INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, VOL. 39, 2055–2071 (1996)

OPTIMIZATION STRATEGIES IN UNSTRUCTURED MESH GENERATION

PABLO D. ZAVATTIERI

Instituto Balseiro, Universidad Nac. de Cuyo and Comisión Nacional de Energía Atómica, Argentina

ENZO A. DARI* AND GUSTAVO C. BUSCAGLIA

Instituto Balseiro, Universidad Nac. de Cuyo and Centro Atómico Bariloche, 8400 Bariloche, Comisión Nacional de Energía Atómica, Argentina

SUMMARY

We propose a new optimization strategy for unstructured meshes that, when coupled with existing automatic generators, produces meshes of high quality for arbitrary domains in 3-D. Our optimizer is based upon a non-differentiable definition of the quality of the mesh which is natural for finite element or finite volume users: the quality of the worst element in the mesh. The dimension of the optimization space is made tractable by restricting, at each iteration, to a suitable neighbourhood of the worst element. Both geometrical (node repositioning) and topological (reconnection) operations are performed. It turns out that the repositioning method is advantageous with respect to both the usual node-by-node techniques and the more recent differentiable optimization methods. Several examples are included that illustrate the efficiency of the optimizer.

KEY WORDS: unstructured mesh generation; optimization; smoothing; finite elements; finite volume methods

1. INTRODUCTION

Recent work in automatic three-dimensional mesh generation has shown that present algorithms (at least frontal and Delaunay ones) usually fail in providing a valid mesh for finite element computations (see, e.g. Reference 1). Awfully distorted elements appear, with angles between faces as low as 0·01 degrees or as high as 179·99. This led Dari and Buscaglia to propose, in Reference 2, that the construction of a valid mesh should be split into two steps. The first one is the automatic generation of an *initial* mesh, with no restriction on the elements' geometry except for the 'non-overlapping' condition (in fact, even this condition can be relaxed). The second one is the *optimization* of the mesh, repositioning the nodes and possibly modifying the topological structure (also called node-adjacency structure) so as to maximize some appropriate quality measure.

A natural question is whether the quality of frontal- or Delaunay-generated meshes can be improved by node repositioning, and how much. In Reference 2 it was shown that drastic improvement can be obtained by node repositioning, yielding minimum angles as high as 10–30° and maximum ones as low as 150–170°. The point is, thus, that the *topological* structure of as-generated meshes is quite acceptable, while their *geometrical* structure is poor. Nevertheless, topological changes can indeed improve unstructured meshes. These changes can be accomplished by successive application of some elementary operations (a 3-D analogue to the

* Fellow of CONICET, Argentina

CCC 0029–5981/96/122055–17

Received 22 May 1995
Revised 21 August 1995

well-known diagonal-swapping in 2-D). The usefulness of topological optimization was also proved in Reference 2, but the algorithm failed in certain situations.

In this paper, we propose some mesh optimization strategies that solve many of the difficulties encountered by users of 3-D mesh generation programs. First we consider *geometrical optimization*, also called *smoothing*; and then *topological optimization*, also called *relaxation* or *reconnection*. Both of them have precedents in the literature. Smoothing has long been used as a cosmetic process for meshes, and several strategies already exist.[3-7] These smoothing algorithms consist, in general, of the solution of some (linear or non-linear) elastic auxiliary problem. Such approaches, which are very effective in 2-D, frequently fail in three-dimensions (see, e.g. Reference 8). As a consequence, a new line of research has appeared under the general name of *mesh optimization*,[2,9-14] concerning mainly what we refer here to as geometrical mesh optimization or smoothing (i.e. node-repositioning based on the optimization of the mesh quality). The smoothing methods we propose here incorporate several improvements with respect to previous ones:

(1) They are global, in the sense that the positions of the nodes are not updated one by one, but instead a complete sector of the mesh (perhaps the whole mesh) is moved at each optimization step. This increases the dimension of each optimization step, allowing for better ascent directions to be found. Cabello *et al.*,[10] Stamatis and Papailiou[12] and Zhang and Trépanier[14] have recently presented some encouraging results in this direction, but their meshes were two-dimensional. Another optimization method of global nature *for structured meshes* was early proposed by Kennon and Dulikravich.[11]

(2) They are *non-differentiable* (or almost non-differentiable). Previous global optimization strategies[10-12,14] define the mesh quality as some average of the quality of its elements. This definition ignores what finite element (especially CFD) practitioners have known for long: One unacceptable element renders the mesh useless. Our methods incorporate the proposal made by Dari and Buscaglia,[2,9]

$$Q_{\text{global}} = \min_K Q_K \tag{1}$$

which considers the global quality of a mesh as being the same as the quality of its worst element. The problem is thus that of maximizing a non-differentiable objective function of min type.

(3) They are *efficient*. The simultaneous optimization of all node locations of a 3-D mesh consisting of, say, 200 000 elements, is obviously intractable. In Section 4.1 it is shown that, to improve the worst element of the mesh, it is only necessary to move those nodes belonging to a small vicinity of it (up to its neighbours of order 1 or 2). In 3-D this reduces the dimension of the problem to, say, a few hundreds. Also, we have incorporated an inexpensive measure of the quality of a tetrahedron, namely

$$Q_K = C_d \frac{V_K}{P_K^d} \tag{2}$$

where V_K is the volume of element K, P_K its perimeter (the sum of the lengths of its six edges, three in 2-D), $d = 2$ (2-D) or 3 (3-D), and C_d a constant which renders the quality of an equilateral simplex equal to one (= 20·784619 in 2-D and = 1832·8208 in 3-D). Our experiments have shown that this definition of quality has another advantage: It penalizes obtuse angles more strongly than acute ones.

Concerning mesh relaxation (or topological optimization), an important precedent is the article of Frey and Field[15], which considered the diagonal swapping in 2-D. Ours is one possible

extension of such strategies to 3-D. We propose four elementary operations, to be performed (if possible) only when the geometrical quality (2) of the elements involved increases as a result of the operation. This geometrical criterion is quite different from the topological one proposed by Dari and Buscaglia,[2] and the results are much better. Other important precedents are the methods of Coupez[16] and of Marcum and Weatherill,[17] who consider other classes of topological operations. Our relaxation method can be viewed as a variant of Coupez's one.

The plan of this article is as follows: In Section 2, we present the geometrical optimization methods. In Section 3 this is complemented with a topological optimization algorithm. In Section 4 we report and discuss several examples. Our conclusions are left for Section 5.

2. GEOMETRICAL OPTIMIZATION ALGORITHMS

2.1. *The objective function*

We define the quality of a generic element (simplex) K of the mesh through equation (2). This quality is a differentiable function of the position of the nodes (vertices) of the element, except for singular situations (all of them contained in the set of nodal configurations of zero volume). Our objective function is the global quality of the mesh, defined through equation (1). This is the first proposal, and the resulting method will be hereafter referred to as *non-differentiable optimization method.* The second proposal is a regularization of the objective function (1), given by

$$\tilde{Q}_{\text{global}} = -\sum_{K}(1 - Q_K)^p \tag{3}$$

where p is a suitably chosen exponent. As all element qualities are lower than 1, (3) is a proper definition. Moreover, as p is increased, the worst elements in the mesh will eventually dominate the summation. The non-differentiable global quality (1) is thus recovered as p goes to infinity. The objective function defined by (3) is a differentiable function, and the consequent method will thus be called *differentiable optimization method.*

2.2. *The parameter (or optimization) space*

The objective function will be maximized over the space of nodal locations. The dimension of this space becomes intractable if the complete mesh is considered. We tackle this difficulty optimizing, at each step, only the nodal locations of those nodes that belong to the worst element of the mesh, together with their nth-order neighbours (with $n = 1$ or 2). The positions of this small number of nodes are the parameter space in which we maximize the objective function. As the optimization goes on, the worst element of the mesh changes, and so does the parameter space.

We will refer as *moving nodes* to those nodes that are moving at a given stage of the algorithm. The *moving nodes* are selected among the *movable nodes.* In the present version of our method, boundary nodes are non- movable. If all the nodes belonging to an element are non-movable, the element is not considered. Once the worst element of the mesh has reached its maximum quality, we mark its nodes as non-movable so that the algorithm proceeds to optimize other bad elements of the mesh.

The algorithm through which we administer the identification of nodes as moving, movable or non-movable is the following

Global Algorithm
Initialization and Global Parameters: Specify a neighbourhood level NL and a maximum number of iterations M. Set boundary nodes as non-movable. Initialize an integer auxiliary constant PREV with 0.
Iterations:

1. Identify the worst element among those elements of the mesh containing at least one *movable node*, KWORST. If the number of *movable nodes* is zero, Stop.
2. If KWORST = PREV, set the four (three in 2-D) nodes of KWORST as *non-movable* and go back to 1.
3. Identify the nodes in a neighbourhood of order NL of KWORST. The neighbours of order 0 are the four nodes belonging to KWORST. Neighbours of order 1 are those nodes that are not neighbours of order 0, but are connected by an edge with at least one neighbour of order 0; and so on.
4. Set as *moving nodes* those identified in 3 that have not previously been set as *non-movable*. With these *moving nodes* apply the *Local Algorithm* (see below) to optimize the quality of the sub-mesh formed by the elements that contain the *moving nodes*. This sub-mesh is called hereafter *optimization cluster*.
5. Assign to PREV the value KWORST.
6. Go back to 1, or Stop if the number of iterations is equal to M.

2.3. *The non-differentiable optimization algorithm*

We have implemented a steepest-descent-like algorithm taken from Polak.[18] Let ψ be the objective function Q_{global}, and let $\bar{\mathbf{x}} \in \Re^n$ be the vector of co-ordinates of the moving nodes. The minus sign is introduced to turn the problem into a *minimization* one, just because the terminology in the literature is 'minimum-oriented'. We thus have to minimize $\psi = \max_K f_K$, where $f_K = -Q_K$ is the quality of element K, with negative sign. Of course, only those elements containing moving nodes are considered, let us assign to these elements the numbers $1, 2, \ldots m$. If $\bar{x}_0$ represents the actual position of the moving nodes, the update Δx results from the following

Local Algorithm:

1. Compute the search direction

$$\bar{\delta} = -\sum_{K=1}^{m} \mu_K^* \nabla f_K(\bar{x}_0) \tag{4}$$

where $\{\mu_K^*\}_{K=1,\ldots,m} \equiv \underline{\mu}^*$ is the solution of

$$\underline{\mu}^* = \arg\min_{\underline{\mu} \in \Xi} \left\{ \left[\sum_{K=1}^{m} \mu_K [\psi(x_0) - f_K(x_0)] \right]^2 + L^2 \left\| \sum_{K=1}^{m} \mu_K \nabla f_K(\bar{x}_0) \right\|^2 \right\} \tag{5}$$

with

$$\Xi = \left\{ \underline{\mu} \in \Re^m \mid \mu_K \geqslant 0, \sum_{K=1}^{m} \mu_K = 1 \right\} \tag{6}$$

This quadratic programming sub-problem is solved using the Active Set Method (see, e.g. Reference 19). Explicit expressions for ∇f_K can be found in the Appendix. The scaling factor L in (5) is a typical length. It is defined as the square root of the average area of elements in the cluster (times a constant that makes L equal to the edge length if the elements are

equilateral triangles or tetrahedra). Notice that with this definition L is constant along *Local Iterations.*

2. Given $\alpha, \beta \in (0, 1)$, compute the step length λ^* using Armijo's rule:

$$\lambda^* = \max\{\lambda \in \Re \mid \lambda = \beta^k (k = 0, 1, 2, \ldots), \psi(\bar{x}_0 + \lambda\bar{\delta}) - \psi(\bar{x}_0) \leqslant \lambda\alpha \|\bar{\delta}\|^2\} \tag{7}$$

3. $\Delta\bar{x} = \lambda^*\bar{\delta}$.

Remark 1. Solving the quadratic problem (5) to find the search direction can be very costly, especially in 3-D where the number of elements containing moving nodes is large even with NL taken as 1. Following the ideas in Reference 18, the efficiency can be improved including a user-defined tolerance ε and computing the set I_ε of ε-affected elements defined as those elements with quality close to the minimum one, i.e.

$$K \in I_\varepsilon \Leftrightarrow |\psi(\bar{x}_0) - f_K(\bar{x}_0)| < \varepsilon \tag{8}$$

Now, the dimension of the quadratic problem is lowered replacing Ξ by Ξ_ε defined by

$$\Xi_\varepsilon = \left\{\underline{\mu} \in \Re^m \mid \mu_K \geqslant 0, \sum_{K=1}^{m} \mu_K = 1, \mu_K = 0 \text{ if } K \notin I_\varepsilon\right\} \tag{9}$$

Clearly, the original problem is recovered when ε is large enough.

Remark 2. The iterations of the *Local Algorithm* must be stopped when the change in the objective function is below some tolerance $\mathrm{TOL}_{\mathrm{loc}}$. A good scale for this tolerance is the value taken for ε: We usually adopt $\mathrm{TOL}_{\mathrm{loc}} = \varepsilon/10$. However, not all optimization clusters must be iterated until convergence. Most clusters with bad initial quality attain very high quality in just a few iterations, and then spend lots of *Local Iterations* improving the quality from, say, 0·4 to 0·45. A good implementation should consider this possibility and stop the *Local Iterations* whenever a cluster has reached a quality that is greater that some given *threshold* value, so as to avoid wasting CPU time improving already-good clusters, and turn to worse ones. From our experience, a good choice for this threshold quality value is 0·2 or 0·3.

2.4. The differentiable optimization algorithm

We have implemented the standard steepest-descent method.[19] If ψ now is $-\tilde{Q}_{\mathrm{global}}$ (equation (3)), the descent direction is defined by

$$\bar{\delta} = -\nabla\psi = -\sum_{K=1}^{m} p[1 + f_K(\bar{x}_0)]^{p-1} \nabla f_K(\bar{x}_0) \tag{10}$$

where again $f_K = -Q_K$. Steps 2 and 3 remain as in 2.3. Explicit expressions for ∇f_K can be found in the Appendix.

3. THREE-DIMENSIONAL TOPOLOGICAL OPTIMIZATION ALGORITHM

The geometrical optimization considered in Section 2 is supplemented with the change of the node-adjacency structure of the mesh. Once the geometrical optimization iterations have stopped, we try to improve the mesh quality further through topological changes.

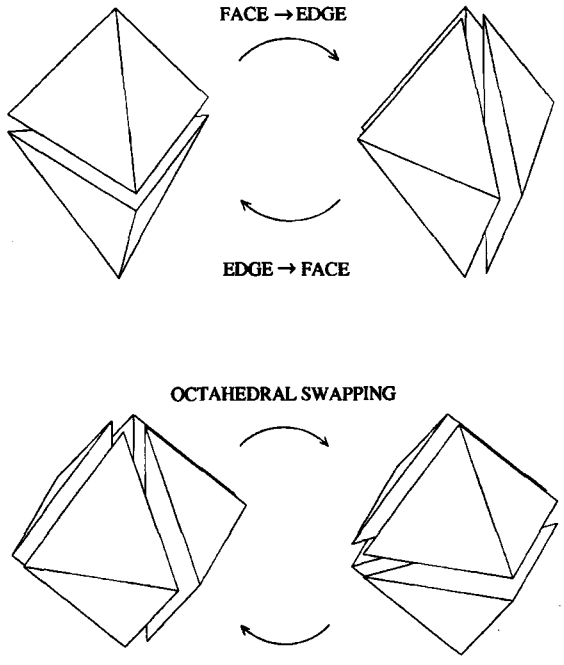

Figure 1. Topological optimization operations

We consider the following four topological operations in 3-D (See Figure 1):

Face → edge operation. Given two elements sharing a face, this operation consists of connecting the opposite vertices, deleting the common face. The resulting structure contains three elements instead of two (this operation is called local reconnection by Marcum and Weatherill[17]).

Edge → face operation. The inverse of *Face → edge*, it can only be performed if the edge is shared by three tetrahedra.

Octahedral swapping. If an edge in the mesh is shared by four tetrahedra, the union of these elements is an octahedron. This operation consists of swapping the edge among the three pairs of opposite nodes in the octahedron.

Cluster reconnection. Consider a cluster of tetrahedra that share some specific node or some specific edge of the mesh. This operation consists of modifying the internal structure of the cluster, leaving the surface of the cluster unchanged. Our implementation allows for the following changes of structure: (a) Deletion of the internal node of the cluster (if any) and connection of all of the external faces to one of the remaining nodes. (b) Connection of all of the external faces to a node located at the centre of gravity of the cluster. (c) Reconnection of the cluster, connecting all of the external faces to the most favorable node. Change (a) reduces the number of nodes by one if the cluster has an internal node, while change (b) increases the number of nodes by one if the cluster has no internal node. Change (c) applies to clusters without internal nodes. A *cluster reconnection* operation consists of analysing all possible changes and selecting the one that yields elements of best quality.

The last operation is taken from Coupez's method.[16] The algorithm we propose is based on the mesh quality, which is a *geometrical* quantity. We have observed that performing topological changes decided on a geometrical criterion is a much better choice than that proposed by Dari

and Buscaglia,[2] who used a topological criterion (they checked if the average number of elements sharing an edge approached the somewhat artificial value of 5·10).

Topological Algorithm

1. Analyse all faces in the mesh. If a *face* → *edge* operation yields elements of better quality, perform it.
2. Analyse all edges in the mesh shared by three tetrahedra, if an *edge* → *face* operation yields elements of better quality, perform it.
3. Analyse all edges in the mesh shared by four tetrahedra, performing *octahedral swapping* so as to obtain the best quality.
4. Analyse all clusters in the mesh. If a *cluster reconnection* operation yields elements of better quality, perform it.
5. Go back to 1 until the topological structure remains unchanged.

The computer cost is greatly reduced if, instead of analysing the whole mesh, just the surroundings of the worst elements are considered.

4. NUMERICAL EXAMPLES

4.1. Illustrative examples in 2-D: comparison of non-differentiable and differentiable methods, and determination of the order of neighborhood

We begin with an example in two dimensions, taking as domain the surface of Lake Nahuel Huapi, a very large and beautiful lake just in front of our institute. Its irregular boundaries are an excellent test for mesh generation algorithms in 2-D. This example will clearly show that 2-D mesh generation can be considered a solved problem. What lies ahead is to achieve the same robustness in 3-D, and we will show that optimization strategies significantly contribute in this direction.

We have specified as target density a non-uniform one, refined at the south east where Bariloche City lies. The mesh, generated using the Delaunay algorithm with program ENREDO,[20] consists of 11 172 triangles and 6056 nodes (see Figure 2). The minimum quality measure (equation (2)) for the as-generated mesh was 0·3445, corresponding to minimum/maximum angles in the mesh of 12·04/148·27°. The purpose of this example is to compare the performance of non-differentiable and differentiable methods, and to determine adequate values for the order of neighbourhood NL, for the tolerance ε in the non-differentiable method (equation 8), and for the regularization exponent p (equation (3)). In this academic example the tolerance of the Local Algorithm is specified as $\mathrm{TOL}_{\mathrm{loc}} = 10^{-5}$. No topology optimization is performed.

We first analyse the order of neighbourhood NL. Increasing NL means moving more nodes at a time to optimize the quality of a badly shaped element, and results in larger (and more costly) local optimization problems. We have investigated this with the non-differentiable method, and the results can be found in Table I, in which a comparison with the node-by-node algorithm of Reference 2 is included. From the results in Table I it is evident that better qualities can be attained moving several nodes simultaneously ($\mathrm{NL} \geqslant 0$) than moving one at a time (node-by-node), at the expense of implementing more elaborate optimization algorithms.

It is, however, clear that mesh LAKE1 is 'too good' for our comparison purposes. We will pursue the analysis with a perturbed mesh LAKE2, which is obtained by random perturbation of the positions of the internal nodes of the Delaunay mesh LAKE1. The corresponding results can be found in Table II. Notice the bad quality of mesh LAKE2, and how it improves after optimization.

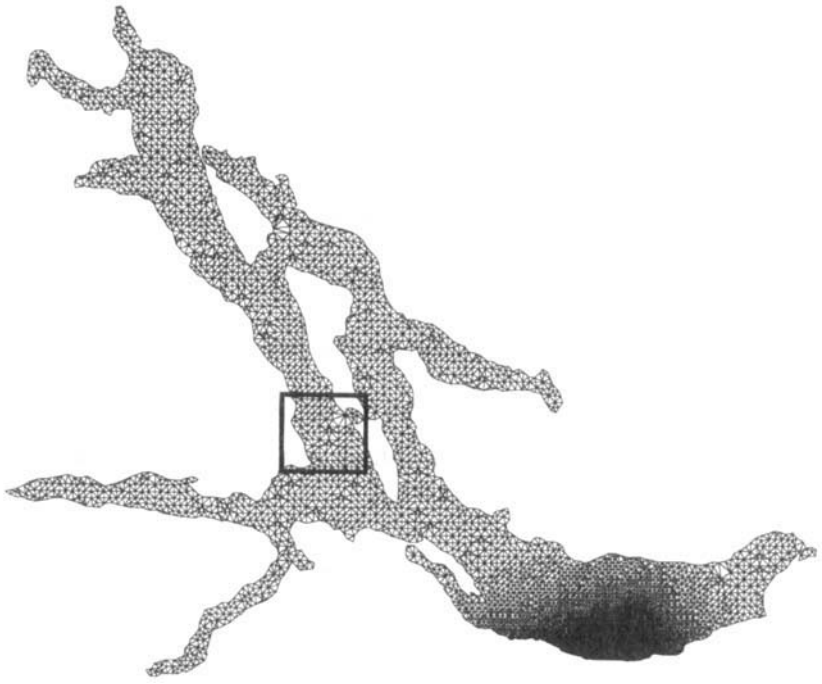

Figure 2. Mesh LAKE1. 11 172 triangles and 6056 nodes. Output from the Delaunay generator. The box indicates the region where the worst element of mesh LAKE2 is. The effect of optimization in this region of the mesh can be seen in Figure 3

Table I. Analysis of the neighbourhood level NL of affected nodes and comparison of our non-differentiable method with the node-by-node method of Reference 2†

MESH LAKE1 11172 elements. 6056 nodes. Initial quality: 0·3445 Min/max angle: 12·04/148·27 Generation method: Delaunay	Non-differentiable method $\varepsilon = 0{\cdot}1$, number of iterations $M = 100$				Node-by-node method
	NL = 0	NL = 1	NL = 2	NL = 3	
Final mesh quality (quality of the worst element)	0·566	0·639	0·643	0·643	0·542
Final minimum angle (degrees)	16·8	20·2	20·7	21·6	15·8
Final maximum angle (degrees)	123·0	120·3	120·3	120·3	123·5
CPU time (s) on SUN IPX	31	132	223	623	13

† Node-by-node methods usually sweep the nodes a few times [14], in this case we have swept the nodes five times

Several conclusions can be drawn from Table II. First of all it is evident that, although it is important to move several nodes at a time to achieve good qualities, it is unnecessary to go beyond a neighbourhood level of 1. This means that the optimization of a mesh can be decoupled, as we have done, in the successive optimization of small clusters of elements, each cluster centered in an element of bad quality. It should not be necessary to optimize the whole mesh simultaneously as proposed in References 10 and 12, and thus 3-D meshes can be dealt with without

Table II. Analysis of the neighbourhood level NL of affected nodes and comparison of our non-differentiable method with the node-by-node method of Reference 2[1]

MESH LAKE2 11172 elements. 6056 nodes. Initial quality: 0·0055 Min/max angle: 0·16/179·0 Generation method: Delaunay	Non-differentiable method $\varepsilon = 0{\cdot}1$, number of iterations $M = 200$ NL = 0	NL = 1	NL = 2	NL = 3	Node-by-node method
Final mesh quality (quality of the worst element)	0·594	0·630	0·643	0·643	0·530
Final minimum angle (degrees)	17·8	19·4	21·2	23·3	15·4
Final maximum angle (degrees)	125·3	121·5	120·2	120·2	126·3
CPU time (s) on SUN IPX	106	245	471	1330	40

Table III. Analysis of the regularization exponent p (differentiable method) and the tolerance ε (non-differentiable method) and comparison of both methods

MESH LAKE2. Perturbed. 11 172 elements. 6056 nodes. Initial quality: 0·0055 Min/max angle: 0·16/179·0 Method: Delaunay	Differentiable method NL = 1, number of iterations $M = 200$ $p = 1$	$p = 3$	$p = 5$	$p = 8$	Non-differentiable method NL = 1, $M = 200$ $\varepsilon = 0{\cdot}01$	$\varepsilon = 0{\cdot}3$
Final mesh quality (quality of the worst element)	0·401	0·535	0·624	0·628	0·630	0·630
Final minimum angle (degrees)	11·4	15·5	19·4	19.4	19·4	19·4
Final maximum angle (degrees)	143·9	121·7	121·7	121·7	121·5	121·5
CPU time (s) on SUN IPX	301	339	371	427	236	2768

excessive computing cost. Also, it should be pointed out that the computing time of our non-differentiable optimization method is quite low (only six times higher for NL = 1) as compared to that of node-by-node methods that are traditionally considered as cheap, and the final qualities attained are significantly higher. We should warn that we arrived at this efficiency after a careful implementation of the method. In particular, the introduction of the tolerance ε (see the remarks in Section 2.3) with the consequent reduction in the dimension of the quadratic problem (5) is essential for the method to be competitive. Also, as in any steepest-descent method, the line-search routine must be programmed with care.

Adopting now NL = 1, we compare the performance of the non-differentiable method with that of the differentiable one; with several values of the regularization exponent p and of the tolerance ε. From the results in Table III it is clear that rather high values of p are needed to make the differentiable method comparable to the non-differentiable one. High values of p render the local optimization problems ill-conditioned, and in particular sensitive to the specific choice of the tolerance $\mathrm{TOL}_{\mathrm{loc}}$. If $\mathrm{TOL}_{\mathrm{loc}}$ is increased from 10^{-5} to 10^{-2} the mesh quality obtained through the differentiable method with $p = 8$ chanes from 0·628 to 0·466. On the other hand, if $\mathrm{TOL}_{\mathrm{loc}}$ is increased to 10^{-2} the quality obtained through the non-differentiable method with $\varepsilon = 0{\cdot}01$ only changes from 0·630 to 0·621. Also notice that the non-differentiable method is faster than the differentiable one for any $\varepsilon < 0{\cdot}1$, and that its results are independent of ε. We usually adopt $\varepsilon = 0{\cdot}05$, but it is sometimes necessary to reduce this value. See Section 4.3 for more details about the selection of ε.

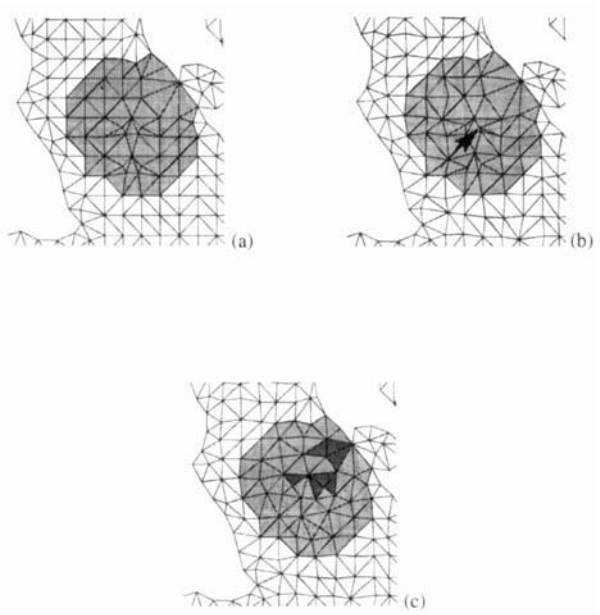

Figure 3. Effect of optimization on nodal positions. Details of the meshes in the region where the worst element of mesh LAKE2 (arrow) lies. (a) Mesh LAKE1; (b) Mesh LAKE2; (c) Mesh LAKE2 after optimizing the cluster of the worst element with NL = 2. The elements in the optimization cluster are shown in gray. In darker gray are shown those elements that share the minimum quality in the optimized cluster

As we have already mentioned, the techniques proposed in this paper are only necessary in three dimensions, as in this case the automatic generators fail to provide valid meshes. The purpose of the examples above has been to illustrate with 2-D meshes (which can be plotted) the performance of our methods and to select appropriate values for the numerical parameters. In Figure 3 we plot the region of mesh LAKE2 where the worst element is, together with the same region of mesh LAKE1 and of mesh LAKE2 after optimizing the cluster of the worst element (with NL = 2). The worst element of mesh LAKE2 is needle-like, marked by the arrow. After optimization, the minimum quality is shared by seven elements (in darker gray, Figure 3(c)), evidencing that the optimum is at a point of lack of differentiability. Notice that the optimized cluster is clearly of higher quality than in the as-generated mesh, LAKE1. In the next section, we turn to the study of a full 3-D geometry and of meshes obtained with state-of-the-art automatic generators.

4.2. A detailed example in 3-D: mesh around an aircraft

We now report an example in three dimensions, namely the construction of a mesh around an aircraft. The surface mesh, consisting of 4914 nodes and 9820 triangles and generated with program TRISURF,[21] can be seen in Figure 4. The Delaunay method developed by Dari[1] was applied to the domain defined by this inner surface and an outer cube with edge length 60 times the wingspan of the aircraft. The resulting mesh (PLANE1) is a good example of the performance of state-of-the-art 3-D mesh generators: From a total of 290 306 tetrahedral elements, there are 139 with qualities under 0·01. The quality of the worst element is *negative*, − 0·39, meaning that during the boundary recovery step of the Delaunay method some overlapping of elements occurred. From our viewpoint, this mesh is just the output of the first part of the generation

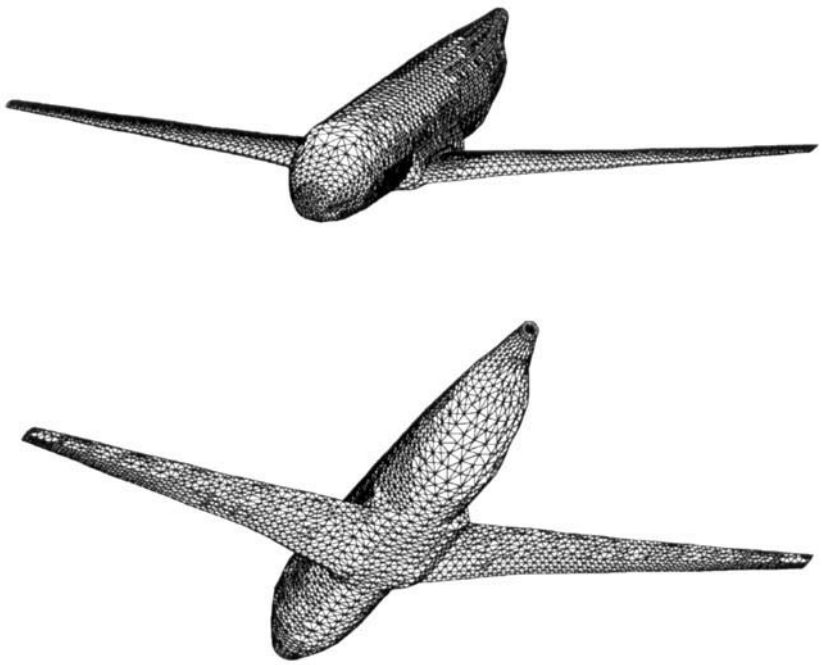

Figure 4. Surface mesh of an aircraft, consisting of 4914 nodes and 9820 triangles

process, the second one being the optimization procedure. Geometrical optimization alone drastically improves the quality of this mesh. One iteration of the non-differentiable method suffices to unfold the mesh and eliminate the overlapping. After 100 iterations with NL = 1 and $\varepsilon = 0{\cdot}05$ mesh PLANE2 was obtained, with a minimum quality of 0·031 (see mesh statistics in Table IV). The evolution of the mesh quality with iterations can be seen in Figure 5. In mesh PLANE1, there are 821 elements with quality under 0·2, while in mesh PLANE2 this number has reduced to 606.

Topological optimization (see Section 3) was applied to mesh PLANE2 (resulting after 100 geometrical optimization iterations). The mesh quality increased as a result of this operation from 0·031 to 0·121, while the number of elements with quality below 0·2 decreased to 32. The mesh was further optimized through 100 + 100 additional geometrical optimization iterations, with a topological optimization in between. The evolution of the mesh quality along the optimization process can be seen in Figures 5(a) and 5(b). The final or 'fully optimized' mesh (mesh PLANE3) has a quite good quality of 0·176, and its statistics can be found in Table IV.

4.3. More examples

The second example we address is a quite small mesh (just 4824 elements and 896 nodes) around a sphere. It serves as an example of optimization of a mesh generated with the advancing-front method. The domain consists of a cylinder with a spherical hole at its center. The cylinder's height is 80 and its radius 40. The radius of the hole is 1. The mesh size far away from the hole was defined as 30, relative to the mesh size at the hole (the volume of an equilateral tetrahedron having the mesh size specified at the hole boundary would be 0·01). The as-generated

Table IV. Effect of optimization on the quality of a mesh around an aircraft.

MESH AROUND AN AIRCRAFT Generation method: Delaunay	As generated (PLANE1)	Geometrically optimized (PLANE2)	Fully optimized (PLANE3)
Number of nodes	50 329	50 329	50 290
Number of elements	290 306	290 306	289 700
Geometrical optimizations performed	0	1	3
Topological optimizations performed	0	0	2
Quality of the worst element	−0·39	0·031	0·176
Minimum angle of the mesh (degrees)	< 0	1·21	5·87
Maximum angle of the mesh (degrees)	> 180	176·6	161·8
Number of elements with quality below 0·2	821	606	3
Cumulative CPU time (s) on SUN IPX needed to construct mesh	1200	1440	2004

Note: Geometrical optimization method is non-differentiable, NL = 1, $\varepsilon = 0{\cdot}05$, and $M = 100$. 'Fully optimized' means a sequence of three geometrical optimizations, with two topological optimizations in between. The minimum/maximum angles of the mesh are calculated considering both edge-to-edge and face-to-face angles within all tetrahedra of the mesh

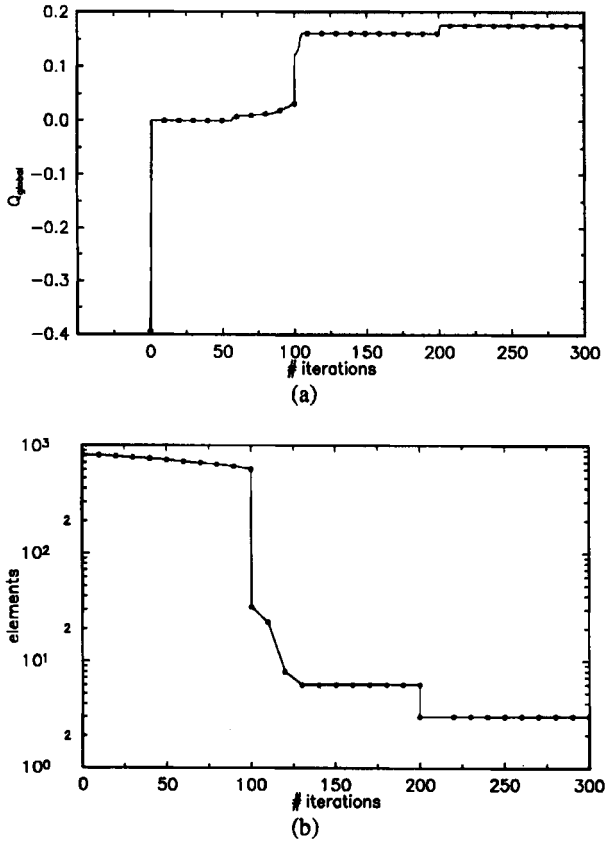

Figure 5. Mesh around an aircraft. Evolution of mesh quality with geometrical optimization iterations. The effect of performing topological optimization after 100 and 200 geometrical iterations is also made evident. (a) Quality of the worst element versus iterations; (b) number of elements with quality below 0·2 versus iterations

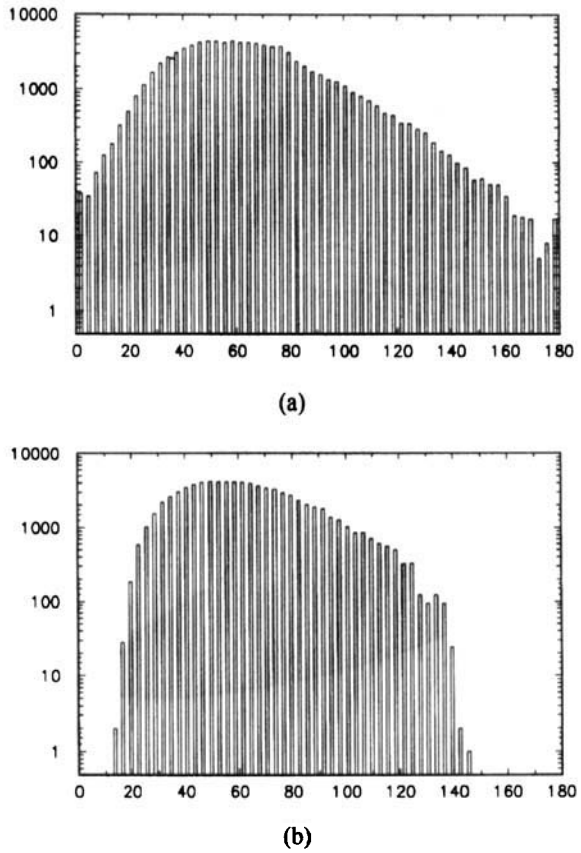

Figure 6. Distribution of angles in mesh CHF1 before (a) and after (b) optimization. The range [0·180] degrees has been divided into 60 three-degree-wide intervals. To each interval the number of angles falling in it has been assigned. Notice how the tails near 0 and 180° are removed by optimization

mesh has minimum/maximum angles of 0·0824°/179·23°, clearly indicating the existence of almost-singular elements. We refer to this mesh as CHF1. The effect of optimization is quite impressive: The quality of the worst element increases from $1{\cdot}97\ 10^{-3}$ to 0·40, and the minimum/maximum dihedral angles improve to 19°/144·7°. The distribution of angles in mesh CHF1 before and after optimization can be found in Figure 6. We are now carrying out an analysis about the impact of such quality improvement on solid mechanics calculations, which will be reported in the near future.

The last example is a mesh around the keel of a competition sailboat. A detail of the surface mesh can be seen in Figure 7. The volume mesh was generated using a Delaunay-based method and consists of 206 549 elements and 37 947 nodes. It will be labeled as KEEL1. The quality of this mesh is strictly zero, with angles of 0° and of 180°. The number of elements with quality below 0·2 is 1339. The evolution of the mesh during optimization can be seen in Figure 8. We performed a sequence of 100 + 100 + 100 geometrical optimization iterations, with topological optimizations in between. The final mesh, KEEL2, has a quality of 0·174, with dihedral angles of more than 7·22 and less than 162·4°. The number of elements with quality below 0·2 in mesh KEEL2 is 2. The beneficial effect of mesh optimization is again evident. Quite surprisingly, to obtain a reasonable performance in CPU time (900 s on a SUN IPX) we had to reduce ε to 0·005. The tuning of ε is straightforward: If it is observed that the dimension of Ξ_ε systematically exceeds

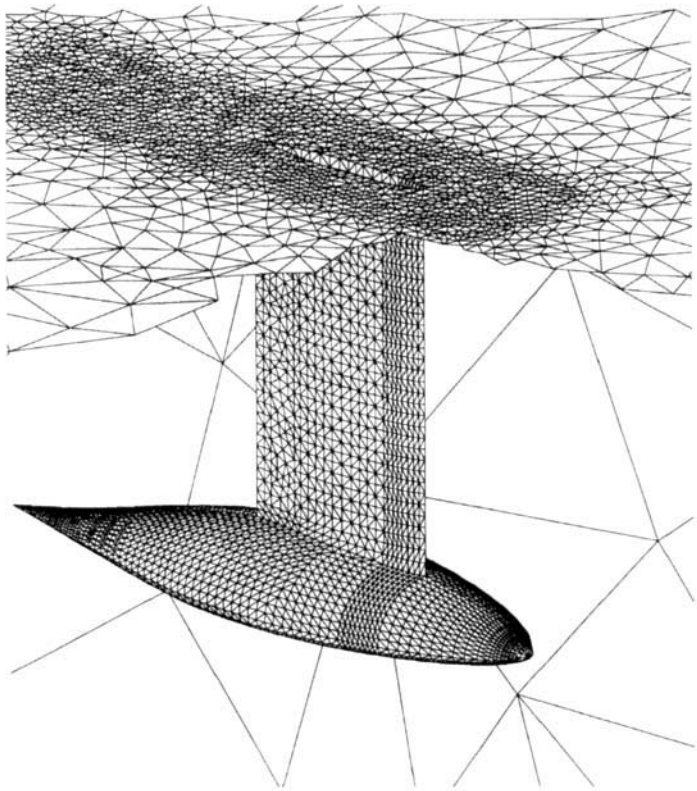

Figure 7. Surface mesh of the keel of a competition sailboat (detail)

10 or 20 during the first global iterations, then reduce ε to make the calculation of the search direction (equation (5)) cheaper, at the expense of increasing the number of *Local Iterations* (cf. Section 2.3).

5. CONCLUSIONS

A new optimization method for unstructured 3-D finite element meshes has been presented. The method is based upon non-differentiable optimization techniques that emphasize the impact on numerical computations of one or very few badly shaped elements within the domain. These low-quality elements are generated by most of the existing automatic mesh generators, and can be avoided applying our method to the as-generated mesh. Several improvements to the raw algorithm have been discussed, which render it computationally effective. The viewpoint of considering 3-D mesh generation as a two-step process beginning with an initial mesh generation followed by optimization, as previously proposed by Dari and Buscaglia,[2] has thus been thoroughly investigated in this work and proved to be a valuable tool for the generation of unstructured meshes in arbitrary domains.

Finally, let us remark that the final mesh obtained through our optimization method indeed depends on the original node distribution and mesh topology. A complete control on the grid quality can only be attained if effective, quality-oriented algorithms are applied along all of the

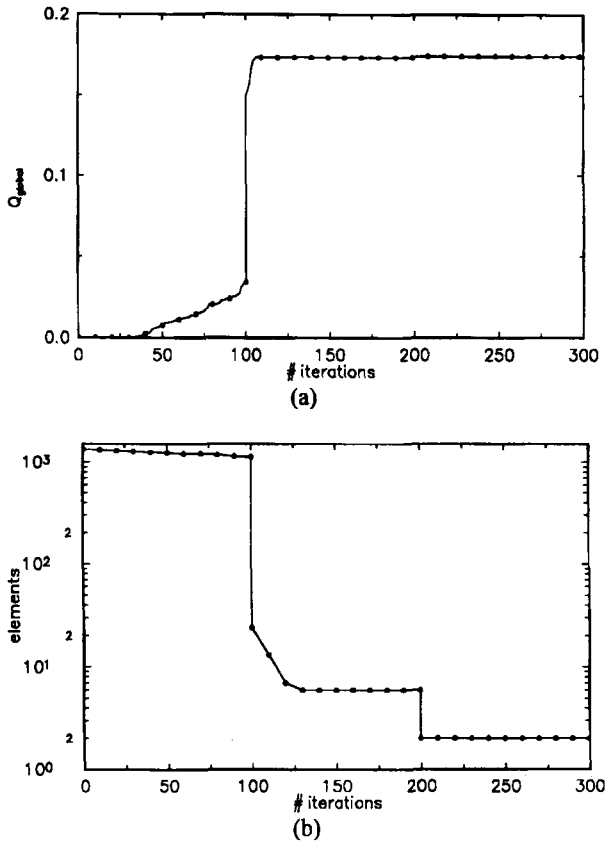

Figure 8. Evolution of the quality of mesh KEEL1 during optimization. Two topological optimizations were performed after 100 and 200 geometrical optimization iterations, respectively: (a) Evolution of the quality of the worst element; (b) number of elements with quality below 0·2

mesh generation process (surface meshing, density definition, node insertion, element creation and optimization).

ACKNOWLEDGMENTS

This work was partially supported by Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), and by Convenio Argentino-Brasileño de Mecánica Computacional (CONICET(Argentina)-CNPq(Brasil)). Our thanks to D. L. Arnica and M. J. Vénere for providing the surface mesh generator TRISURF, and to A. Larreteguy for proofreading.

APPENDIX

Derivatives of the quality function

Two-dimensional case: The quality of an element K is $Q_K = 20{\cdot}7846\, V_K/P_K^2$ and thus

$$\nabla Q_K = 20{\cdot}7846\left(\frac{\nabla V_K}{P_K^2} - \frac{2V_K \nabla P_K}{P_K^3}\right)$$

Let x, y be the two space co-ordinates, let the three nodal points of the triangle be labelled $\mathbf{A} = (x_A, y_A)$, $\mathbf{B} = (x_B, y_B)$ and $\mathbf{C} = (x_C, y_C)$, and let $L_{AB} = \|\mathbf{A} - \mathbf{B}\|$, $L_{AC} = \|\mathbf{A} - \mathbf{C}\|$, and $L_{BC} = \|\mathbf{B} - \mathbf{C}\|$. It is clear that

$$P_K = L_{AB} + L_{AC} + L_{BC}$$

and that

$$V_K = \frac{(x_B - x_A)(y_C - y_A) - (x_C - x_A)(y_B - y_A)}{2}$$

Now, the gradients of these quantities with respect to the co-ordinates of the nodes are

$$\nabla V_K = \begin{pmatrix} \partial V_K/\partial x_A \\ \partial V_K/\partial y_A \\ \partial V_K/\partial x_B \\ \partial V_K/\partial y_B \\ \partial V_K/\partial x_C \\ \partial V_K/\partial x_C \end{pmatrix} = \begin{pmatrix} (y_B - y_C)/2 \\ (x_C - x_B)/2 \\ (y_C - y_A)/2 \\ (x_A - x_C)/2 \\ (y_A - y_B)/2 \\ (x_B - x_A)/2 \end{pmatrix}$$

$$\nabla P_K = \begin{pmatrix} \partial P_K/\partial x_A \\ \partial P_K/\partial y_A \\ \partial P_K/\partial x_B \\ \partial P_K/\partial y_B \\ \partial P_K/\partial x_C \\ \partial P_K/\partial y_C \end{pmatrix} = \begin{pmatrix} (x_A - x_B)/L_{AB} + (x_A - x_C)/L_{AC} \\ (y_A - y_B)/L_{AB} + (y_A - y_C)/L_{AC} \\ (x_B - x_A)/L_{AB} + (x_B - x_C)/L_{BC} \\ (y_A - y_B)/L_{AB} + (y_B - y_C)/L_{BC} \\ (x_C - x_A)/L_{AC} + (x_C - x_B)/L_{BC} \\ (y_C - y_A)/L_{AC} + (y_C - y_B)/L_{BC} \end{pmatrix}$$

Three-dimensional case. The quality of an element K is $Q_K = 1832{\cdot}82\, V_K/P_K^3$ and thus

$$\nabla Q_K = 1832{\cdot}82\left(\frac{\nabla V_K}{P_K^3} - \frac{3V_K \nabla P_K}{P_K^4}\right)$$

Let x, y, z be the three space co-ordinates, let the four nodal points of the tetrahedron be labeled $\mathbf{A} = (x_A, y_A, z_A)$, $\mathbf{B} = (x_B, y_B, z_B)$, $\mathbf{C} = (x_C, y_C, z_C)$ and $\mathbf{D} = (x_D, y_D, z_D)$, and let $L_{AB} = \|\mathbf{A} - \mathbf{B}\|$, $L_{AC} = \|\mathbf{A} - \mathbf{C}\|$, and so on. It is clear that

$$P_K = L_{AB} + L_{AC} + L_{AD} + L_{BC} + L_{BD} + L_{CD}$$

and that

$$V_K = \frac{(\mathbf{D} - \mathbf{A}) \cdot (\mathbf{B} - \mathbf{A}) \wedge (\mathbf{C} - \mathbf{A})}{6}$$

Now, the derivatives of these quantities with respect to the co-ordinates of node A are (permutation of indices allows to calculate the rest of the derivatives)

$$\frac{\partial V_K}{\partial x_A} = \frac{(y_B - y_D)(z_C - z_D) - (y_C - y_D)(z_B - z_D)}{6}$$

$$\frac{\partial V_K}{\partial y_A} = \frac{(z_B - y_D)(x_C - x_D) - (z_C - z_D)(x_B - x_D)}{6}$$

$$\frac{\partial V_K}{\partial z_A} = \frac{(x_B - x_D)(y_C - y_D) - (x_C - x_D)(y_B - y_D)}{6}$$

$$\frac{\partial P_K}{\partial x_A} = \frac{(x_A - x_B)}{L_{AB}} + \frac{x_A - x_C}{L_{AC}} + \frac{x_A - x_D}{L_{AD}}$$

$$\frac{\partial P_K}{\partial y_A} = \frac{(y_A - y_B)}{L_{AB}} + \frac{y_A - y_C}{L_{AC}} + \frac{y_A - y_D}{L_{AD}}$$

$$\frac{\partial P_K}{\partial z_A} = \frac{(z_A - z_B)}{L_{AB}} + \frac{z_A - z_C}{L_{AC}} + \frac{z_A - z_D}{L_{AD}}$$

REFERENCES

1. E. A. Dari, 'Contributions to the automatic triangulation of three-dimensional domains', *Doctoral Thesis*, Instituto Balseiro, 1994 (in Spanish).
2. E. A. Dari and G. C. Buscaglia, 'Mesh optimization: how to obtain good unstructured 3-D finite element meshes with not-so-good mesh generators', *Struct. Optim.*, **8**, 181–188 (1994).
3. S. R. Kennon, 'Supersonic inlet calculations using an upwind finite-volume method on adaptive unstructured grids', *AIAA Paper 89-0113, 27th Aerospace Sci. Meeting*, Reno, USA, 1989.
4. J. Peraire, *Ph.D. Thesis*, Department of Civil Engineering, University College of Swansea, 1986.
5. J. Peraire, K. Morgan and J. Peiró, 'Unstructured mesh methods for CFD', *I.C. Aero Report 90-04*, Imperial College, UK, 1990.
6. J. Peiró, *Ph.D. Thesis*, Department of Civil Engineering, University College of Swansea, 1989.
7. T. E. Tezduyar, S. Aliabadi, M. Behr, A. Johnson and S. Mittal, 'Parallel finite element computation of 3-D flows—computation of moving boundaries and interfaces, and mesh update strategies', *Preprint 93-042*, Army High Performance Computing Res. Ctr., Univ. of Minnesota, USA, 1993.
8. E. A. Dari and G. C. Buscaglia, 'Topics on finite element meshes for problems with moving boundaries', in K. Morgan *et al.* (eds.), *Finite Elements in Fluids, New Trends and Applications*, Pineridge Press, Swansea 1993, pp. 726–735.
9. G. C. Buscaglia and E. A. Dari, 'Improving the quality of three-dimensional finite element meshes: an optimization-based method', *Proc. PACAM'IV 4th Pan American Congress of Applied Mechanics*, 3–6 January 1995, Buenos Aires, Argentina, Vol. 3, 1995 pp. 481–486.
10. J. Cabello, R. Löhner and O.-P. Jacquotte, 'Recent improvements of a variational method for the optimization of directionally stretched unstructured meshes', in N. P. Weatherill *et al.* (eds.), *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, Pineridge Press, Swansea 1994, pp. 163–175.
11. S. R. Kennon and G. S. Dulikravich, 'Generation of computational grids using optimization', *AIAA J.*, **24**, 1069–1073, 1986.
12. A. G. Stamatis and K. D. Papailiou, 'An unstructured grid optimization method', in K. Morgan *et al.* (eds.), *Finite Elements in Fluids, New Trends and Applications*, Pineridge Press, Swansea 1993, pp. 676–685.
13. P. D. Zavattieri, G. C. Buscaglia and E. A. Dari, 'Finite element mesh optimization in three dimensions', to appear in *Latin Amer. Appl. Res.*
14. H. Zhang and J.-Y. Trépanier, 'An algorithm for the optimization of directionally stretched triangulations', *Int. j. numer. methods eng.*, **37**, 1481–1497 (1994).
15. W. H. Frey and D. A. Field, 'Mesh relaxation: a new technique for improving triangulations', *Int. j. numer. methods eng.*, **31**, 1121–1133 (1991).
16. T. Coupez, 'A mesh improvement method for 3D automatic remeshing', in N. P. Weatherill *et al.* (eds.), *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, Pineridge Press, Swansea 1994, pp. 615–625.
17. D. L. Marcum and N. P. Weatherill, 'Unstructured grid generation using iterative point insertion and local reconnection', *AIAA Paper 94-1926, 12th AIAA Applied Aerodynamics Conf.*, Colorado Springs, USA, 1994.
18. E. Polak, 'On the mathematical foundations of nondifferentiable optimization in engineering design', *SIAM Rev.*, **29**, 21–89 (1987).
19. R. Fletcher, *Practical Methods of Optimization, Vol. 2: Constrained Optimization*, Wiley, New York, 1981.
20. M. J. Vénere and E. A. Dari, 'Finite element mesh generator ENREDO, unpublished.
21. M. J. Vénere and D. L. Arnica, 'Surface finite element mesh generation', Presented at *4th Int. Conf. on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, Swansea, Wales, UK, 6–8 April 1994.