Efficient unstructured quadrilateral mesh generation

Josep Sarrate and Antonio Huerta*,†

Department de Matemàtica Aplicada III, E.T.S. de Ingenieros de Caminos, Canales y Puertos, Universitat Politècnica de Catalunya, Campus Nord, E-08034 Barcelona, Spain

SUMMARY

This work is devoted to the description of an algorithm for automatic quadrilateral mesh generation. The technique is based on a recursive decomposition of the domain into quadrilateral elements. This automatically generates meshes composed entirely by quadrilaterals over complex geometries (there is no need for a previous step where triangles are generated). A background mesh with the desired element sizes allows to obtain the preferred sizes anywhere in the domain. The final mesh can be viewed as the optimal one given the objective function is defined. The recursive algorithm induces an efficient data structure which optimizes the computer cost. Several examples are presented to show the efficiency of this algorithm.

KEY WORDS: mesh generation; mesh reasoning; unstructured; quadrilateral; finite elements

1. INTRODUCTION

In the past decades, computational methods aiming an accurate approximation to the solution of partial differential equations have received considerable attention. In particular, the finite element method has steadily increased its range of applicability and its computational efficiency. However, it is hampered by the need to generate a discretization adapted to a general geometry and to the desired distribution of element sizes. In fact, the preparation of accurate data is usually, in real applications, the largest portion of the overall cost of the analysis. Consequently, a large number of automatic mesh generation techniques have been devised to overcome these problems. Most of these efforts, and also most of the important achievements, concern unstructured triangular mesh generators [1–4], while the inherent difficulties associated to unstructured quadrilateral mesh generators have precluded, until very recently, general efficient methodologies [5–12]. Nevertheless, the use of mixed formulations in incompressible fluid and solid mechanics where quadrilateral elements are preferred by several authors, have increased the general interest in unstructured quadrilateral discretizations.

*Correspondence to: Antonio Huerta, Departament de Matemàtica Aplicada III, E.T.S. de Ingenieros de Caminos, Canales y Puertosa, Universitat Politècnica de Catalunya, Campus Nord, E-08034 Barcelona, Spain
†E-mail: antonio.huerta@upc.es

Contract/grant sponsor: Ministerio de Educación y Cultura; contract/grant number: TAP98-0421

Efficient automated meshing techniques are expected to have certain features in order to ensure its applicability in a wide scope of cases, which can range from regular domains with uniform element sizes to non-singly connected domains with large boundary curvatures and non-uniform element sizes. Haber *et al.* [13] present an excellent discussion of such features: *precise modelling of the boundaries*, *good correlation between the interior mesh and the information prescribed at the boundary*, *minimal input effort*, *broad range of applicability*, *general topology*, *automatic topology generation*, *and favorable element shapes*. Some of these features can be easily implemented; for instance, Bézier or B-splines interpolation curves allow a *precise modelling of the boundaries*. Others, such as *minimal input effort* and *broad range of applicability* are much more difficult to obtain. Therefore, all the developed techniques for mesh generation should include most of the previous features and this is the goal of the proposed algorithm.

However, from a practical point of view, the basic task of a mesher is to generate the nodal co-ordinates and the element connectivity. Ho-Le [14] classifies the mesh generation techniques according to the sequence used by the algorithm: topology first and then nodal co-ordinates, nodal co-ordinates previous to the element connectivity or both at the same time.

The unstructured quadrilateral mesh generator presented here is one of the latter. It is based in the original algorithm proposed by Sluiter and Hansen [5] and further developed by Talbert and Parkinson [7]. It automatically generates meshes composed by quadrilaterals over complex geometries. The algorithm proposed here differs from Reference [7] in several crucial points. Now the user-defined element size can be specified anywhere in the domain. This point is crucial in adaptive techniques based on error estimation [15, 16], where the mesh size distribution is decided by the estimated error everywhere in the domain. Moreover, the technique proposed here is organized, for computational efficiently, in three phases. The first one, *determination of the best splitting line* is based in an improved optimality function. The second one *node placement* is the one associated with the desired element size. The final step, which is usual in mesh generators, is charged of *mesh quality enhancement*.

Previous to the domain discretization, the boundary nodes are defined using any interpolation technique. Then, the mesh generation process starts: the initial domain is partitioned by a splitting line that connects two boundary nodes. The splitting line is chosen minimizing an objective function. Second, the nodes are positioned along the splitting line with the desired spacing. Two new subdomains are now considered and the splitting process is repeated recursively up to the desired element size. Finally, a continuous reasoning method is developed to obtain a non-distorted mesh.

The outline of the remainder of the paper is as follows. In Section 2 a brief classification of the mesh generation algorithms is presented. Then, basic concepts on background meshes and boundary processing information are reviewed. In Section 3 the two basic phases of the new mesh generator algorithm are developed: *determination of the best splitting line* and *node placement*. In Section 4 the mesh quality enhancement procedures are presented. Section 5 is devoted to the analysis of the computational cost of the developed algorithm. Finally, in Section 6, we present and discuss several examples.

2. BASIC CONSIDERATIONS/INPUT DATA

2.1. *Classification of mesh generation methods*

Several authors [11, 17] have presented classifications of mesh generation methods. Apart from the manual or semi-automatic generation techniques which are left for simple cases or academic studies,

three main categories are devised. First, the methods based on *adaptation of mesh templates* are defined. The simplest of such techniques is the Grid-based approach which consists of superposition of a grid template over the domain, the cells that fall outside the domain are discarded and those which intersect the contour of the object are readapted to fit into the domain of analysis [2, 18]. This induces a regular discretization in the interior of the domain but an extremely poor one along the boundaries, and it is not always possible to keep the offset elements as quadrilaterals [2]. Instead of a regular template a grid based on a quadtree construction (in 2D and octree in 3D) can be used, it is the Quadtree–octree method [17]. These meshes which are usually of good quality far away from the boundaries are generally very poor in their vicinity.

Second, the *generalized mappings* methods are based, for general objects, in a previous subdivision of the general domain into simpler regions; then, adequate mappings are used to discretise each subdomain. The Transport-mapping methods, discussed in detail in Haber *et al.* [13], are probably the most popular techniques of this category. In this case, the general object is first subdivided in simpler domains with usually three or four sides and then the mappings referred to predefined templates. One of the first of such techniques was itself based on the finite element interpolation functions, the *isoparametric mapping method* [19]. Then *transfinite mappings* or *discrete transfinite mappings* were developed [13, 20] to generalize the interpolations, to better describe the boundaries (curves and surfaces are described exactly), and to prescribe specific constraint curves where the mesh lines must pass. Although, these latter developments allow for subdomains having more than four sides and even with non-simply connected regions, the initial partition of the original object reduces the advantages of these methods. This drawback is also present in the Conformal mapping methods [21] which could be viewed as a generalization of the original transport-mapping methods. In fact, they can deal with simply connected regions with more than four sides. However, they suffer from the fact that element shape and mesh density are difficult to control.

Other generalized mapping methods are the Partial differential equations methods [22–24]. Due to their importance and extended use, they usually are in a category by themselves. However, they also rely on a mapping, although in this case the mapping is not defined explicitly but it is computed by solving a predetermined system of partial differential equations. The *Laplacian mesh generation* is probably the better known technique in this case. In fact, these techniques are widely used in finite differences because they induce structured meshes. However, one of the main advantages of finite elements is the possibility of employing unstructured meshes.

Finally, the *geometric decomposition* methods have become the most representative and used. Delaunay triangulations, [4, 25–28] among others, are, due to their mathematical basis, robust and efficient in two and three-dimensional simplex generation. However, they cannot be generalized for quadrilaterals or hexahedral elements. Moreover, stretching of elements is non-trivial. This is not the case of Advancing Front Methods [3, 9, 10, 29–31]. These methods are among the most widely used. Finally, methods based on Recursive decomposition of the domain have been used for triangular elements [11] and quadrilaterals [7].

The technique developed here is based on the later family of methods.

2.2. *Background mesh*

The user-defined element size can be specified either (i) at the boundary [7], the node spacing and therefore the element size inside the domain are determined by the boundary values through interpolation, or (ii) at the nodes of a background mesh [3, 10], the element size is determined

anywhere in the domain through linear interpolation inside the corresponding triangle. Other possibilities do exist, see for instance, Reference [31]. While the former is much more efficient from a computational point of view (extensive search algorithms are avoided), the latter allows an accurate distribution of the desired mesh size. As usual, the requirements for the background mesh are very relaxed. It must cover the domain completely to be discretized but most of them do not describe the geometry accurately. The quadrilateral mesh generator proposed here can work with either approach to prescribe the user-defined element size.

2.3. *Boundary processing/information*

In order to minimize the input effort, the boundary of the domain is defined by some base points which define any user-preferred interpolation technique. Usually, Bézier or B-splines is employed. Once the parametrization of the contour is defined, the nodes along the boundary are generated according to the prescribed element density. Note that the total number of nodes must be even to ensure that a quadrilateral tessellation is possible (this requirement also applies to any subdomain boundary that will be generated during the splitting process). Multiconnected domains are easily transformed in singly connected ones by making the necessary cuts. These cuts, or return segments, consist of two superposed contour segments along which the generated nodes coincide. When the meshing process is completed, these return segments are erased and the twin nodes which might have appeared are disregarded. This procedure is standard, see, for instance Reference [10].

3. THE QUADRILATERAL MESH GENERATOR

3.1. *Determination of the best splitting line*

As previously mentioned, the mesh generator is based on a recursive decomposition of the domain, until only quadrilaterals are left. After the initial boundary is processed, it is transformed into a simple closed polygonal line. The possible splitting lines must have their endpoints over two non-consecutive nodes of this polygon. Once the optimal line has been chosen, its corresponding new nodes are generated (in the *node placement* phase); this leads to two new singly closed polygonal lines, where the recursivity of the algorithm is exploited.

The splitting line cannot be arbitrary, moreover, the optimal one should be chosen. In order to find the best splitting line an objective function (cost function) is defined. In fact, the search for the optimal line is a discrete minimization problem with constrains because all the lines are not acceptable.

The total number of possible lines between two non-consecutive vertexes of an n-vertex polygon is $(n^2-3n-2)/2$; all these lines are not acceptable. First, lines joining two aligned vertexes along one contour segment must be disregarded. Second, lines that may, entirely or in part, run outside the boundary in non-convex domains must also be eliminated, see Figure 1. In Reference 7 an algorithm for node visibility is proposed, but others are also possible.

The definition of objective function is a basic point to ensure proper results. It is evaluated a large number of times. Thus, the objective function must be simple to evaluate and include the necessary criteria for mesh optimality. It is defined as a linear combination of five indicators that quantify geometrical criteria, namely

$$\text{Objective Function} = c_1\phi + c_2\sigma + c_3\varepsilon + c_4\ell + c_5\alpha \tag{1}$$

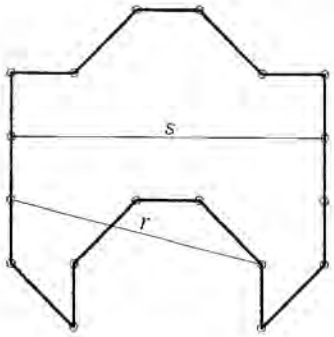


Figure 1. Line r is not acceptable because it runs outside the domain, line s is acceptable.

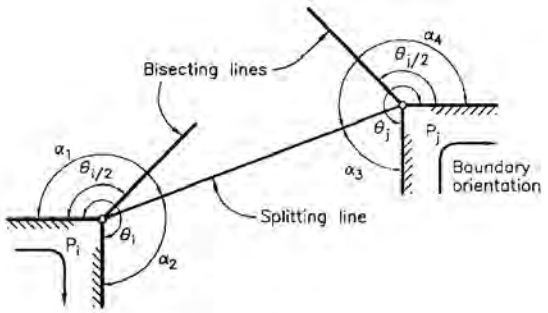


Figure 2. Angles formed by a candidate splitting line.

where c_i, $i=1,\ldots,5$, are used as weighting values. Equation (1) is evaluated for each acceptable splitting line. The best one, which will decompose the domain into two polygons, is the one that minimizes (1). The geometrical criteria are presented next.

1. *Splitting angles.* This criterion favours splitting lines that coincide with bisecting lines. Each candidate splitting line between nodes P_i and P_j forms four angles, as it is shown in Figure 2. The basic goal is to choose a splitting line as close to the bisecting line as possible and to penalize splitting lines bisecting contours with angles less than $\pi/2$. To this end, the splitting angle criterion is defined as

$$\phi = \begin{cases} 1 & \text{if } (\alpha_1+\alpha_2<\pi/2) \text{ and } (\alpha_3+\alpha_4<\pi/2) \\ \mu(\zeta_1,\zeta_2) & \text{if}(\pi/2\leqslant\alpha_1+\alpha_2\leqslant 2\pi/3) \text{ or} \\ & (\pi/2\leqslant\alpha_3+\alpha_4\leqslant 2\pi/3) \\ \psi(\alpha_1+\alpha_2,\alpha_3+\alpha_4) & \text{if } (\alpha_1+\alpha_2>2\pi/3) \text{ and } (\phi_2>2\pi/3) \end{cases} \tag{2}$$

where α_1, α_2, α_3 and α_4 are the angles between the splitting line and the subdomain boundary (see Figure 2), $\psi(\alpha_1+\alpha_2,\alpha_3+\alpha_4)$ is a function that tends to select splitting lines close to the bisecting line:

$$\psi(\alpha_1+\alpha_2,\alpha_3+\alpha_4)=\frac{|\alpha_1-\alpha_2|+|\alpha_3-\alpha_4|}{(\alpha_1+\alpha_2)+(\alpha_3+\alpha_4)} \tag{3}$$

and $\mu(\zeta_1,\zeta_2)$ is a function that allows a smooth transition between the extreme values. It is defined as

$$\mu(\zeta_1,\zeta_2)=(1-\zeta_1\zeta_2)+\zeta_1\zeta_2\psi(\alpha_1+\alpha_2,\alpha_3+\alpha_4) \tag{4}$$

where

$$\zeta_1=\begin{cases} \dfrac{(\alpha_1+\alpha_2)-2\pi/3}{\pi/6} & \text{if } \pi/2\leqslant\alpha_1+\alpha_2\leqslant 2\pi/3 \\ 1 & \text{otherwise} \end{cases}$$

Table I. Local structuring index values.

Inner nodes	
Common Elements	σ_i
3	5.6
4	4.0
5	36.0
6	60.0
Any Other	80.0
Nodes on initial boundary	
θ_i	σ_i
$0 \leqslant \theta_i < 2\pi/3$	100.0
$2\pi/3 \leqslant \theta_i \leqslant 2\pi$	0.0
Nodes on subdomain boundary	
θ_i	σ_i
$0 \leqslant \theta_i < 2\pi/3$	80.0
$2\pi/3 \leqslant \theta_i \leqslant 2\pi$	0.0

and

$$\zeta_2 = \begin{cases} \dfrac{(\alpha_3 + \alpha_4) - 2\pi/3}{\pi/6} & \text{if } \pi/2 \leqslant \alpha_3 + \alpha_4 \leqslant 2\pi/3 \\ 1 & \text{otherwise} \end{cases}$$

It should be noted that $0 \leqslant \phi \leqslant 1$. Moreover, for rectangular subdomains with a prescribed constant element size, ϕ will be zero when both bisecting lines coincide with the splitting line.

2. *Structuring index.* This criterion tries to construct structured meshes if possible. The algorithm considers every created subdomain independent of its neighbours. Nevertheless, the goal of the algorithm is to obtain elements with a distortion that is as small as possible. Therefore, it seems reasonable to have a measure of the desirable number of elements to which every node belongs. This is called *local structuring*. This may be evaluated by assigning a score that measures the *structure* at each endpoint of the candidate splitting line. Then, the estimator of the local structuring for a splitting line between points P_i and P_j is found by averaging the scores at each endpoint, leading to

$$\sigma = \frac{\sigma_i + \sigma_j}{200} \tag{5}$$

For inner nodes in a quadrilateral mesh, it seems reasonable that the optimal number of common elements should be four. Then, the mesh will be as structured as possible and the four formed angles will tend to $\pi/2$. When this number is four at one node i the mesh is called *locally structured at node i*. Table I shows the assigned scores for inner nodes. Note that for locally structured nodes a minimum score is assigned.

The score assigned to a node i on the boundary depends on the angle θ_i defined by the adjacent segments $\overline{\mathbf{x}_{i-1}\mathbf{x}_i}$ and $\overline{\mathbf{x}_i\mathbf{x}_{i+1}}$, where $\mathbf{x}_{i-1}$, $\mathbf{x}_i$, $\mathbf{x}_{i+1}$ are the co-ordinates of nodes $i-1$, i and $i+1$, respectively. Two cases are considered: (i) initial boundary and (ii) subdomain boundary. As shown in Table I, the main idea is to penalize a splitting line departing from (or reaching) a node

on the subdomain boundary with $\theta_i \leqslant 2\pi/3$. It should be noted that slightly bigger penalization is assigned to nodes on the original boundary that meet this criterion, see Table I. This is because mesh quality enhancement techniques will not be able to improve either their position nor their connectivity.

3. *Node placement error*. This criterion allows to choose the best splitting line in limit cases. On every splitting line, new nodes are generated according to an interpolation between its end nodes or values defined on a background mesh. However, it is not always possible to find an integer number of new nodes which fits the prescribed density, and the actual chosen position will introduce an error that should be quantified. To this end, we first compute the 'theoretical' number of element sides to be generated along the splitting line, n_e^*. Then, we approximate it by a positive integer number (at least one element per splitting line), n. This integer number must also meet the parity requirement in order to generate two subdomains with an even number of element sides. Then, the node placement parameter is defined as

$$\varepsilon = |n_e^* - n| \tag{6}$$

where $0 \leqslant \varepsilon \leqslant 1$. Note that for practical purposes this difference is only significant when splitting lines partitioned in few element sides are considered. Therefore, parameter (6) is only taken into account when subdomains defined by six nodes, i.e. small subdomains, are treated and, is not considered for a general subdomain (its weight is set to zero in this case).

4. *Splitting line length*. A splitting line must join two points which are as close as possible. The reason is that information on mesh density may sit on its endpoints. A line which is too long may have some of its intermediate points near a zone with a very different assigned density, leading to a great inconsistency of the results. This is the case shown in Figure 3. A nondimensional estimation ℓ of the splitting line length between two nodes P_i and P_j can be obtained by dividing its natural length by the domain characteristic length defined as, see Figure 4,

$$l_{\text{char}} = \sqrt{(x_{\max} - x_{\min})^2 + (y_{\max} - y_{\min})^2} \tag{7}$$

Hence

$$\ell = \frac{l_{ij}}{l_{\text{char}}} \tag{8}$$

and, as usual, $0 \leqslant \ell \leqslant 1$.

5. *Symmetry*. When the domain presents clear symmetries, the final mesh should maintain these symmetries. Unfortunately, unstructured mesh generators do not usually create symmetric grids. An estimation of the symmetry is used in the developed algorithm. It uses the difference between the areas of the subdomains defined by every splitting line. Obviously, this is a very simple measure of the domain symmetry and it has no effect on non-symmetrical domains. However, it has proved to be really useful for simple cases or symmetrical domains (especially those with parallel sides). The non-dimensional estimator that has been used is

$$\alpha = \frac{|a_2 - a_1|}{a_2 + a_1}, \tag{9}$$

where a_1 and a_2 are the respective areas closed by each subdomain defined by the candidate splitting line. In fact, parameter (9) measures the areas in both sides of a splitting line. However, a line which divides the domain into two parts with both areas as similar as possible, seems to

Figure 3. Influence of the splitting line length in the results. In the case on right, the centre of the domain will not be meshed with the proper density.

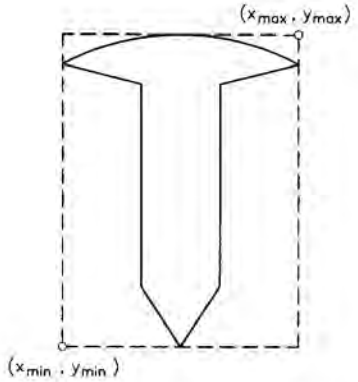


Figure 4. Definition of the characteristic length of a given domain.

be the best option, even in non-symmetrical domains. Note that some flags can be added in order to check that domains with the same area are symmetrical. However, in order to increase the computational efficiency of the algorithm they are not added.

Based on numerical experiments over a wide range of problems, the best set of weighting values are: $c_1 = 0.52$, $c_2 = 0.17$, $c_3 = 0.00$, $c_4 = 0.17$, $c_5 = 0.14$.

Note that the cost function definition (1) is a generalization the one developed by Talbert and Parkinson, [7]. However, in (1) two new criteria are added (structuring index and symmetry criteria) and the remaining criteria have a different definition (splitting angle, node placement error and splitting line length). For instance, in the new splitting angle criterion, the splitting line tends toward the bisecting line instead to the normal line. An absolute error instead of a relative error is used to measure the node placement. Moreover, in each subdomain a characteristic length, instead of the maximum length, is used to normalize (7) in the present algorithm to speed up the meshing process. Finally, it must be remarked that the weighting coefficients are normalized.

3.2. Six-node closure

The recursive splitting algorithm previously introduced finishes when a domain containing only four nodes (an element) is found. However, a six-node subdomain appears in a variety of final configurations during the partition process. The manner in which these domains are transformed into quadrilaterals is important for the final mesh quality. Although Talbert and Parkinson [7] have used a template-based method to split up the six-node subdomains, in the present algorithm the same objective function, see Equation (1), has been used. Nevertheless, two modifications are introduced. First, the splitting angle criterion has been modified in order to enforce $\pi/2$ angles. Obviously, in most cases this will not be possible at all. The measure of the deviation of the

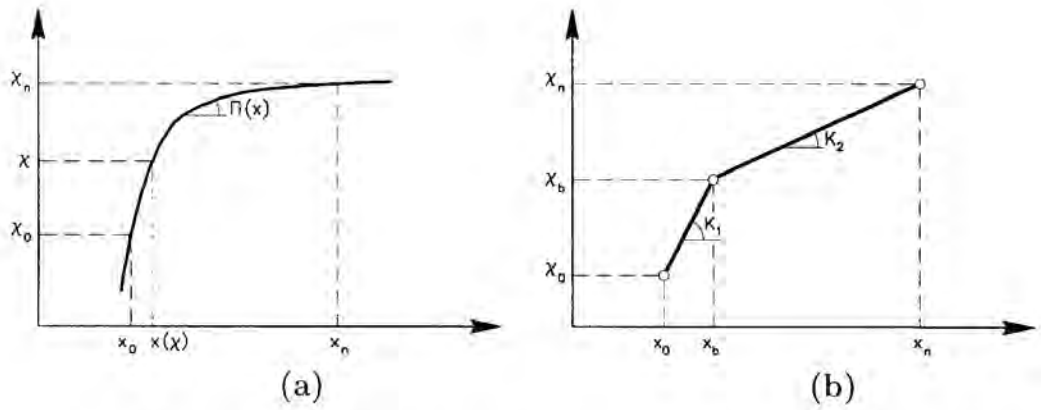


Figure 5. Relocation of χ_i points along the splitting line: (a) general case; (b) definition of the function $\Pi(x)$ as two constants.

actual angle from $\pi/2$ can be computed as

$$\phi = \frac{\sum_{k=1}^{4} |\phi_k| - \frac{\pi}{2}|}{2\pi} \tag{10}$$

Second, two loops, with different weighting values in the cost function are used to transform these hexagons into quadrilateral elements. In general, the size of this subdomains is small. Therefore, neither the splitting line length nor the symmetry criterion is very relevant. The weighting values for each loop are:

first loop $\quad c_1 = 0.56,\ c_2 = 0.33,\ c_3 = 0.11,\ c_4 = 0.00,\ c_5 = 0.00$

second loop $\quad c_1 = 0.60,\ c_2 = 0.25,\ c_3 = 0.15,\ c_4 = 0.00,\ c_5 = 0.00$

These values have been deduced empirically in order to obtain high-quality meshes.

3.3. Node placement

The developed node placement algorithm is based on a fixed reference system over which the nodes are previously generated following an arbitrary distribution (for instance, equidistributed nodes). Then, through a transformation, nodes are relocated along the splitting line according to the prescribed element sizes.

Let $\chi \in [\chi_0, \chi_n]$ be a fixed reference and let λ_i with $i = 0, \ldots, n$, be the $n+1$ points distributed along the segment $[\chi_0, \chi_n]$. The goal of the present algorithm is to compute the new position of the points $x_i = x(\chi_i)$, $i = 0, \ldots, n$, along the segment $[x_0, x_n]$, where $x_0 \equiv x(\chi_0)$ and $x_n \equiv x(\chi_n)$, in such a way that the new co-ordinates x have a different distribution, see Figure 5(a).

The nodal position, $x(\chi)$, is defined implicitly as

$$\frac{\int_{x_0}^{x(\chi)} \Pi(\xi)\, \mathrm{d}\xi}{\int_{x_0}^{x_n} \Pi(\xi)\, \mathrm{d}\xi} = \frac{\chi - \chi_0}{\chi_n - \chi_0} \tag{11}$$

where $\Pi(x)$ is a function defined over $[x_0, x_n]$. Note that if $\Pi(x) = K$, K being an arbitrary constant, then

$$\frac{x(\chi) - x_0}{x_n - x_0} = \frac{\chi - \chi_0}{\chi_n - \chi_0} \quad \Longrightarrow \quad x(\chi) = \frac{x_n - x_0}{\chi_n - \chi_0}(\chi - \chi_0) + x_0$$

and nodes will be relocated proportionally to the fixed reference distribution. On the other hand, if $\Pi(x)$ is defined as

$$\Pi(x) = \begin{cases} K_1 & \text{if } \chi_0 \leqslant \chi \leqslant \chi_b \\ K_2 & \text{if } \chi_b < \chi \leqslant \chi_n \end{cases}$$

being $K_1 > K_2$, see Figure 5(b), then the distance between nodes in the interval $[x_0, x_b]$ will be smaller than the distance between nodes in the interval $[x_b, x_n]$. Therefore, high values of K tend to reduce the distance between nodes along the splitting line. In fact, $\Pi(x)$ can be understood as the inverse of the prescribed element side size, $h(x)$:

$$\Pi(x) = \frac{1}{h(x)} \tag{12}$$

Thus, the theoretical number of element sides along a splitting line is

$$n_e^* = \int_{x_0}^{x_n} \Pi(\xi)\, \mathrm{d}\xi = \int_{x_0}^{x_n} \frac{1}{h(\xi)}\, \mathrm{d}\xi \tag{13}$$

Finally, the position of node $x_k = x(\chi_k)$ can be computed from Equations (11) and (12) as

$$\int_{x_0}^{x_k} \frac{\mathrm{d}\xi}{h(\xi)} = \frac{k}{n} \int_{x_0}^{x_n} \frac{\mathrm{d}\xi}{h(\xi)}$$

Therefore, the new position of any node can be computed recurrently as

$$\int_{x_k}^{x_{k+1}} \frac{\mathrm{d}\xi}{h(\xi)} = \frac{k+1}{n} \int_{x_0}^{x_n} \frac{\mathrm{d}\xi}{h(\xi)} - \frac{k}{n} \int_{x_0}^{x_n} \frac{\mathrm{d}\xi}{h(\xi)} = \frac{n_e^*}{n} \tag{14}$$

where n is the real number of element sides to be generated along the splitting line. Notice that n is not arbitrary. First, it must be an integer number. Second, it must verify the parity condition introduced in Section 2. In the appendix, Equation (14) is developed and an explicit and analytic expression to recurrently compute the node position is obtained for two-dimensional problems.

Figure 6 shows the splitting line election and the node placement phases. The domain was proposed in Reference [8] and it consists of two squares centred at the same point, with sides in a ratio of 0.4 and rotated 90°. Since it is a multiconnected domain a return segment is used in order to joint the outer and the inner boundaries. In the first column, the mesh generation process is presented when the element size distribution is prescribed at the vertexes of the two rotated squares. Figure 6(a) shows the initial boundaries with nodes generated on them. Figures 6(b) and 6(c) show two intermediate states in the splitting process. Figure 6(d) shows the generated grid composed by quadrilateral elements. As it can be seen, very distorted elements appear. This problem will be overcome during the mesh quality enhancement phase. Note that a symmetric mesh is obtained over a symmetric domain with symmetric element size distribution. In the second column, the same process is detailed when the background mesh is used. Note that a non-symmetric mesh is obtained in this case due to the presence of a return segment.

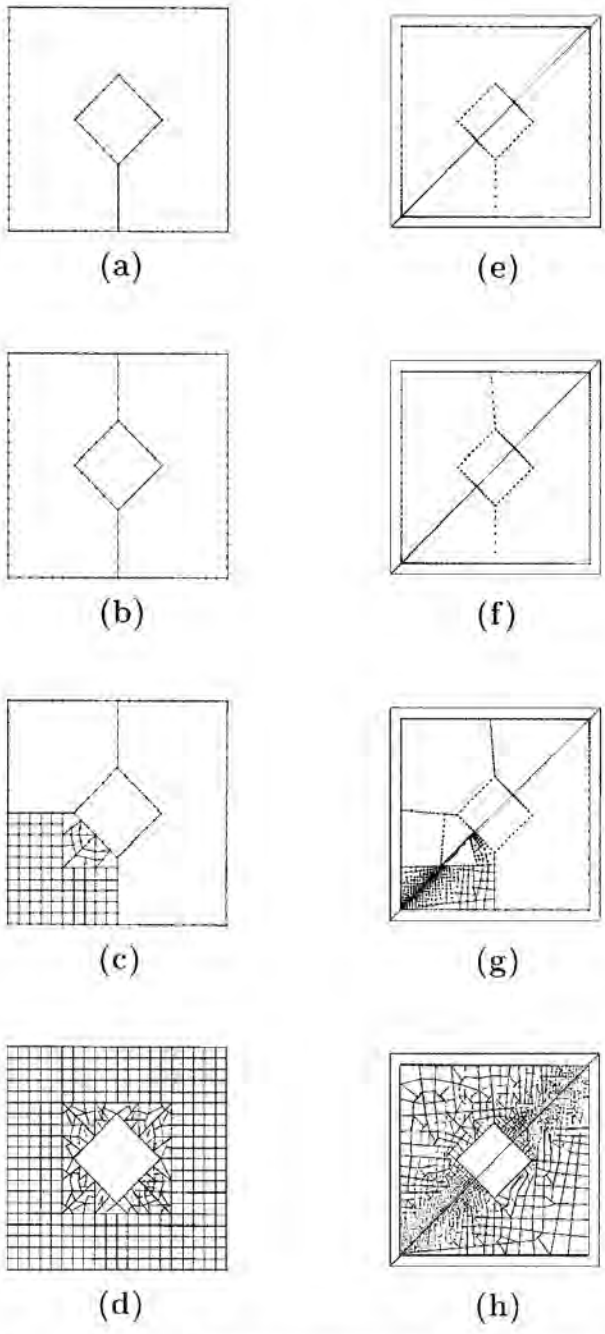


Figure 6. Several steps in the mesh generation process for the two rotated squares problem.

4. MESH QUALITY ENHANCEMENT

4.1. Introduction

The recursive splitting algorithm described in the previous section may generate elements that are very distorted. Therefore, mesh quality enhancement procedure has to be developed in order to improve the overall mesh quality. As it is usual in quadrilateral mesh generation algorithms [7–10, 12], two types of procedures are considered. The first one, often called *make-up* techniques, is focused in the improvement of the mesh topology. The second one, called *mesh smoothing*, improves the shape of the elements by modifying the position of the inner nodes once the topology is fixed.

4.2. Make-up techniques

After the split process is completed, some topological properties are improved. Since one of the goals of the present discretization algorithm is to generate meshes as structured as possible, it seems reasonable to favour four elements per interior node (`NE` = 4, `NE` being the number of elements per node). However, for unstructured meshes some nodes with `NE` bigger or smaller than four will appear. Nevertheless, it is better to enforce `NE` close to four. Note that this condition also precludes very distorted elements. In fact, the number of elements that meet one node has already been taken into account in the selection of the best splitting line (structuring index criterion). Therefore, only two techniques of those developed in others mesh generators have been introduced in our algorithm: node elimination and element elimination (see References [9, 10, 12] for details). It should be noted that any *make-up* techniques is used if a node on the boundary is concerned.

4.3. Smoothing algorithm

Since *make-up* techniques only modify the mesh topology, after they are applied very distorted elements still appear. Then, it is necessary to apply a smoothing technique to improve the mesh quality. The most commonly used technique is the so-called Laplacian method [32], which computes the new nodal position solving the Laplace equation. This technique has an important drawback because it is possible that, in non-convex domains, nodes run outside it. Techniques to preclude such a pitfall either increase the computational cost enormously or introduce new terms in the formulation that are particular for each geometry. Giuliani [33] developed a new reasoning algorithm based on geometrical criteria. This method modifies the position of every node in order to minimize a geometric-oriented average distortion of elements meeting on it. These modifications are done with an explicit iterative procedure. In this case, nodes cannot depart from the domain because this is an unstable position in terms of distortion and squeeze.

h-adaptive techniques [15, 16] first compute a solution on a given coarse mesh. Then, a new element size distribution is computed from a local measure of the estimated error. Therefore, it is crucial in these processes that the mesh generator preserves the prescribed element size. In this sense, it is essential that the smoothing algorithm also maintains the size. Giuliani method gives proven results in a smooth element size distributions for both 2D and 3D problems. However, it yields unsatisfactory meshes when sharp changes of density appear. This is due to the fact that zones with high density tend to lose it after several remeshing iterations at advanced stages of the analysis.

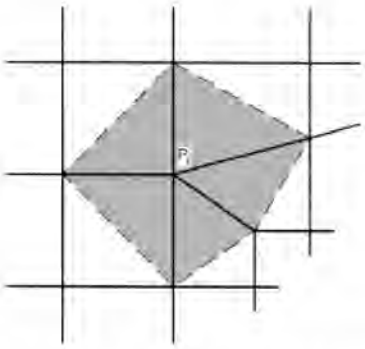
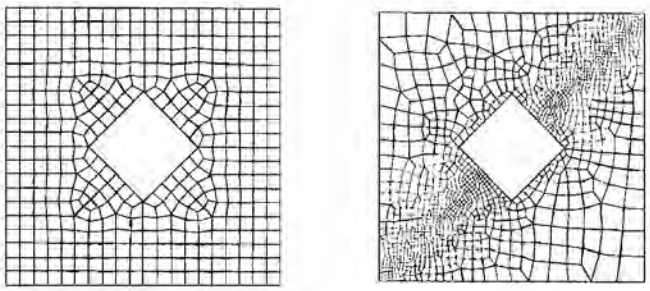


Figure 7. Representation of the set of triangles (shadowed) around node P_i.



(a) (b)

Figure 8. Final mesh for the two rotated squares example: (a) without background mesh; (b) with background mesh.

The cause of this problem may be found at its basic reasoning principle, see Reference [33] for details. For each node P_i in a quadrilateral grid, a local measure of the mesh distortion is defined in terms of a set of triangles. These triangles are obtained by joining all nodes connected to node P_i via the element sides, see Figure 7. The new position of P_i is found by minimizing the sum of their distortions. This is iteratively repeated for all the nodes in a Gauss–Seidel like procedure, until convergence is achieved. In the original method, the distortion of the triangles is defined in order to obtain triangles of similar sizes. Thus, the final mesh will show smooth variations of the element size, and zones where high element density is prescribed will tend to lose it. In order to overcome this problem a modification of the distortion of the triangles is introduced, see Reference [34] for details. It tends to preserve the original size of the triangles, and therefore, the final mesh will maintain the prescribed element size. Examples presented in Reference [34] prove that this modification produces a robust algorithm that generates well-shaped elements of the prescribed size.

Figure 8(a) shows the final mesh of the two rotated squares problem after the mesh quality enhancement phase has been applied, for the uniform element size distribution case. Note that a symmetric mesh is obtained and that the prescribed element size is preserved. Figure 8(b) shows

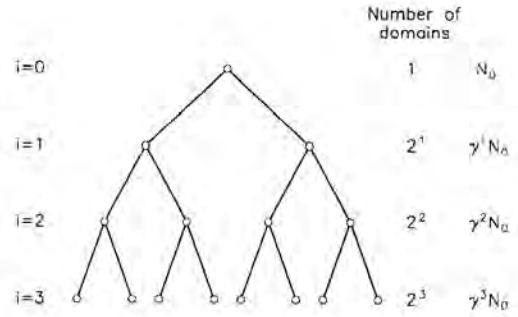


Figure 9. Bin-tree representation of the proposed algorithm.

the final mesh for a non-uniform element size distribution imposed with a background grid. A smooth variation in the element size is obtained notwithstanding the remarkable element size gradient. In both cases very few distorted elements are generated.

5. ALGORITHM EFFICIENCY

This section is devoted to the analysis of the computational cost of the developed meshing algorithm. In this analysis it is assumed that a uniform nodal density is prescribed. However, numerical examples show that the derived estimation of the computational cost can also be used with non-uniform meshes. The computational cost is defined as the number of objective function evaluation, (1). It is important to note that the described algorithm is naturally suited for a bin-tree structure of the element data. A recursive binary structure may be constructed since every domain is split into a 'left' and 'right' subdomains, see Figure 9. Hence, the total cost is the number of evaluations of the objective function in every level of the bin-tree representation. Note that for each level, the objective function has to be evaluated in several subdomains in order to find the best splitting line for every subdomain, see Figure 9.

To this end, we first compute the cost involved in obtaining the best splitting line for a subdomain with N_i nodes on its boundary. If it is assumed that (i) the cost of each evaluation of the objective function is always constant, and that (ii) the cost of the visibility algorithm is at least equal to the cost of the lines whose objective function is not evaluated. Therefore, the objective function is evaluated

$$(N_i - 1) + (N_i - 2) + \cdots + 1 = \tfrac{1}{2} N_i (N_i - 1) \tag{15}$$

times in each subdomain with N_i nodes on its boundary.

Second, we evaluate an upper bound of the number of nodes in each subdomain of the ith partition. Given a certain level, $i-1$, in the partition process, let N_{i-1} be the number of nodes on its boundary. We assume that the best splitting line generates two new subdomains with $\gamma_i^{\mathrm{R}} N_{i-1}$ and $\gamma_i^{\mathrm{L}} N_{i-1}$ nodes each ($0 < \gamma_i^{\mathrm{R}} < 1$ and $0 < \gamma_i^{\mathrm{L}} < 1$). Let N_0 be the number of nodes of the initial boundary and $\gamma = \max_{k=0,\ldots,i-1} \{\gamma_k^{\mathrm{R}}, \gamma_k^{\mathrm{L}}\}$. Then, the upper bound of the number of nodes in each subdomain of the ith partition can be approximated by

$$N_i = \gamma^i N_0 \tag{16}$$

Third, we evaluate the number of subdomains in a given ith partition. If it is assumed that all branches in the bin-tree representation always reach the same level (note that this is equivalent to assume that a uniform nodal density is prescribed). Then, the total number of subdomains is 2^i, see Figure 9.

Now, it is possible to compute the number of times the objective function is evaluated for the ith partition (the cost in the ith partition). Moreover, it can be written in terms of the number of nodes on the initial boundary

$$C_i = \tfrac{1}{2}\gamma^i N_0(\gamma^i N_0 - 1)2^i \approx \tfrac{1}{2}N_0^2 k^i$$

where $k = 2\gamma^2$.

Since the total cost of the developed algorithm can be evaluated as the sum of the cost of all partitions (levels in the bin-tree representation), the number of partitions required to generate the whole mesh has to be deduced. Starting from an initial boundary with N_0 nodes, the domain is recursively subdivided until quadrilateral elements are left. Let p be the number of partitions needed to obtain a quadrilateral element. Taking into account Equation (16), the number of partitions can be approximated by

$$\gamma^p N_0 = 4 \tag{17}$$

From (17) an estimate of the number of partitions can be deduced,

$$p = a \ln N_0 + b \tag{18}$$

where a and b are two constants independent of N_0, $a = -1/\ln \gamma$ and $b = \ln 4$. Note that $p = \mathcal{O}(\ln N_0)$.

Finally, the total cost of the developed algorithm can be evaluated as

$$TC = \sum_{i=0}^{p} C_i = \frac{1}{2}N_0^2\left(1 + k^2 + k^3 + \cdots + k^p\right) = \frac{1}{2}N_0^2\frac{k^{p+1} - 1}{k - 1} \approx N_0^2 k^{a \ln (N_0) + b + 1}$$

where the relationship (18) has been taken into account. Therefore,

$$TC = \mathcal{O}\left(N_0^2 k^{a \ln(N_0) + b + 1}\right) = \mathcal{O}\left(N_0^{2 + a \ln (k)}\right) \tag{19}$$

In order to relate in a simple manner the computational cost with the generated mesh, comparison will be made with the total number of nodes of the final mesh, N_T. For simple domains, the final mesh has approximately $N_T \approx N_0^2$ nodes. This assumption is corroborated in the examples shown below. Thus, the total computational cost is

$$TC = \mathcal{O}\left(N_T^{1 + a/2 \ln (k)}\right) \equiv \mathcal{O}(N_T^{\nu}) \tag{20}$$

Note that, for an initial square domain with a constant element size prescribed distribution, $\gamma = 3/4$ because the best splitting line is always the shorter one. Therefore, $k = 2\gamma^2 = 9/8$, $a = -1/\ln \gamma = -1/\ln 3/4$ and $\nu = 1.2$.

A numerical experiment has been carried out in order to find the numerical value of ν in expression (20). The examples presented in Figures 6 and 8 as well as the four first examples presented in the next section have been meshed several times with different element size distributions for each domain. In all cases, the minimum value of the total number of elements has been 5000.

Table II. Slopes of the linear regressions.

Example	N_T vs N_0	CPU vs N_0	CPU vs N_T
Only examples without background mesh	1.96	2.32	1.15
Only examples with background mesh	1.75	2.27	1.25
All examples	1.74	2.01	1.21

Notice that these examples contain all the characteristics that a general domain could show: simple-connected and multi-connected domains, constant and variable element size distributions, as well as the utilization of background meshes. For each case, the following linear regression have been performed:

$$\ln N_T = c_1 \ln(N_0) + d_1$$

$$\ln \mathrm{TC} = c_2 \ln(N_0) + d_2$$

$$\ln \mathrm{TC} = c_3 \ln(N_T) + d_3$$

where the total cost has been approximated by the CPU time spent to generate the whole mesh. Table II shows the obtained values for c_i, $i = 1, \ldots, 3$. These results corroborate the assumption that $N_T \approx N_0^2$. Moreover, when all cases are considered, the general performance of the algorithm is in concordance with expression (20) with $v = 1.2$.

6. EXAMPLES

In order to assess the quality of the mesh generation algorithm described above, five numerical examples are presented. They illustrate the capabilities of the new meshing algorithm in several environments: (i) constant element size distribution prescribed on the boundary of the domain, (ii) variable element size distribution prescribed on the boundary of the domain, (iii) element size distribution prescribed on a background mesh, (iv) domain surrounded by a ragged boundary and (v) application to adaptive techniques based on error estimation.

In the first example, the water around a dock is discretized. A constant element size distribution is prescribed on its boundary. Figure 10(a) shows an intermediate state in the meshing process. The final mesh, composed by 8291 nodes and 7600 elements, is presented in Figure 10(b). It is important to note that with a simple input (just the co-ordinates of the vertexes of the contour and the prescribed element size) a complex domain is discretized with a structured mesh without any previous partition of the domain.

The second example corresponds to a rail cross-section. The element size distribution is prescribed at its vertexes and a high element concentration is defined in the load zone. Figure 11 shows the final mesh. It is composed by 1541 nodes and 1413 elements. As it can be observed, a structured mesh is generated where the geometry and prescribed values allow it. Also, a symmetric mesh is generated at the base of the rail. Moreover, a smooth transition is obtained between low- and high-density areas.

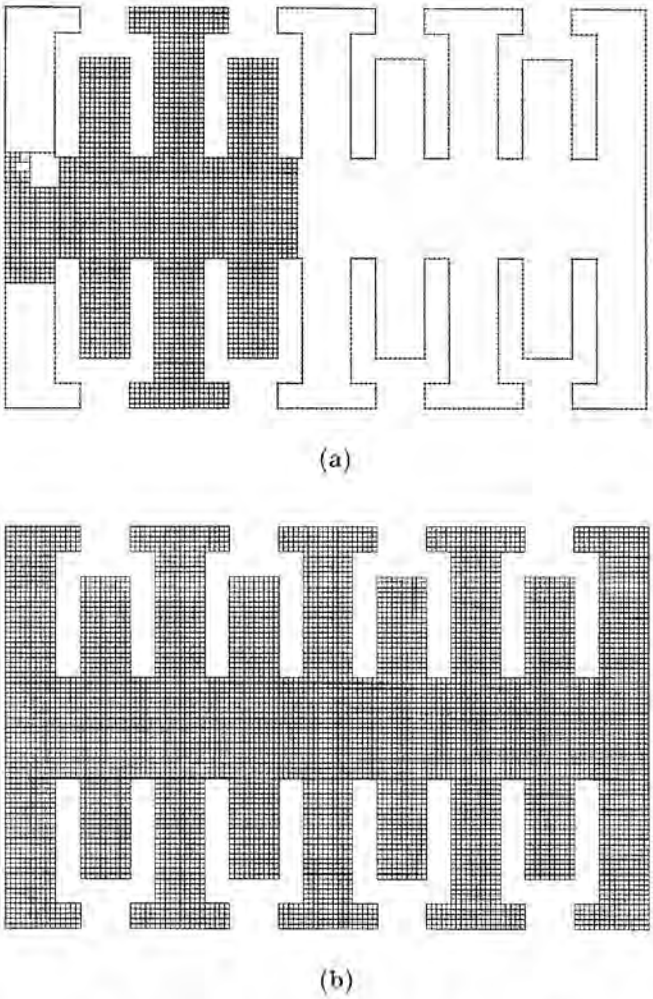


Figure 10. Discretization of a dock: (a) intermediate state in the meshing process; (b) final mesh.

The third example is the discretization of a gear. In this case a background mesh is used to concentrate elements along one direction. The final mesh and the background mesh are shown in Figures 12(a) and 12(b), respectively. The final mesh is composed of 1654 nodes and 1568 elements. As it can be seen, well-shaped elements are generated in this case, even on the curved part of the boundary. Furthermore, smooth size transitions are also obtained and spurious element concentration does not appear.

In the fourth example the developed algorithm is applied to the discretization of a domain defined by a ragged boundary. In particular, a mesh for inner part of the port of Barcelona (Spain) is generated. It is composed of 29 691 nodes and 28 537 elements. Figure 13 shows the final mesh. The boundary corresponding to the harbor mouth is defined by a circular arc on the right-hand side of the mesh. The breakwater corresponds to the boundary on the top of the mesh, whereas quays on the dry land corresponds to the left-hand side and bottom boundaries.

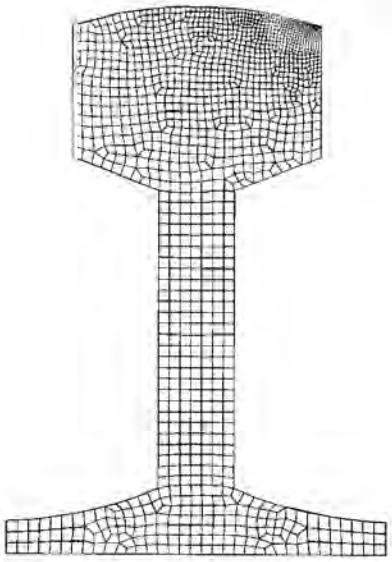
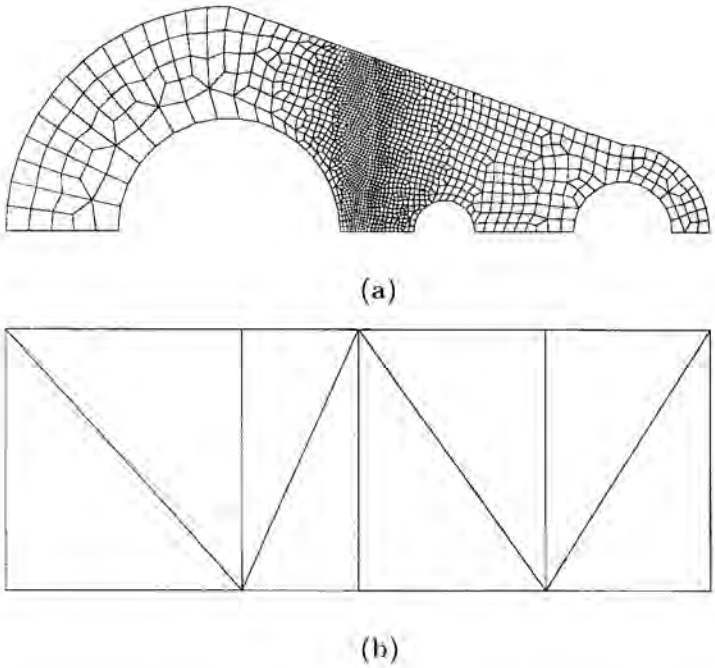


Figure 11. Discretization of a rail cross section.



(a)

(b)

Figure 12. Discretization of a gear: (a) generated mesh; (b) background mesh.

Figure 13. Discretization of the port of Barcelona.

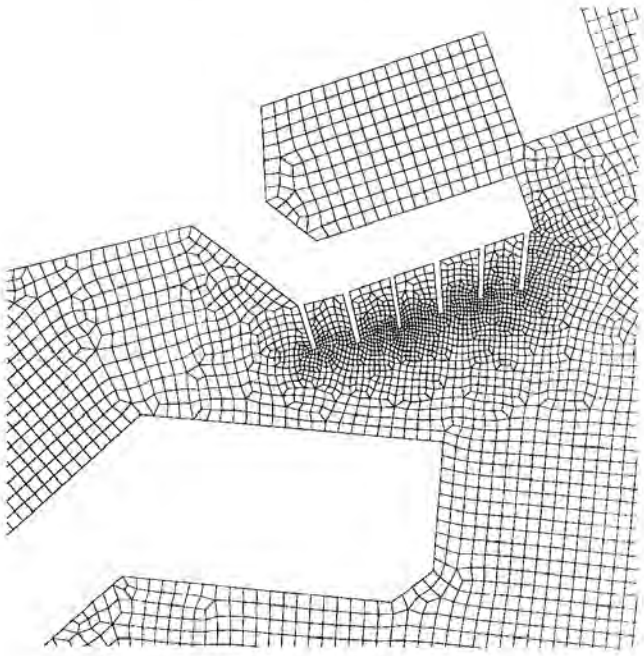


Figure 14. Detail of a part of the outer dock of the port of Barcelona.

Due to some boundary details (small docks, sharp corners), different values of the element size are prescribed. For instance, small elements are used near small docks whereas bigger elements are used near straight boundaries or in the harbor mouth. Figure 14 shows the generated mesh around small docks located between the dry land and the breakwater. As it can be seen, well-shaped elements are obtained and the final mesh tends towards a structured mesh when possible. A detail of the final mesh around two small docks near the harbor mouth is presented in Figure 15. As it can be observed, a smooth transition between high and low element density areas is obtained. Moreover, well-shaped elements are generated even in non-convex corners.

The fifth example shows how the developed algorithm can be applied to adaptive techniques. The error estimator developed in References [35, 36], is applied to the adaptive computation of

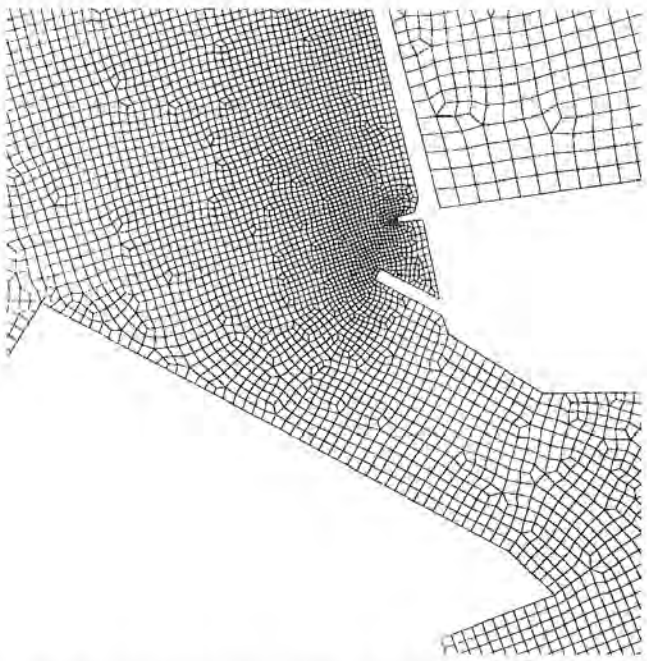


Figure 15. Detail of a part of the inner dock near the mouth of the port of Barcelona.

the compression of a plane strain rectangular specimen with two imperfections (circular openings inside the material) [37]. The mesh generator algorithm is included in the adaptive process as follows. Given an initial quadrilateral mesh, the finite element solution and error estimation are computed on it. From this error estimation, a desired element size is evaluated at the nodes. Then, the initial mesh is transformed into a triangular mesh by connecting two opposed nodes of each element. Finally, this new mesh together with the evaluated new element size are used as background mesh to generate a new one. This process is repeated until the prescribed accuracy is reached, see Reference [37]. Figure 16 shows a succession of generated meshes. It is worth noting that, as the adaptive process evolves the elements concentrates in two bands according to the error estimation. Note that regular and well-shaped elements are generated even in a small region where a high gradient of the element size is prescribed.

7. CONCLUSIONS

In this paper a new automatic and efficient two-dimensional unstructured quadrilateral element mesh generator is presented. The user interaction has been reduced to the specification of the boundary geometry and the desired element size at some base point on the boundary or at the nodes of a background mesh. The technique is based on a recursive splitting of the domain until only quadrilateral elements of the desired size are left. Moreover, the algorithm is decomposed, for computational efficiency, into three phases: (i) determination of the best splitting line (where new criteria have been developed in order to define the objective function), (ii) node placement (where a

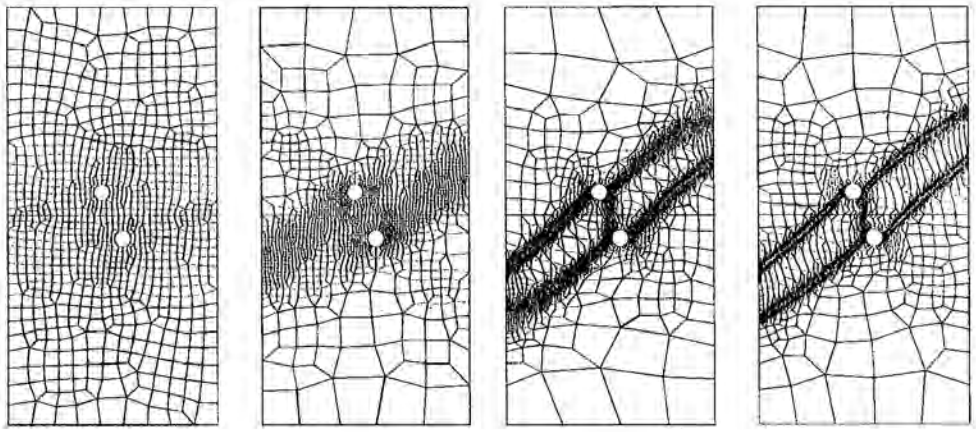


Figure 16. Application of the mesh generation algorithm to adaptive computations: (a) Initial mesh (462 elements); (b) Second mesh (856 elements); (c) Third mesh (2235 elements); (d) Final mesh (3307 elements).

new algorithm has been deduced), and (iii) mesh quality enhancement (where a modification of the smoothing method developed by Giuliani has been presented). The final mesh can be interpreted as the mesh that optimizes the given objective function. The recursive algorithm induces an efficient bin-tree structure which has been used to prove that the cost of the new algorithm is $\mathcal{O}(N_T^{1.2})$. A wide range of numerical experiments have verified this result. Finally, several examples have been presented to show the new algorithm capabilities.

APPENDIX

Our goal here is, first generalize Equation (14) for the two-dimensional problem, and second, develop an explicit and analytic expression to recurrently compute the node position. To this end, consider a segment defined by the end points $[P_0, P_n]$ (a candidate to splitting line). Along $[P_0, P_n]$ nodes, P_k, $k = 1, 2, \ldots, n-1$, must be placed with the desired distance, i.e. the requested element size. The distance between two consecutive nodes is given by a background mesh of triangles. If two consecutive nodes P_k and P_{k+1} lie inside one triangle, the following equation is verified:

$$\int_{P_k}^{P_{k+1}} \frac{\mathrm{d}\xi}{h(\xi)} = \frac{n_e^*}{n} \tag{A1}$$

where n_e^* is given by Equation (13), namely

$$n_e^* = \int_{P_0}^{P_n} \Pi(\xi)\,\mathrm{d}\xi = \int_{P_0}^{P_n} \frac{1}{h(\xi)}\,\mathrm{d}\xi \tag{A2}$$

Since $[P_k, P_{k+1}]$ lies inside a triangle, the element size along this segment is linear [3, 7], that is,

$$\Pi(\xi) = \frac{1}{h(\xi)} = \frac{1}{a\xi + b} \tag{A3}$$

where a and b are two constants. These constants are computed from the values of the element size $\hat{h}_i$ and $\hat{h}_j$ at the intersection of the line defined by P_k and P_{k+1} and the background triangle. Let us denote by $\hat{P}_i$ and $\hat{P}_j$ the points where the line $P_k P_{k+1}$ intersects its corresponding triangle, and by $\hat{\xi}_i$ and $\hat{\xi}_j$ the values of parameter ξ assigned to points $\hat{P}_i$ and $\hat{P}_j$. Then,

$$a = \frac{\hat{h}_j - \hat{h}_i}{\hat{\xi}_j - \hat{\xi}_i}$$

$$b = \hat{h}_i - \hat{\xi}_i \left[\frac{\hat{h}_j - \hat{h}_i}{\hat{\xi}_j - \hat{\xi}_i} \right]$$

Note that function $h(\xi)$ is always positive because it is the element size. Therefore, Equation (A3) can be used to compute the integral that appears on the left-hand side of expression (A1). On an arbitrary element side defined by P_k and P_{k+1} that lies on a segment limited by the end points $\hat{P}_j$ and $\hat{P}_{j+1}$ the following equality applies:

$$\int_{P_k}^{P_{k+1}} \frac{\mathrm{d}\xi}{h(\xi)} = \frac{\mathrm{d}(\hat{P}_j, \hat{P}_{j+1})}{\hat{h}_{j+1} - \hat{h}_j} \ln\left(\frac{h_{k+1}}{h_k}\right)$$

where $d(\cdot, \cdot)$ denotes the distance between two points. Replacing the previous result in Equation (A1)

$$h_{k+1} - h_k = h_k \left\{ \exp\left[\frac{n_e^*}{n} \frac{\hat{h}_{j+1} - \hat{h}_j}{\mathrm{d}(\hat{P}_j, \hat{P}_{j+1})} \right] - 1 \right\} \tag{A4}$$

Finally, since a linear interpolation has been used between the prescribed values at points $\hat{P}_j$ and $\hat{P}_{j+1}$, we get

$$\frac{\hat{h}_{j+1} - \hat{h}_j}{d(\hat{P}_j, \hat{P}_{j+1})} = \frac{h_{k+1} - h_k}{d(P_k, P_{k+1})}$$

which can be replaced in Equation (A4)

$$d(P_{k+1}, P_k) = \frac{d(\hat{P}_j, \hat{P}_{j+1})}{\hat{h}_{j+1} - \hat{h}_j} h_k \left\{ \exp\left[\frac{n_e^*}{n} \frac{\hat{h}_{j+1} - \hat{h}_j}{d(\hat{P}_j, \hat{P}_{j+1})} \right] - 1 \right\} \tag{A5}$$

Equation (A5) is the keystone in the node placement algorithm. It allows to compute the position of the nodes along the splitting line in a recurrent manner when a background mesh is used. Note that if background mesh is not used, expression (A5) is still valid. In this case, the prescribed values are defined at the splitting line end points. Note that Equation (A5) has been deduced

assuming that nodes P_k and P_{k+1} lie in the same triangle. If they lie in different triangles, segment $[P_k, P_{k+1}]$ is partitioned in two parts, and the same expression applies.

In Equation (A5), the theoretical number of elements side to be generated along the splitting line, n_e^*, must be evaluated. According to (A3), between two consecutive intersections of the splitting line with the background mesh, $\hat{P}_i$ and $\hat{P}_j$, the following equation is verified:

$$\int_{\hat{P}_i}^{\hat{P}_j} \frac{\mathrm{d}\xi}{h(\xi)} = \frac{1}{a} \ln(a\,\xi + b)\Big|_{\hat{\xi}_i}^{\hat{\xi}_j} = \frac{\hat{\xi}_j - \hat{\xi}_i}{\hat{h}_j - \hat{h}_i} \ln\left(\frac{\hat{h}_j}{\hat{h}_i}\right) \tag{A6}$$

Using the previous equation, it is possible to find an analytic expression for the theoretical number of elements side to be generated along the segment defined by the end points $[P_0, P_n]$:

$$\begin{aligned} n_e^* &= \int_{P_0}^{P_n} \frac{\mathrm{d}\xi}{h(\xi)} = \int_{P_0}^{\hat{P}_1} \frac{\mathrm{d}\xi}{h(\xi)} + \sum_{i=1}^{m-1} \int_{\hat{P}_i}^{\hat{P}_{i+1}} \frac{\mathrm{d}\xi}{h(\xi)} + \int_{\hat{P}_m}^{P_n} \frac{\mathrm{d}\xi}{h(\xi)} \\ &= \frac{\hat{d}(P_0, P_1)}{\hat{h}_1 - h_0} \ln\left(\frac{\hat{h}_1}{h_0}\right) + \sum_{i=1}^{m-1} \frac{d(\hat{P}_i, \hat{P}_{i+1})}{\hat{h}_{i+1} - \hat{h}_i} \ln\left(\frac{\hat{h}_{i+1}}{\hat{h}_i}\right) + \frac{d(\hat{P}_m, P_n)}{h_n - \hat{h}_m} \ln\left(\frac{h_n}{\hat{h}_m}\right) \end{aligned} \tag{A7}$$

where $\hat{P}_1, \ldots, \hat{P}_m$ denotes the intersection points between the splitting line $[P_0, P_n]$ and the background mesh.

REFERENCES

1. Cavendish JC. Automatic triangulation of arbitrary planar domains for the finite element method. *International Journal for Numerical Methods in Engineering* 1974; **8**:679–697.
2. Kikuchi N. Adaptive grid-design methods for finite element analysis. *Computer Methods in Applied Mechanics and Engineering* 1986; **55**:129–160.
3. Peraire J, Vahdati M, Morgan K, Zienkiewicz OC. Adaptive remeshing for compressible flow computations. *Journal of Computational Physics* 1987; **72**:449–466.
4. Rebay S. Efficient unstructured mesh generation by means of delaunay triangulation and Bowyer–Watson algorithm. *Journal of Computational Physics* 1993; **106**(1):125–138.
5. Sluiter MLC, Hansen DL. A general purpose automatic mesh generator for shell and solid finite elements. *Proceedings of the 2nd International Computer Engineering Conference. Computers in Engineering*, ASME: Computer Engineering Division, 1993:29–34.
6. Wordenweber B. Finite element mesh generation. *Computer Aided Design* 1984; **16**:285–291.
7. Talbert JA, Parkinson AR. Development of an automatic two-dimensional finite element mesh generator using quadrilateral elements and Bèzier curve boundary definition. *International Journal for Numerical Methods in Engineering* 1990; **29**:1551–1567.
8. Liu YC, El Maraghy HA, Zhang KF. An expert system for forming quadrilateral finite elements. *Engineering Computations* 1990; **7**:249–257.
9. Blacker TD, Stephenson MB. Paving: a new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering* 1991; **32**:811–847.
10. Zhu JZ, Zienkiewicz OC, Hinton E, Wu J. A new approach to the development of automatic quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering* 1991; **32**:849–866.
11. Rank E, Schweingruber M, Sommer M. Adaptive mesh generation and transformation of triangular to quadrilateral meshes. *Communications in Numerical Methods in Engineering* 1993; **9**:121–129.
12. Lee CK, Lo SH. A new scheme for the generation of a graded quadrilateral mesh. *Computers and Structures* 1994; **52**(5):847–857.
13. Haber R, Shephard MS, Abel JF, Gallagher RH, Greenberg DP. A general two-dimensional graphical finite element preprocessor utilizing discrete transfinite mappings. *International Journal for Numerical Methods in Engineering* 1981; **17**:1015–1044.
14. Ho-Le K. Finite element mesh generation methods: a review and classification. *Computer Aided Design* 1988; **20**(1): 27–38.

15. Díez P, Huerta A. A unified approach to remeshing strategies for finite element h–adaptivity. *Computer Methods in Applied Mechanics and Engineering* 1999; **176**(1-4):215–229.
16. Huerta A, Rodríguez–Ferran A, Díez P, Sarrate J. Adaptive finite element strategies based on error assessment. *International Journal for Numerical Methods in Engineering* 1999; **46**:1803–1818.
17. George PL. *Automatic Mesh Generation. Application to Finite Element Methods*. Wiley: Paris, 1991.
18. Thacker WC, González A, Putland GE. A method for automating the construction of irregular computational grids for storm surge forecast models. *Journal of Computational Physics* 1980; **37**:371–387.
19. Zienkiewicz OC, Phillips DV. An automatic mesh generation scheme for plane and curved surfaces by isoparametric co-ordinates. *International Journal for Numerical Methods in Engineering* 1971; **3**:519–528.
20. Gordon WJ, Hall CA. Construction of curvilinear co-ordinate systems and applications to mesh generation. *International Journal for Numerical Methods in Engineering* 1973; **7**:461–477.
21. Brown PR. A non-interactive method for automatic generation of finite element meshes using the Schwarz–Christoffel transformation. *Computer Methods in Applied Mechanics and Engineering* 1981; **25**:101–126.
22. Thompson JF, Warsi ZUA, Mastin CW. *Numerical Grid Generation. Foundations and Applications*. Elsevier: New York, 1985.
23. Thompson JF. A general three-dimensional elliptic grid generation system on a composite structure. *Computer Methods in Applied Mechanics and Engineering* 1987; **64**:377–411.
24. Knupp PM. A robust elliptic grid generator. *Journal of Computational Physics* 1992; **100**:409–418.
25. Cavendish JC, Field DA, Frey WH. An approach to automatic three-dimensional finite element mesh generation. *International Journal for Numerical Methods in Engineering* 1985; **21**:329–347.
26. Lewis RW, Zheng Y, Gethin DT. Three-dimensional unstructured mesh generation: Part 3. Volume meshes. *Computer Methods in Applied Mechanics and Engineering* 1996; **134**:285–310.
27. Zheng Y, Lewis RW, Gethin DT. Three-dimensional unstructured mesh generation: Part 1. Fundamental aspects of triangulation and point creation. *Computer Methods in Applied Mechanics and Engineering* 1996; **134**:249–268.
28. Zheng Y, Lewis RW, Gethin DT. Three-dimensional unstructured mesh generation: Part 2. Surface meshes. *Computer Methods in Applied Mechanics and Engineering* 1996; **134**:269–284.
29. Lo SH. A new generation scheme for arbitrary planar domains. *International Journal for Numerical Methods in Engineering* 1985; **21**:1403–1426.
30. Löhner R, Parikh P. Three-dimensional grid generation by the advancing-front method. *International Journal for Numerical Methods in Fluids* 1988; **8**:1135–1149.
31. Jin H, Wiberg NE. Two-dimensional mesh generation, adaptive remeshing and refinement. *International Journal for Numerical Methods in Engineering* 1990; **29**:1501–1526.
32. Herrmann LR. Laplacian–isoparametric grid generation scheme. *Journal End. Mechanical Division. ASCE* 1976; **102**:749–756.
33. Giuliani S. An algorithm for continuous reasoning of the hydrodynamic grid in Arbitrary Lagrangian–Eulerian codes. *Nuclear Engineering and Design* 1982; **72**(2):205–212.
34. Sarrate J, Huerta A. An improved algorithm to smooth graded quadrilateral meshes preserving the prescribed element size. *Communications in Numerical Methods in Engineering* (in press).
35. Díez P, Egozcue JJ, Huerta A. A posteriori error estimation for standard finite element analysis. *Computer Methods in Applied Mechanics and Engineering* 1998; **163**(1-4):141–157.
36. Huerta A, Díez P. Error estimation including pollution assessment for non linear finite element analysis. *Computer Methods in Applied Mechanics and Engineering* 2000; **181**:21–41.
37. Díez P, Arroyo M, Huerta A. Adaptivity based on error estimation for viscoplastic softening materials. *Mechanics of Cohesive and Frictional Materials* 2000; **5**:87–112.