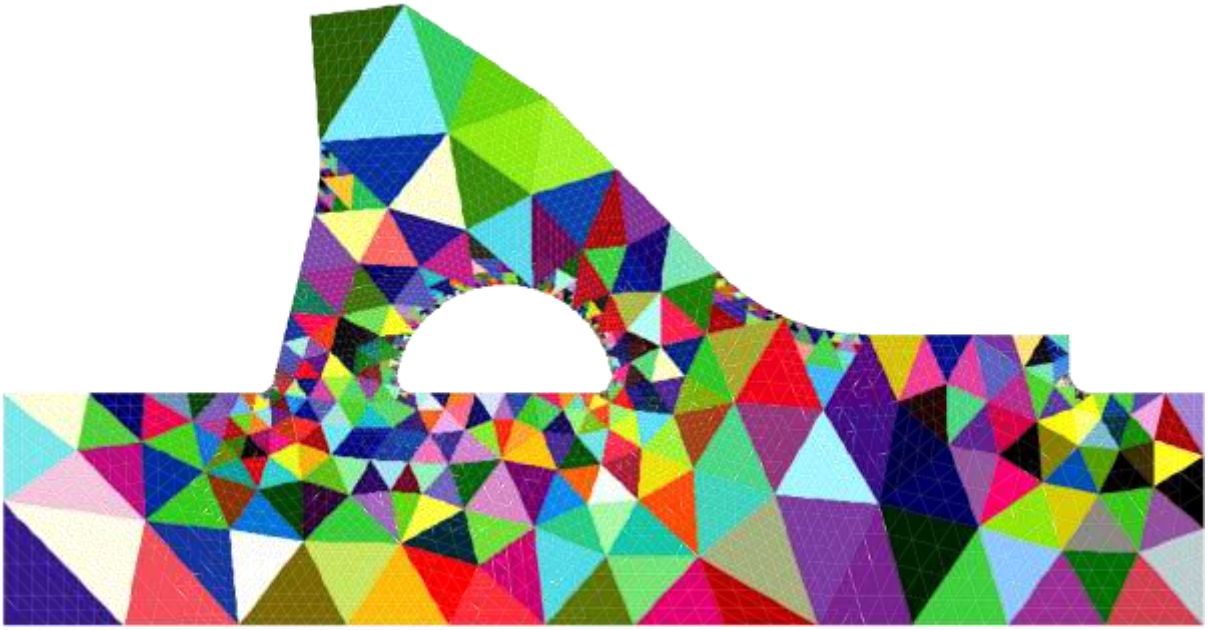


Figure 12. Gravity Dam. Detail of Mesh 4 around the area of interest.



a) *h*-adapted mesh of quadratic triangular elements.



b). Elements with the same *KMatrix* object.

Figure 13. Gravity Dam. Mesh 4.

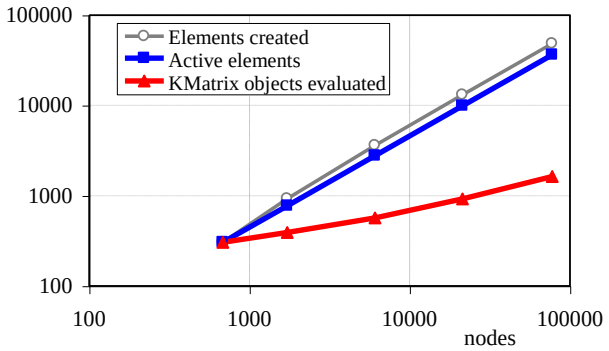


Figure 14. Gravity Dam. Evolution of the number of active elements and the number of *KMatrix* objects.

Figure 16 shows the evolution of the factorization times with respect to the number of degrees of freedom of the system matrices of the mesh sequences. The graph clearly shows that the best performance is obtained with the *ndd* reordering. The *colperm* and *symrcm* reorderings are not competitive with the rest of the methods as they soon produce an *out of memory error* in the computer used for the analyses. The *symamd*, *amd* and *ndd* reorderings produce similar factorization times but, in an average sense, the Cholesky factorization times with the *symamd* and *amd* reorderings required 45% and 33% more time than with the *ndd* reordering.

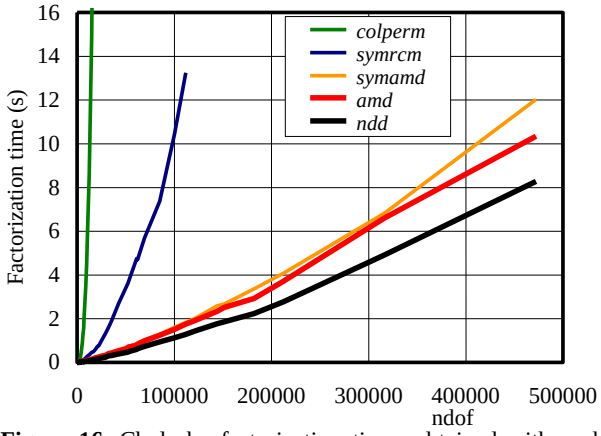


Figure 16. Cholesky factorization times obtained with each reordering scheme for a sequence of *h*-refined meshes.

Figure 15 shows an example of comparison of the reordered matrices obtained with the different permutation schemes. It can be observed that the methods that produce the best performances of the Cholesky

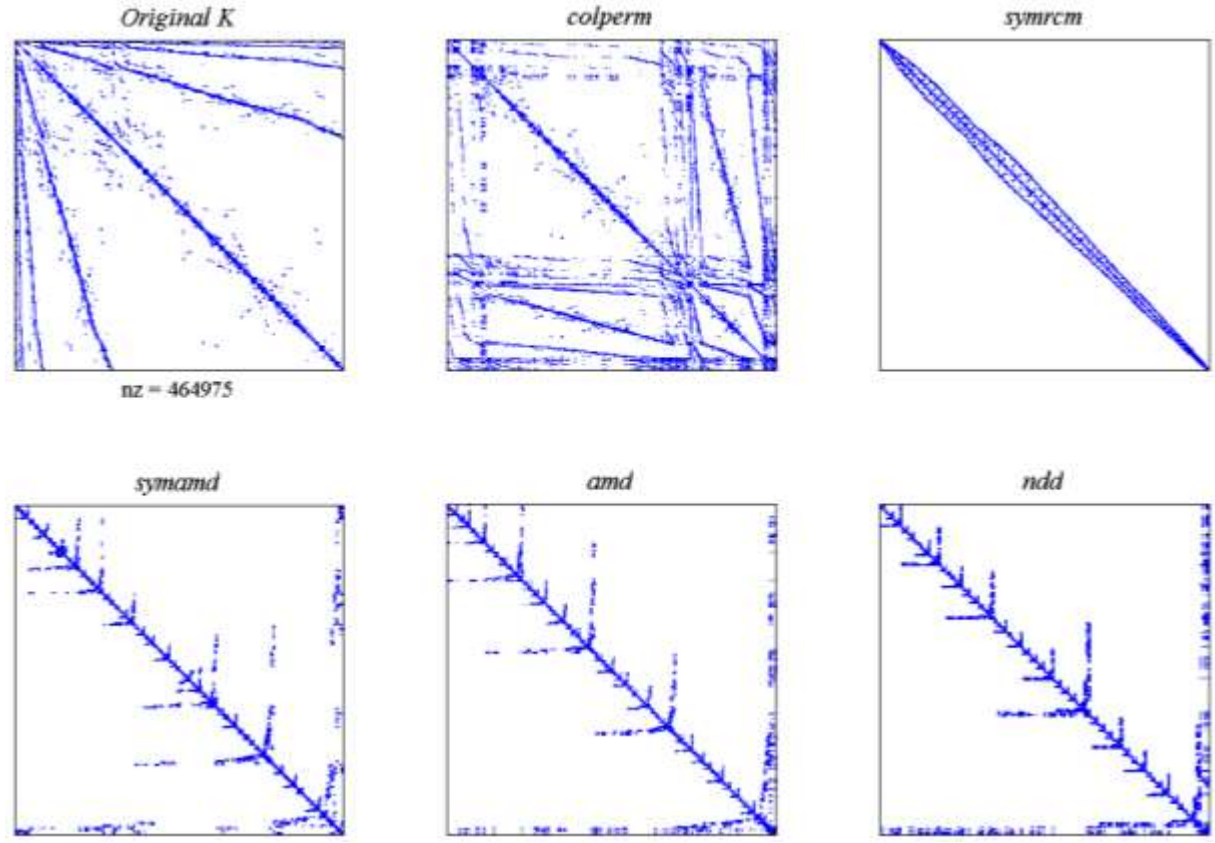


Figure 15. Comparison of reordering schemes of a stiffness matrix, denoted by *Original K* (15054 degrees of freedom). Each plot represents the reordered system matrix obtained by the different permutation.

factorization, the *symamd*, *amd* and *ndd* permutations, produce similar reordering patterns.

Figure **17** represents the factorization of the permuted matrices represented in Figure 15, except for the case of the original matrix, whose factorization generated an *out of memory error*. The *symamd*, *amd* and *ndd* permutations produce similar Cholesky factorizations of a similar number of *non-zero* terms (indicated below each plot).

8. CONCLUSIONS

This paper has presented a *hierarchical h-adaptivity methodology* implemented in a FE code for the resolution of the 2D linear elasticity problem. Linear and quadratic isoparametric triangles and quadrilaterals can be used in the mesh refinement process, which is based on element subdivision and on the use of multi-point constraints to satisfy the C^0 continuity condition between adjacent elements with different refinement levels.

The main conclusions arising from this paper are listed below.

- It has been shown that if two finite isoparametric elements are geometrically similar, the terms involved in the evaluation of their element stiffness matrices are related by a constant value. This constant value is simply a function of the scaling factor that relates both elements. In fact, under this geometrical similarity condition, in the 2D case, the element stiffness matrices for geometrically similar elements are exactly equal if the Hooke's tensor is constant.
- In 2D mesh refinement processes based on element subdivision, if a *parent* element has straight-line contours not lying over curved boundaries with mid-side nodes exactly located over the mid-side point of each side of the element, then, *child* elements can be created geometrically similar to their *parent* element. In this case, the matrices used to evaluate the element stiffness matrix of the *parent* element, after their multiplication by a constant value related to the scaling factor, will be reused by the *child* elements without any further calculation.

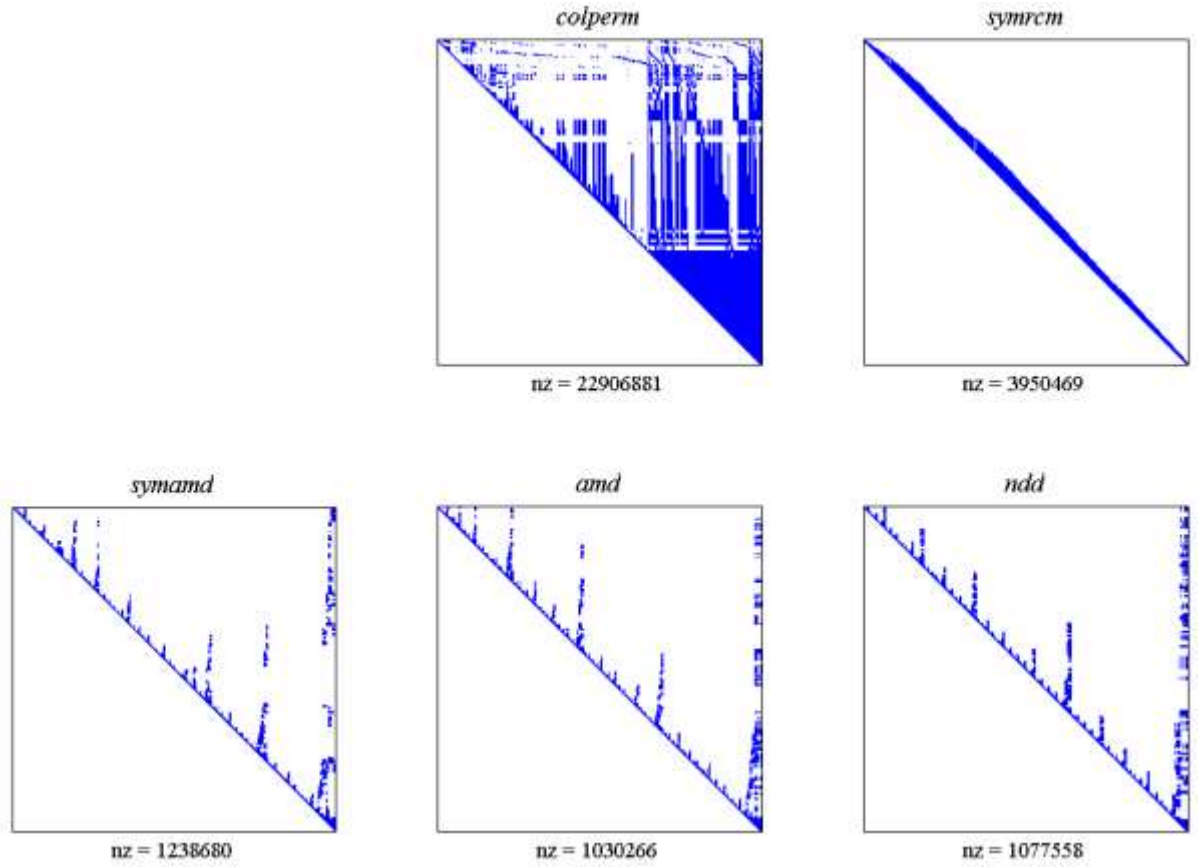


Figure 17. Comparison of the factorizations of the matrices shown in Figure 15. The number of *non-zero* terms of the factorizations is indicated below each plot. The Cholesky factorization of the original matrix, without any reordering, produced an *out of memory error*.

- The *hierarchical h-adaptive* program code notably reduces the computational cost associated to the evaluation of the problem stiffness matrix, but also the cost associated to the computation of any result involving the terms used in the evaluation of the element stiffness matrix (strains and stresses at integration points, volume integrals,…).
- The *hierarchical* data structure is particularly well suited for the implementation of a *multi-grid* solver.
- The *hierarchical* data structure would also simplify the implementation of a *domain decomposition* solver, and, therefore, the parallelization of the process.

The code can be described as *hierarchical* due to the following reasons:

- A *hierarchical* data structure, with *parent-child* relations, is used to store data corresponding to elements and nodes.
- The *hierarchical* data structure simplifies the generation of the new nested meshes.
- The *hierarchical* data structure simplifies the implementation of iterative solvers in which the initial guess could be taken as the solution of previous analyses.
- As in the *p-hierarchical* formulation of the FEM, stiffness matrix information can be reused in other analyses.

Basic implementations of direct and iterative domain decomposition solvers that make use of the nested arrowhead structure have been already developed [31-34].

9. ACKNOWLEDGEMENTS

The authors wish to thank the Spanish Ministerio de Economía y Competitividad for the fiancial support received through the project DPI2013-46317-R and the Generalitat Valenciana through the project PROMETEO/2016/007. The support of the Universidad Politécnica de Valencia is also acknowledged. The authors also want to thank Ana Ródenas's help in the translation of this paper.

10. APPENDIX A: OBJECTS IN THE HIERARCHICAL DATA STRUCTURE

This appendix shows a basic description of the main objects used to create the hierarchical data structure of the FE code:

- **Node**: stores data associated to each node
- **Element**: stores data associated to each element
- **KMatrix**: stores data associated to each stiffness matrix. Geometrically similar elements will be related to a single KMatrix object.

The following acronyms will be used in the definitions of the most important properties of these objects shown in Tables 2 to 4.

nnpe Number of nodes per element
nsides Number of sides of each element
nmesh Number of meshes in the *h*-adaptive sequence
ndofpe Number of degrees of freedom per element: *ndofpe* = *nnpe* x 2 (2-D)
ngauss Number of integration Gauss points used into each element
nlevels Number element levels.

Table 2. Node object.

Node(*n*)

Property	Type	Size	Information
.XYZ	Double	2 × 1	Geometrical coordinates of node *n* (2D)
.NParents	Int	1 × 1	Total number of *parent-nodes* for node *n*
.Parents	Int	1 × *NParents*	Numbers of the *parent-nodes* corresponding to node *n*
.Weights	Double	1 × *NParents*	Value of the shape functions of *parent-nodes* evaluated at *n* (*child-node*)
.HangingNode	Boolean	1 × *nmesh*	Flag used to indicate if node *n* is/is-not a *hanging node* in each of the meshes in the mesh sequence
.Boundary	Int	1 × 1	Code of geometrical entity associated to *n* (zero if *n* is in the interior of the domain)
.Subdom	Int	1 × *nlevels*	Subdomain in which the node is located at each level

Table 3. Element object.

Element(*e*)

Property	Type	Size	Information
.Top	Int	1 × *nnpe*	Element *e* topology (node numbers in *e*)
.Level	Int	1 × 1	Refinement level for element *e*
.NeighbElems	Int	1 × *nsides*	Number of neighbor element at each side of *e*, with the same *refinement level*.
.NeighbSides	Int	1 × *nsides*	Number of neighbor element side at each side of *e*.
.Active	Boolean	1 × *nmesh*	Flag used to indicate that *e* is/is-not active in each mesh of the *h*-adaptive sequence
.Children	Int	1 × 4	C*hildren*-elements numbers
.Parent	Int	1 × 1	*Parent*-element number
.KNum	Int	1 × 1	*KMatrix* object number storing data associated to *e*
.Heritage	Boolean	1 × 1	Flag used to indicate that *e* is Type A or Type B element.

Table 4. KMatrix Object.

KMatrix(*m*)

Property	Type	Size	Information
.ke	Double	*ndofpe* × *ndofpe*	Element stiffness matrix **k**
.BGPt	Double	3 × *ndofpe* × *ngauss*	**B** matrix evaluated at each Gauss integration point
.DetJGPt	Double	1 × *ngauss*	Determinant of the Jacobian matrix |**J**| evaluated at each Gauss integration point
.OrigLevel	Int	1 × 1	Refinement level corresponding to the element used to create *m*

11. REFERENCES

[1] J. Thomson, Z. Warsi, C. W. Mastin, Numerical grid generation: Foundations and applications, Elsevier, Amsterdam 1, 1985, 985.

[2] M. S. Shephard, An algorithm for defining a single near-optimum mesh for multiple-load-case problems, Int J Numer Methods Eng 15, 1980,pp. 617-625.

[3] I. Babuška and W. Rheinboldt, Adaptive approaches and reliability estimations in finite element analysis, Comput. Methods Appl. Mech. Eng. 17, 1979, pp. 519-540.

[4] J. F. Thompson, B. K. Soni, N. P. Weatherill, Handbook of Grid Generation, CRC, 1999.

[5] H. Jin and N. E. Wiberg, Two-dimensional mesh generation, adaptive remeshing and refinement, Int J Numer Methods Eng 29, 1990, pp. 1501-1526.

[6] J. Z. Zhu, E. Hinton, O. C. Zienkiewicz, Mesh enrichment against mesh regeneration using quadrilateral elements, Communications in Numerical Methods in Engineering 9, 1993, pp. 547-554.

[7] S. Zhang, On the nested refinement of quadrilateral and hexahedral finite elements and the affine approximation, Numerische Mathematik 98, 2004, pp. 559-579.

[8] M. A. Yerry and M. S. Shephard, Automatic three-dimensional mesh generation by the modified-octree technique, Int J Numer Methods Eng 20, 1984, pp. 1965-1990.

[9] M. C. Rivara, A grid generator based on 4-triangles conforming mesh-refinement algorithms, Int J Numer Methods Eng 24, 1987, pp. 1343-1354.

[10] M. T. Jones and P. E. Plassmann, Adaptive refinement of unstructured finite-element meshes, Finite Elements Anal. Des. 25, 1997, pp. 41-60.

[11] A. Tabarraei and N. Sukumar, Adaptive computations on conforming quadtree meshes, Finite Elements Anal. Des. 41, 2005, pp. 686-702.

[12] Á Plaza, J. P. Suárez, M. A. Padrón, S. Falcón, D. Amieiro, Mesh quality improvement and other properties in the four-triangles longest-edge partition, Comput. Aided Geom. Des. 21, 2004, pp. 353-369.

[13] A. Plaza, M. A. Padrón, J. P. Suárez, Non-degeneracy study of the 8-tetrahedra longest-edge partition, Applied Numerical Mathematics 55, 2005, pp. 458-472.

[14] C. James, D. A. Cavendish, H. William, An approach to automatic three-dimensional finite element mesh generation, Int J Numer Methods Eng 21, 1985, pp. 329-347.

[15] M. C. Rivara, Algorithms for refining triangular grids suitable for adaptive and multigrid techniques, Int J Numer Methods Eng 20, 1984, pp. 745-756.

[16] J. F. Abel and M. S. Shephard, An algorithm for multipoint constraints in finite element analysis, Int J Numer Methods Eng 14, 1979, pp. 464-467.

[17] C. Farhat, C. Lacour, D. Rixen, Incorporation of linear multipoint constraints in substructure based iterative solvers. part 1: A numerically scalable algorithm, Int J Numer Methods Eng 43, 1998, pp. 997-1016.

[18] J. J. Ródenas, M. Tur, J. E. Tarancón, F. J. Fuenmayor, H-adaptatividad de elementos finitos en refinamiento por subdivisión de malla. In XIV Congreso Nacional de Ingeniería Mecánica. Anales de Ingeniería Mecánica, Asociación Española de Ingeniería Mecánica, 2000.

[19] M. Tur, J. Fuenmayor, J. J. Ródenas, E. Giner, 3D analysis of the influence of specimen dimensions on fretting stresses, Finite Elements Anal. Des. 39, 2003, pp. 933-949.

[20] A. Suzuki and M. Tabata, Finite element matrices in congruent subdomains and their effective use for large-scale computations, Int J Numer Methods Eng 62, 2005, pp. 1807-1831.

[21] J. J. Ródenas, J. E. Tarancón, J. Albelda, A. Roda, F. J. Fuenmayor, Hierarquical properties in elements obtained by subdivision: A hierarquical h-adaptivity program. In Adaptive Modeling and Simulation 2005, P. Díez, N.E. Wiberg (Eds.), CIMNE, 2005.

[22] Matlab®,7.9.0.529 (R2009b)The MathWorks Inc, Natick, MA, 2009

[23] O. C. Zienkiewicz and J. Z. Zhu, A simple error estimation and adaptive procedure for practical engineering analysis, Int. J. Numer. Methods Eng. 24 1987, pp. 337-357.

[24] J. J. Ródenas, Fuenmayor, F.J., A. Vercher, Improvement of the superconvergent patch recovery technique by the use of constraint equations: The SPR-C technique, Int. J. Numer. Methods Eng. 70, 2007, pp. 705-727.

[25] O. C. Zienkiewicz and J. Z. Zhu, The superconvergent patch recovery and a posteriori error estimates. part I: The recovery technique, Int. J. Numer. Methods Eng. 33, 1992, pp. 1331-1364.

[26] P. Ladeveze and D. Leguillon, Error estimate procedure in the finite element method and applications, SIAM Journal on Numerical Analysis 20, 1983, pp. 485-509.

[27] P. Ladeveze, P. Marin, J. P. Pelle, J. L. Gastine, Accuracy and optimal meshes in finite element computation for nearly incompressible materials, Comput. Methods Appl. Mech. Eng. 94, 1992, pp. 303-315.

[28] P. Coorevits, P. Ladeveze, J. P. Pelle, An automatic procedure with a control of accuracy for finite element analysis in 2D elasticity, Comput. Methods Appl. Mech. Eng. 121, 1995, pp. 91-120.

[29] E. Stein, M. Rüter, S. Ohnimus, Adaptive finite element analysis and modelling of solids and structures. findings, problems and trends, Int J Numer Methods Eng 60, 2004, pp. 103-138.

[30] M. Rüter and E. Stein, Goal-oriented a posteriori error estimates in linear elastic fracture mechanics, Comput. Methods Appl. Mech. Eng. 195, 2006, pp. 251-278.

[31] J. J. Ródenas, J. Albelda, C. Corral, J. Mas, "Efficient implementation of domain decomposition methods using a hierarchical h-adaptive finite element program". In III European Conference on Computational Mechanics. Solids, Structures and Coupled Problems in Engineering. Book of Abstracs, Springer, 2006.

[32] J. J. Ródenas, J. Albelda, C. Corral, J. Mas, "A domain decomposition iterative solver based on a hierarchical h-adaptive FE code". In Proceedings of the Fifth International Conference on Engineering Computational Technology, B.H.V. Topping, G. Montero, R. Montenegro (Eds.), Civil-comp Press, 2006.

[33] J. J. Ródenas, C. Corral, J. Albelda, J. Mas, C. Adam, "Solver directo de división recursiva en subdominios basado en un programa de refinamiento h-adaptable de estructura jerárquica". In Métodos Numéricos e Computacionais em Engenharia. CMNE CILAMCE 2007, J.César de Sá, Raimundo Delgado, Abel d. Santos, Antonio Rodríguez-Ferrán, Javier Oliver: Paulo R.M. Lyra, José L. D. Alves (Eds.), APMTAC, SEMMNI, ABMEC, 2007.

[34] J. J. Ródenas, C. Corral, J. Albelda, J. Mas, C. Adam, "Nested domain decomposition direct and iterative solvers based on a hierarchical h-adaptive finite element code. In ADMOS 2007". IV International Conference on Adaptive Modeling and Simulation, K. Runesson, P. Díez (Eds.), Internacional Center for Numerical Methods in Engineering (CIMNE), 2007.