A new *Meccano* Technique for Adaptive 3-D Triangulations

J.M. Cascón[1], R. Montenegro[2], J.M. Escobar[2], E. Rodríguez[2] and G. Montero[2]

[1] Department of Mathematics, Faculty of Sciences, University of Salamanca, Spain, `casbar@usal.es`

[2] Institute for Intelligent Systems and Numerical Applications in Engineering, University of Las Palmas de Gran Canaria, Campus Universitario de Tafira, Las Palmas de G.C., Spain, `rafa@dma.ulpgc.es`, `jescobar@dsc.ulpgc.es`, `barrera@dma.ulpgc.es`, `gustavo@dma.ulpgc.es`

This paper introduces a new automatic strategy for adaptive tetrahedral mesh generation. A local refinement/derefinement algorithm for nested triangulations and a simultaneous untangling and smoothing procedure are the main involved techniques. The mesh generator is applied to 3-D complex domains whose boundaries are projectable on external faces of a coarse object *meccano* composed of cuboid pieces. The domain surfaces must be given by a mapping between *meccano* surfaces and object boundary. This mapping can be defined by analytical or discrete functions. At present we have fixed mappings with orthogonal, cylindrical and radial projections, but any other one-to-one projection may be considered. The mesh generator starts from a coarse tetrahedral mesh which is automatically obtained by the subdivision of each hexahedra, of a *meccano* hexahedral mesh, into six tetrahedra. The main idea is to construct a sequence of nested meshes by refining only those tetrahedra which have a face on the *meccano* boundary. The virtual projection of *meccano* external faces defines a valid triangulation on the domain boundary. Then a 3-D local refinement/derefinement is carried out such that the approximation of domain surfaces verifies a given precision. Once this objective is reached, those nodes placed on the *meccano* boundary are really projected on their corresponding true boundary, and inner nodes are relocated using a suitable mapping. As the mesh topology is kept during node movement, poor quality or even inverted elements could appear in the resulting mesh. For this reason, we finally apply a mesh optimization procedure. The efficiency of the proposed technique is shown with several applications to complex objects.

1 Introduction

In finite element simulation in engineering problems, it is crucial to automatically adapt the three-dimensional discretization to geometry and to solution. Many authors have devoted great efforts in the past to solve this problem in different ways [3, 11, 13, 30], but automatic 3-D mesh generation is still an open problem. Generally, as the complexity of the problem increases (domain geometry and model), the methods for approximating the solution are more complicated. At present, it is well known that most mesh generator are based on Delaunay triangulation and advancing front technique.

In the last few years, we have developed a tetrahedral mesh generator that approximates the orography of complex terrains with a given precision [21, 22]. To do so, we only use digital terrain information. The generated mesh have been applied for numerical simulation of environmental phenomena, such as wind field adjustment [26], fire propagation or atmospheric pollution [25]. The following procedures were mainly involved in this former automatic mesh generator: a Delaunay triangulation method [5, 12], a 2-D refinement/derefinement algorithm [8], based on the 4-T Rivara's algorithm [27], and a simultaneous untangling and smoothing algorithm [6]. Many ideas were introduced in this mesh generator and they have been taken into account for developing the new mesh generator proposed in this paper.

On the other hand, local adaptive refinement strategies are employed to adapt the mesh to singularities of numerical solution. These adaptive methods usually involve remeshing or nested refinement [14, 17, 18, 28]. Another interesting idea is to adapt simultaneously the model and the discretization in different regions of the domain. A perspective about adaptive modeling and meshing is studied in [4]. The main objective of all these adaptive techniques is to achieve a good approximation of the *real* solution with a minimal user intervention and a low computational cost. For this purpose, the mesh element quality is also an essential aspect for the efficiency and numerical behaviour of finite element method. The element quality measure should be understood depending on the isotropic or anisotropic character of the numerical solution.

In this paper we present new ideas and applications of an innovative tetrahedral mesh generator which was introduced in [24]. This automatic mesh generation strategy uses no Delaunay triangulation, nor advancing front technique, and it simplifies the geometrical discretization problem in particular cases. The main idea of the new mesh generator is to combine a local refinement/derefinement algorithm for 3-D nested triangulations [17] and a simultaneous untangling and smoothing procedure [6]. 3-D complex domains, which surfaces can be mapped from a *meccano* face to object boundary, are discretized by the mesh generator. Resulting adaptive meshes have an appropriate quality for finite element applications.

At present, this idea has been implemented in ALBERTA code [1, 29]. This software can be used for solving several types of 1-D, 2-D or 3-D problems with adaptive finite elements. The local refinement and derefinement can be done by evaluating an error indicator for each element of the mesh and it is based on element bisection. To be more specific, the newest vertex bisection method is implemented for 2-D triangulations [20]. Actually, ALBERTA has implemented an efficient data structure and adaption for 3-D domains which can be decomposed into hexahedral elements as regular as possible. Each hexahedron is subdivided into six tetrahedra by constructing a main diagonal and its projections on its faces, see Figure 1 (a). The local bisection of the resulting tetrahedra is recursively carried out by using ideas of the longest edge [27] and the newest vertex bisection methods. Details about the local refinement technique implemented in ALBERTA for two and three dimensions can be analyzed in [17, 20]. This strategy works very efficiently for initial meshes with a particular topology and high-quality elements (obtained by subdivision of regular quadrilateral or hexahedral elements). In these cases, the degeneration of the resulting 2-D or 3-D triangulations after successive refinements is avoided. The restriction on the initial element shapes and mesh connectivities makes necessary to develop a particular mesh generator for ALBERTA. In this paper we summarize the main ideas introduced for this purpose. Obviously, all these techniques could be applied for generating meshes with other types of codes. Besides, these ideas could be combined with other type of local refinement/derefinement algorithms for tetrahedral meshes [14, 18, 28].

In the following section we present a description of the main stages of the new mesh generation procedure. In section 3 we show test problems and practical applications which illustrate the efficiency of this strategy. Finally, conclusions and future research are presented in section 4.

2 Description of the Mesh Generator

In this section, we present the main ideas which have been introduced in the mesh generation procedure. In section 2.1 and 2.2, we start with the definition of the domain and its subdivision in an initial 3-D triangulation that verifies the restrictions imposed in ALBERTA. In section 2.3, we continue with the presentation of different strategies to obtain an adapted mesh which can approximate the boundaries of the domain within a given precision. We construct a mesh of the domain by projecting the boundary nodes from a *meccano* plane face to the true boundary surface and by relocating the inner nodes. These two steps are summarized in section 2.4 and 2.5, respectively. Finally, in section 2.6 we present a procedure to optimize the resulting mesh.

2.1 Object *Meccano*

In order to understand the idea of the proposed mesh generator, it is convenient to first consider a domain whose boundary can be projected on the faces of a cube. A second simple case is to consider a cuboid instead of a cube. We can generalize the previous cases with a *meccano* constructed by

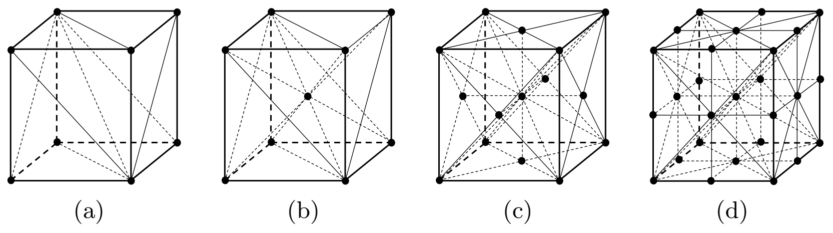


Fig. 1. Refinement of a cube by using Kossaczky's algorithm: (a) cube subdivision into six tetrahedra, (b) bisection of all tetrahedra by inserting a new node in the cube main diagonal, (c) new nodes in diagonals of cube faces and (d) global refinement with new nodes in cube edges

cuboid pieces. In this last case, we suppose that an automatic decomposition of the *meccano* into cubes (or hexahedra) can be carried out. At present, we have implemented these cases in ALBERTA and we have defined the *meccano* input data as connected cuboids, such that the boundary of the object is obtained by a one-to-one projection (or mapping) from the boundary faces of the *meccano*.

2.2 Coarse Tetrahedral Mesh of the *Meccano*

Once the *meccano* decomposition into cubes is done, we build an initial coarse tetrahedral mesh by splitting all cubes into six tetrahedra [17]. For this purpose, it is necessary to define a main diagonal on each cube and the projections on its faces, see Figure 1(a). In order to get a conforming tetrahedral mesh, all cubes are subdivided in the same way maintaining compatibility between the diagonal of their faces. The resulting initial mesh τ_1 can be introduced in ALBERTA since it verifies the imposed restrictions about topology and structure. The user can introduce in the code the necessary number of recursive global bisections [17] for fixing a uniform element size in the whole initial mesh. In Figures 1 (b), (c) and (d) are presented three consecutive global bisections in a cube. The resulting mesh of Figure 1(d) contains 8 similar cubes to the one represented in Figure 1(a).

If we consider a *meccano* composed of hexahedra pieces instead of cuboids, a similar technique can be applied. In this case, the recursive local refinement [17] may produce poor quality elements depending on the initial mesh quality. The minimum quality of refined meshes is function of the initial mesh quality. A study about this aspect can be seen in [19, 31]. In this paper, as a first approach, we have used a decomposition of the object *meccano* into cuboids.

2.3 Local Refined Mesh of the *Meccano*

The next step of the mesh generator includes a recursive adaptive local refinement strategy of those tetrahedra with a face placed on a boundary face of the initial coarse mesh. The refinement process is done in such a way that the true surfaces are approximated with a linear piece-wise interpolation within a given precision. That is, we look for an adaptive triangulation on the *meccano* boundary faces, such that the resulting triangulation after node projection (or mapping) on the object true boundary is a good approximation of this boundary. The user has to introduce as input data a parameter ε that defines the maximum separation allowed between the linear piece-wise interpolation and the true surface. We remark that the true surface may be given by an analytical or a discrete function, such that each point of *meccano* boundary faces corresponds only to one point on the true surface. We propose two different strategies for reaching our objective.

The first one consists on a simple method. We construct a sequence of tetrahedral nested meshes by recursive bisection of all tetrahedra which contain a face located on the *meccano* boundary faces; see Figure 1. The number of bisections is determined by the user as a function of the desired resolution of the true surface. So, we have a uniform distribution of nodes on these *meccano* faces. Once all these nodes are *virtually* projected on the true surface, a generalization of the derefinement criterion developed in [8], with a given derefinement parameter ε, defines different adaptive triangulations for each *meccano* face. We remark that the generalized derefinement criterion fixes which nodes, placed on *meccano* faces, can not be eliminated in the derefinement process in order to obtain a good approach of the true surface.

To illustrate the new derefinement criterion, we have represented the projection of a *meccano* external node P on its true position P' placed on the object surface Σ, see Figure 2(a). We consider that node P is located in the middle point of its *surrounding edge ac*. The insertion of node P produces the bisection of the *father* triangle abc and, consequently, the bisection of the *father* tetrahedron (T) which contains the face abc. This tetrahedron is subdivided in two *sons* (T_1 and T_2) which are defined by the partition of face abc into the two faces abP and bcP, respectively. If we project nodes a, b and c on surface Σ, we get a planar approximation given by the triangle $a'b'c'$, but if we consider the insertion of node P, we obtain an improved approximation given by the triangles $a'b'P'$ and $b'c'P'$. Therefore, we have introduced the following generalized derefinement condition:

> Node P could be eliminated if the volume of *virtual* tetrahedron $a'b'c'P'$ is less than ε (although perhaps node P can not be finally removed due to conformity reasons). On the other hand, nodes belonging to the coarse initial mesh are not removed from the sequence of nested meshes.

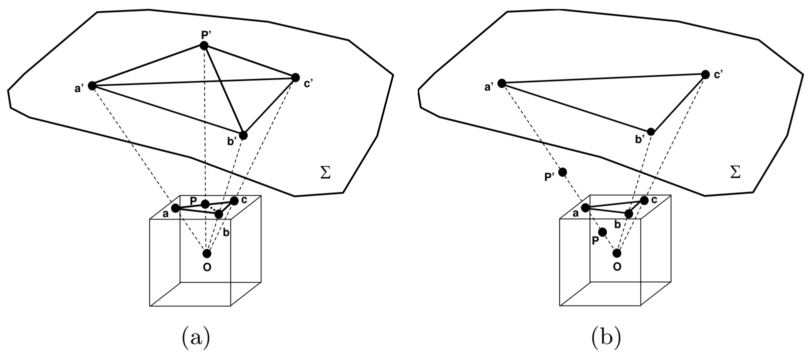


Fig. 2. Node mapping from a cube to real domain: (a) transformation of an external node P and (b) of an inner node P

As the derefinement criterion in ALBERTA is associated to elements, we mark for derefining all tetrahedra containing a node which can be eliminated (i.e., if node P can be eliminated, we mark tetrahedra T_1 and T_2 for derefining). In particular, we make a loop on tetrahedra during the derefinement process from the penultimate level of the sequence to the coarse initial mesh. We only analyze tetrahedra with two sons, such that if the node introduced in their *father*'s bisection verifies the derefinement condition, then we mark their two *sons* for derefining.

The second strategy only works with the local refinement algorithm. In this case, the refinement criterion for tetrahedron T should be:

> Node P must be inserted if the volume of *virtual* tetrahedron $a'b'c'Q'$, being Q' the projection on Σ of any point Q belonging to face abc, is greater or equal than ε. Then, tetrahedron T should be subdivided in T_1 and T_2.

The first strategy is simpler, but it could lead to problems with memory requirements if the number of tetrahedra is very high before applying the derefinement algorithm. Suppose for example that we have the information of a surface defined on a meccano face by a discrete function on a very fine grid. In order to capture the surface geometry without losing information, we need to construct a global refined mesh as fine as the grid associated to the discrete function. Nevertheless, the user could control the number of recursive bisections and the maximum lost volume for the resulting object discretization.

On the other hand, the problem of the second strategy is to determine if a face placed on *meccano* boundary must be subdivided attending to the approximation of the true surface. This analysis must be done every time that a face is subdivided into its two *son* faces. Suppose for example that the true surface is given by a discrete function. Then, the subdivision criterion should stop for a particular face when all the surface discretization points, defined on this face, have been analyzed and all of them verify the approximation criterion. So, this second strategy has the inconvenient that each surface discretization point should be studied many times and, therefore, it generally implies a higher computational cost than the first strategy.

2.4 External Node Mapping on Object Boundary

Although ALBERTA has already implemented a node projection on a given boundary surface during the bisection process, it has two important restrictions: nodes belonging to the initial mesh are not projected and inverted elements could appear in case of projecting new nodes on complex surfaces (i.e. non-convex object). In the latter case, the code does not work properly, since it is only prepared to manage *valid* meshes.

For this reason, a new strategy must be developed in the mesh generator. The projection (or mapping) is really done once we have defined the local refined mesh by using one of the two methods proposed in the previous section. Then, the nodes placed on the *meccano* faces are projected (or mapped) on their corresponding true surfaces, maintaining the position of the inner nodes of the *meccano* triangulation. We have remarked that any one-to-one projection can be defined: orthogonal, spherical, cylindrical, etc. For example, spherical projection from point O has been used in Figure 2.

After this process, we obtain a valid triangulation of the domain boundary, but it could appear a tangled tetrahedral mesh. Inner nodes of the *meccano* could be located now even outside the domain. Thus, an optimization of the mesh is necessary. Although the final optimized mesh does not depend on the initial position of the inner nodes, it is better for the optimization algorithm to start from a mesh with a quality as good as possible. Therefore, we propose to relocate in a reasonable position the inner nodes of the *meccano* before the mesh optimization.

2.5 Relocation of Inner Nodes

There would be several strategies for defining an appropriate position for each inner node of the domain. An acceptable procedure is to modify their relative position as a function of the distance between boundary surfaces before and after their projections. This relocation is done attending to proportional criteria along the corresponding projection line. For example, relocation of inner node P in its new position P', such that $OP' = OP \times Oa' \ / \ Oa$, is represented in Figure 2(b). Although this node movement does not solve the tangle mesh problem, it normally makes it decrease. That is, the number of resulting inverted elements is less and the mean quality of valid elements is greater.

2.6 Object Mesh Optimization: Untangling and Smoothing

An efficient procedure is necessary to optimize the current mesh. This process must be able to smooth and untangle the mesh and it is crucial in the proposed mesh generator.

The most usual techniques to improve the quality of a *valid* mesh, that is, one that does not have inverted elements, are based upon local smoothing. In short, these techniques consist of finding the new positions that the mesh nodes must hold, in such a way that they optimize an objective function. Such a function is based on a certain measurement of the quality of the *local submesh,* $N(v)$, formed by the set of tetrahedra connected to the *free node* v. As it is a local optimization process, we can not guarantee that the final mesh is globally optimum. Nevertheless, after repeating this process several times for all the nodes of the current mesh, quite satisfactory results can be achieved. Usually, objective functions are appropriate to improve the quality of a valid mesh, but they do not work properly when there are inverted elements. This is because they present singularities (barriers) when any tetrahedron of $N(v)$ changes the sign of its Jacobian determinant. To avoid this problem we can proceed as Freitag et al in [9, 10], where an optimization method consisting of two stages is proposed. In the first one, the possible inverted elements are untangled by an algorithm that maximises their negative Jacobian determinants [9]; in the second, the resulting mesh from the first stage is smoothed using another objective function based on a quality metric of the tetrahedra of $N(v)$ [10]. After the untangling procedure, the mesh has a very poor quality because the technique has no motivation to create good-quality elements. As remarked in [10], it is not possible to apply a gradient-based algorithm to optimize the objective function because it is not continuous all over $\mathbb{R}^3$, making it necessary to use other non-standard approaches.

We have proposed an alternative to this procedure [6], so the untangling and smoothing are carried out in the same stage. For this purpose, we use a suitable modification of the objective function such that it is regular all over $\mathbb{R}^3$. When a feasible region (subset of $\mathbb{R}^3$ where v could be placed, being $N(v)$ a valid submesh) exists, the minima of the original and modified objective functions are very close and, when this region does not exist, the minimum of the modified objective function is located in such a way that it tends to untangle $N(v)$. The latter occurs, for example, when the fixed boundary of $N(v)$ is tangled. With this approach, we can use any standard and efficient unconstrained optimization method to find the minimum of the modified objective function, see for example [2].

In this work we have applied, for simultaneous smoothing and untangling of the mesh by moving their inner nodes, the proposed modification [6] to one objective function derived from an *algebraic mesh quality metric* studied in [16], but it would also be possible to apply it to other objective functions which have barriers like those presented in [15].

Besides, a smoothing of the boundary surface triangulation could be applied before the movement of inner nodes of the domain by using the new procedure presented in [7] and [23]. This surface triangulation smoothing technique is also based on a vertex repositioning defined by the minimization of a suitable objective function. The original problem on the surface is transformed into a two-dimensional one on the *parametric space*. In our case, the parametric space is a plane, chosen in terms of the local mesh, in such a way that this mesh can be optimally projected performing a *valid* mesh, that is, without *inverted* elements.

3 Test Examples

The performance of our new mesh generator is shown in the following applications. The first example corresponding to the discretizacion of a *16th IMR conmemoration* glass and the second one is an Earth-Atmosphere system.

3.1 16th IMR conmemoration glass

We consider the discretization of a glass filled with a liquid, we include a legend and a face profile on the external face of the glass (see Figure 3(d)). The input data consists on a *meccano* of five cuboids for the glass and one cuboid for the liquid, see Figure 3(a). This *meccano* is included in a parallelepiped which dimensions are $5 \times 5 \times 8$. The mesh generation strategy automatically defines the boundary between the two materials and gets a good mesh adaption to the geometrical domain characteristics. A one-to-one projection between the *meccano* and the object is defined by a cylindrical projection for the vertical faces and an orthogonal one for the horizontal faces. We modify the original projection in order to include the legend *16th IMR* and the 3-D face profile of Igea on the external wall of the glass. The profile of Igea has been obtained from its 3D surface triangulation provided at *http://www.cyberware.com/*.

The *meccano* is splitted into a 3-D triangulation of 1146 tetrahedra and 320 nodes. We apply 6 recursive bisections on all tetrahedra which have a face placed on the *meccano* boundary or on the material interface. Additionally, we bisect 12 times all tetrahedra which have a face on the side of *meccano* where the legend and the face profile are included. This mesh contains 1189989 nodes and 5426256 tetrahedra. The derefinement parameter is fixed to $\varepsilon = 5\ 10^{-6}$. The resulting adapted mesh contains 24258 nodes and 112388 tetrahedra and it is shown in Figure 3(b). Note that the refinement/derefinement algorithm captures the letters and the face profile. The projection of this *meccano* surface triangulation on the true surface produces a 3-D tangled mesh with 48775 inverted elements, see Figure 4. Relocation of inner nodes reduces the number of inverted elements to 2648. After ten iterations of the optimization procedure, the mesh generator converts the tangled mesh into the one presented in

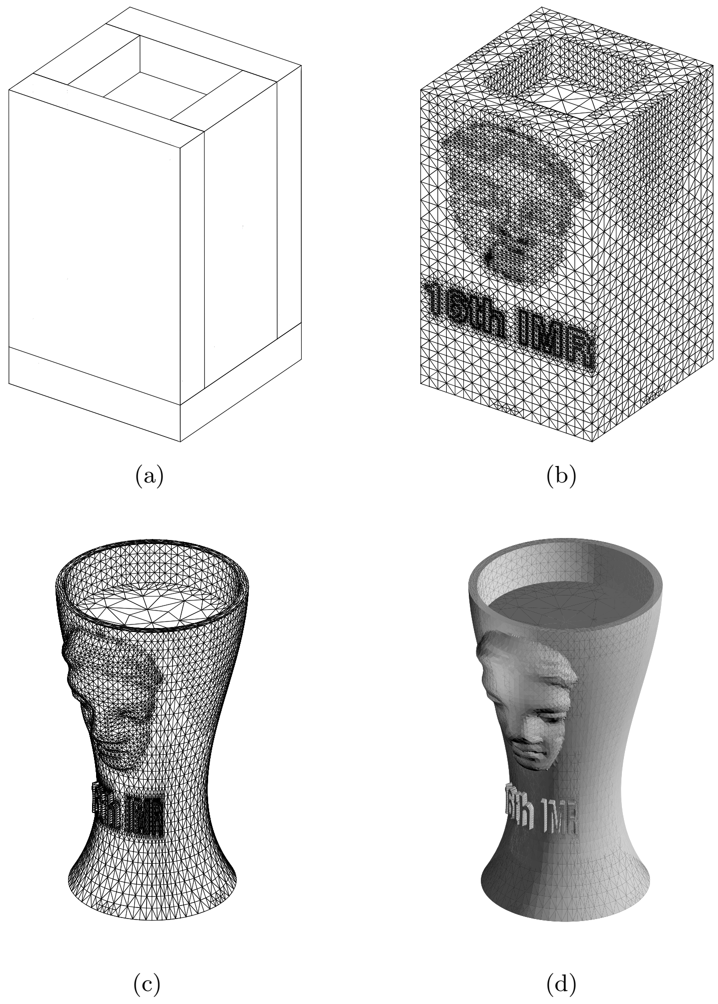


Fig. 3. Main stages of the mesh generator for the *16*th *IMR* glass: (a) *meccano*, (b) mesh adaption after applying the refinement/derefinement procedure, (c) resulting optimized mesh, (d) texture map of the resulting mesh

Figure 3(c). The mesh quality is improved to a minimum value of 0.02 and an average $\overline{q}_{\kappa} = 0.55$. The quality curves for the initial tangled mesh and the final optimized triangulation are shown in Figure 5. There are a few elements (less than one hundred) with a poor quality (less than 0.1). This is due to rough profile of the letters, and could be avoided with a smoother definition.

The CPU time for constructing the initial mesh is approximately 1 minute and for its optimization is less than 3 minutes on a laptop with Intel proces-

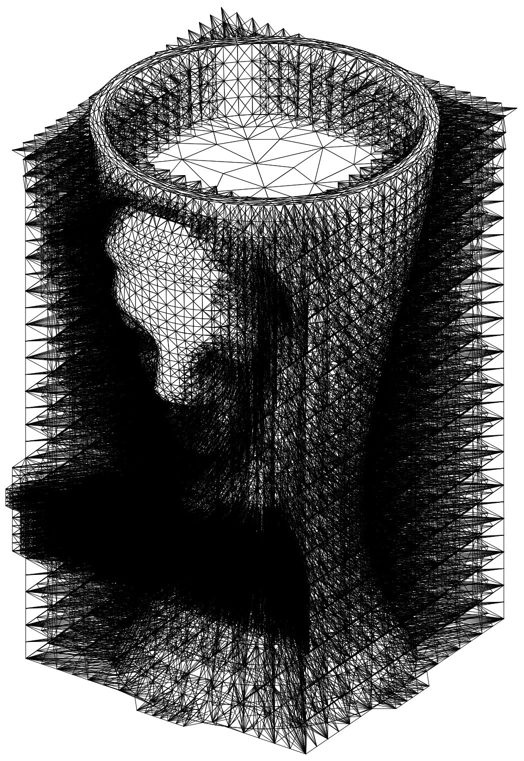


Fig. 4. Tangled mesh with 48775 inverted elements after the projection on glass surface and before relocation of inner nodes. The optimization process is able to convert this mesh in a valid one

sor 1.6 GHz, 1 Gb RAM memory on Linux Fedore Core 4 system. In order to show the efficiency of our method running on a workstation, we report in Table 1 data (number of nodes, tetrahedra, partial CPU times) of several meshes corresponding to the same example with different values of derefinement parameter. Note that the mesh optimization dominates the procedure.

The crucial step of the mesh generation process is the optimization algorithm. As a measurement of its capability notice that the procedure is able to convert the tangled mesh of Figure 4, without any relocation, into the mesh represented in Figure 3(c). However, it requires some more iterations of the optimization algorithm, and the corresponding increasing of CPU time. In Figure 6, we can appreciate the effect of the optimization process into the glass.

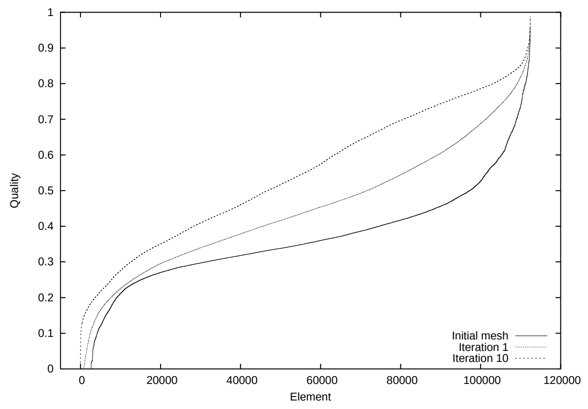


Fig. 5. Quality curves for the initial tangled mesh, an intermidiate one, and the optimized triangulation after ten iterations for the *16*th *IMR* glass

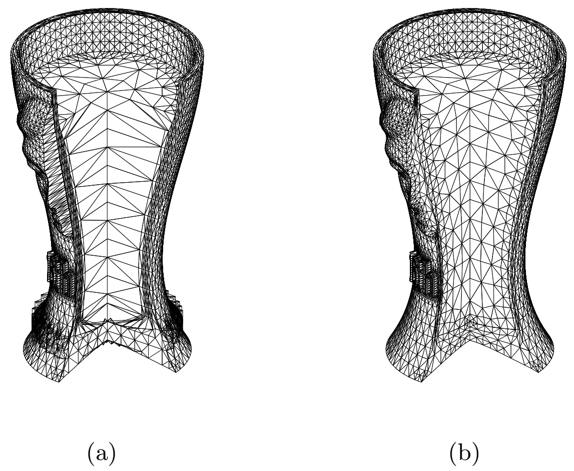


Fig. 6. Cross sections of *16*th *IMR* glass: (a) After relocation of inner nodes, (b) after application of the optimization process

Finally, we remark that the mesh is made up of two different material: glass and liquid. The mesh generator automatically conserves the material interface and generates a conforming mesh between both materials.

3.2 Earth-Atmosphere

We have applied our technique to construct a 3-D triangulation of the Earth and its atmosphere. To define the topography we have used GTOPO30

ϵ	Nodes	Tetrahedra	CPU time (seconds)			
			Meccano Mesh	Projection/Relocation	Untangling	Smoothing
5 10^{-6}	24.258	112.388	54.3	0.3	35.4	39.5
10^{-6}	51.756	235.360	57.8	0.7	119.8	64.2
10^{-7}	94.488	419.632	61.3	1.3	393.3	84.3
10^{-8}	166.964	732.104	64.1	2.2	592.9	201.8

Table 1. Data corresponding to meshes of 16th-IMR glass example for several values of derefinement parameter. These experiments have been done on a Dell Precison 960, with two Intel Xeon doble kernel processor, 3.2 GHz, 64 bits and 8 Gb RAM, on a Red Hat Enterprise Linux WS v.4 system and using the compiler gcc v.3.4.6

(*http://edc.usgs.gov/products/elevation/gtopo30/gtopo30.html*). GTOPO30 is a global digital elevation model (DEM) with a horizontal grid spacing of 30 arc seconds (approximately 1 kilometer).

In this case, the *meccano* is made up of six cuboids for the atmosphere and one cuboid for the Earth. The union of all of them is a cube of dimension $8000 \times 8000 \times 8000$ kilometers. We use a radial projection to generate the topography of the surface of the Earth from GTOPO30, and to project the external faces of the *meccano* on a sphere of ratio 8000 kilometers.

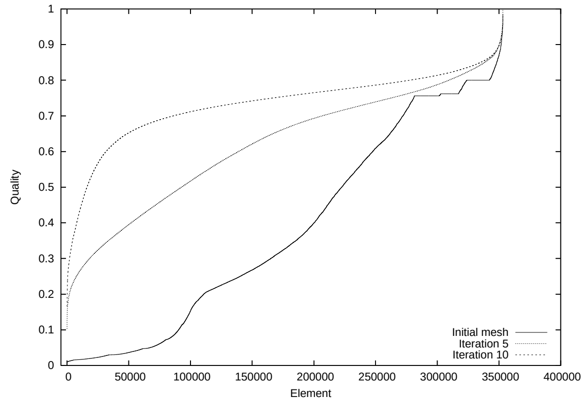
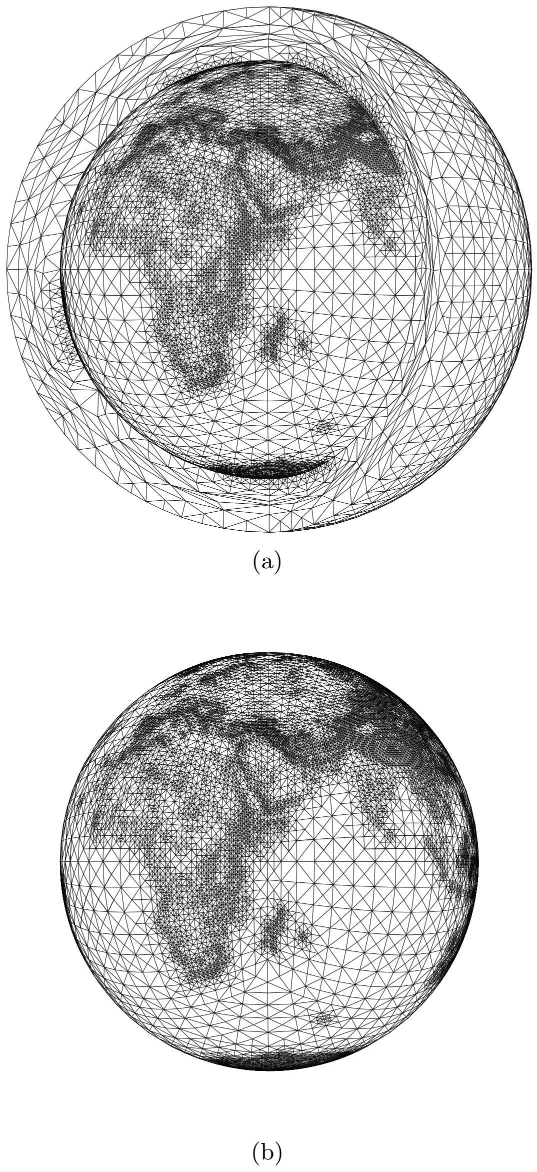


Fig. 7. Quality curves for the initial mesh, intermediate one and the optimized triangulation after ten iterations for the Earth-Atmosphere

The *meccano* is splitted into a 3-D triangulation of 3072 tetrahedra and 729 nodes. We apply 15 recursive bisections on all tetrahedra which have a face placed on the *meccano* boundary or on the interface between the cuboids representing the Earth and the atmosphere. This mesh contains 1119939 nodes and 5053392 tetrahedra. In order to obtain a final mesh with a good repre-



(a)

(b)

Fig. 8. Two views of the final mesh of the Earth and atmosphere

sentation of land areas but with an acceptable number of nodes, we used in this example different values of the derefinement parameter ε. The value of ε on land areas is fixed to 2.5 10^2 and on ocean areas to 5 10^5. The resulting mesh has 64901 nodes and 353142 tetrahedra; see Figure 8.

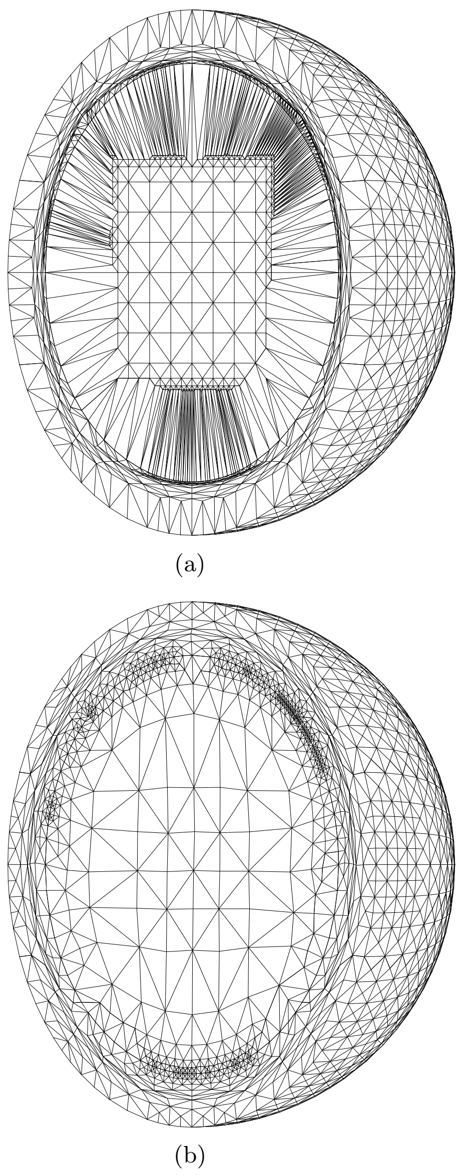


Fig. 9. Cross section of the Earth and atmosphere before (a) and after (b) the application of the mesh optimization process. The refined region in the interface corresponds to land areas

The relocation of inner nodes is enough to obtain a untangled mesh, however its quality is very poor. After ten iterations, the optimization procedure improves the value of the minimum quality from 0.01 to 0.17 and its average from $\overline{q}_\kappa = 0.38$ to $\overline{q}_\kappa = 0.73$; see Figure 7. The efficiency of this mesh optimization can be appreciated in Figure 9 where we represent two cross sections before (a) and after (b) its application.

The CPU time for constructing the mesh is approximately 10 minutes on a laptop, with Intel processor 1.6 GHz, 1 Gb RAM memory on Linux Fedore Core 4 system. In Table 2 we show data (number of nodes, tetrahedra, partial CPU times) of several meshes corresponding to different values of derefinement parameter.

ϵ				CPU time (seconds)			
Land	Sea	Nodes	Tetrahedra	Meccano Mesh	Projection/Relocation	Untangling	Smoothing
250	$5\ 10^4$	64.901	353.142	57.3	1.3	0.0	281.0
50	10^5	105.013	563.092	63.1	2.1	0.0	427.7
5	10^4	165.704	863.818	63.6	3.1	0.0	637.0
0.5	10^3	319.851	1.601.800	67.7	5.8	0.0	1196.3

Table 2. Data corresponding to meshes of Earth-Atmosphere example for several values of derefinement parameter. These experiments have been done at the same workstation specified in caption of Table 1

4 Conclusions and Future Research

The proposed mesh generator is an efficient method for creating tetrahedral meshes on domains with boundary faces projectable on a *meccano* boundary. At present, the user has to define the meccano associated to the object and projections between meccano and object surfaces. Once these aspects are fixed, the mesh generation procedure is fully automatic and has a low computational cost. In future works, we will develop a special CAD package for minimizing the user intervention. Specifically, object surface patches should be defined by using meccano surfaces as parametric spaces.

The main ideas presented in this paper for automatic mesh generation, which have been implemented in ALBERTA, could be used for different codes which work with other tetrahedral or hexahedral local refinement/derefinement algorithms. Taking into account these ideas, complex domains could be meshed by decomposing its *outline* into a set of connected cubes or hexahedra. In future works, new types of pieces and connections could be considered for constructing the *meccano*.

Although this procedure is at present limited in applicability for high complex geometries, it results in a very efficient approach for the problems that fall within the mentioned class. The mesh generation technique is based on sub-processes (subdivision, projection, optimization) which are not in themselves new, but the overall integration using a simple shape as starting point is an original contribution of this paper and it has some obvious performance advantages. Besides, we have introduced a generalized derefinement condition for a simple approximation of surfaces. On the other hand, the new mesh generation strategy automatically fixes the boundary between materials and gets a good mesh adaption to the geometrical domain characteristics.

Acknowledgments

This work has been supported by the Spanish Government and FEDER, grant contracts: CGL2004-06171-C03-03/CLI and CGL2004-06171-C03-02/CLI. Besides, we thanks to the authors of ALBERTA [29] for the code availability in internet [1] and for their suggestions.

References

1. ALBERTA - An Adaptive Hierarchical Finite Element Toolbox, http://www.alberta-fem.de/
2. Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear programing: theory and algorithms. John Wiley and Sons Inc, New York
3. Carey GF (1997) Computational grids: generation, adaptation, and solution strategies. Taylor & Francis, Washington
4. Carey GF (2006) A perspective on adaptive modeling and meshing (AM&M). Comput Meth Appl Mech Eng 195:214–235
5. Escobar JM, Montenegro R (1996) Several aspects of three-dimensional Delaunay triangulation. Adv Eng Soft 27:27–39
6. Escobar JM, Rodríguez E, Montenegro R, Montero G, González-Yuste JM (2003) Simultaneous untangling and smoothing of tetrahedral meshes. Comput Meth Appl Mech Eng 192:2775–2787
7. Escobar JM, Montero G, Montenegro R, E. Rodríguez E (2006) An algebraic method for smoothing surface triangulations on a local parametric space. Int J Num Meth Eng 66:740–760
8. Ferragut F, Montenegro R, Plaza A (1994) Efficient refinement/derefinement algorithm of nested meshes to solve evolution problems. Comm Num Meth Eng 10:403–412
9. Freitag LA, Plassmann P (2000) Local optimization-based simplicial mesh untangling and improvement. Int J Num Meth Eng 49:109–125
10. Freitag LA, Knupp PM (2002) Tetrahedral mesh improvement via optimization of the element condition number. Int J Num Meth Eng 53:1377–1391
11. Frey PJ, George PL (2000) Mesh generation. Hermes Sci Publishing, Oxford
12. George PL, Hecht F, Saltel E (1991) Automatic mesh generation with specified boundary. Comput Meth Appl Mech Eng 92:269–288
13. George PL, Borouchaki H (1998) Delaunay triangulation and meshing: application to finite elements. Editions Hermes, Paris

14. González-Yuste JM, Montenegro R, Escobar JM, Montero G, Rodríguez E (2004) Local refinement of 3-D triangulations using object-oriented methods. Adv Eng Soft 35:693–702
15. Knupp PM (2000) Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. Part II-A frame work for volume mesh optimization and the condition number of the jacobian matrix. Int J Num Meth Eng 48:1165–1185
16. Knupp PM (2001) Algebraic mesh quality metrics. SIAM J Sci Comput 23:193–218
17. Kossaczky I (1994) A recursive approach to local mesh refinement in two and three dimensions. J Comput Appl Math 55:275–288
18. Löhner R, Baum JD (1992) Adaptive h-refinement on 3-D unstructured grids for transient problems. Int J Num Meth Fluids 14:1407–1419
19. Maubach J (1995) Local bisection refinement for n-simplicial grids generated by reflection. SIAM J Sci Comput 16:210–227
20. Mitchell WF (1989) A comparison of adaptive refinement techniques for elliptic problems. ACM Trans Math Soft 15:326–347
21. Montenegro R, Montero G, Escobar JM, Rodríguez E (2002) Efficient strategies for adaptive 3-D mesh generation over complex orography. Neural, Parallel & Scientific Computation 10:57–76
22. Montenegro R, Montero G, Escobar JM, Rodríguez E, González-Yuste JM (2002) Tetrahedral mesh generation for environmental problems over complex terrains. Lecture Notes in Computer Science 2329:335-344
23. Montenegro R, Escobar JM, Montero G, Rodríguez E (2005) Quality improvement of surface triangulations. Proc 14th Int Meshing Roundtable 469–484, Springer, Berlin
24. Montenegro R, Cascón JM, Escobar JM, Rodríguez E, Montero G (2006) Implementation in ALBERTA of an automatic tetrahedral mesh generator. Proc 15th Int Meshing Roundtable 325–338, Springer, Berlin
25. Montero G, Montenegro R, Escobar JM, Rodríguez E, González-Yuste JM (2004) Velocity field modelling for pollutant plume using 3-D adaptive finite element method. Lecture Notes in Computer Science 3037:642–645
26. Montero G, Rodríguez E, Montenegro R, Escobar JM, González-Yuste JM (2005) Genetic algorithms for an improved parameter estimation with local refinement of tetrahedral meshes in a wind model. Adv Eng Soft 36:3–10
27. Rivara MC (1987) A grid generator based on 4-triangles conforming. Mesh-refinement algorithms. Int J Num Meth Eng 24:1343–1354
28. Rivara MC, Levin C (1992) A 3-D refinement algorithm suitable for adaptive multigrid techniques. J Comm Appl Numer Meth 8:281–290
29. Schmidt A, Siebert KG (2005) Design of adaptive finite element software: the finite element toolbox ALBERTA. Lecture Notes in Computer Science and Engineering 42, Springer, Berlin
30. Thompson JF, Soni B, Weatherill N (1999) Handbook of grid generation, CRC Press, London
31. Traxler CT (1997) An algorithm for adaptive mesh refinement in n dimensions. Computing 59:115–137