Implementation in ALBERTA of an Automatic Tetrahedral Mesh Generator

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

[1] 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

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

This paper introduces a new automatic tetrahedral mesh generator on the adaptive finite element ALBERTA code. The procedure can be applied to 3-D domains with boundary surfaces which are projectable on faces of a cube. The generalization of the mesh generator for complex domains which can be split into cubes or hexahedra is straightforward. The domain surfaces must be given as analytical or discrete functions. Although we have worked with orthogonal and radial projections, 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 cube into six tetrahedra. The main idea is to construct a sequence of nested meshes by refining only the tetrahedra which have a face on the cube projection faces. The virtual projection of external faces defines a triangulation on the domain boundary. The 3-D local refinement is carried out such that the approximation of domain boundary surfaces verifies a given precision. Once this objective is achieved reached, those nodes placed on the cube faces are projected on their corresponding true boundary surfaces, and inner nodes are relocated using a linear 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, a mesh optimization procedure must be applied. Finally, the efficiency of the proposed technique is shown with several applications.

1 Introduction

In finite element simulation in engineering, it is crucial to adapt automatically the three-dimensional discretization to the geometry and to the solution. Many authors have made great efforts in the past to solve this problem in different ways. A perspective on adaptive modeling and meshing can be studied in [1]. The main objective is to achieve a good approximation of the *real* solution with a minimal user intervention and a low computational cost. It is clear that as the complexity of the problem increases (domain geometry and model), the methods for approximating the solution are more complicated. ALBERTA [2, 3] is a software which 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 [4]. 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. These elements are subdivided into tetrahedra by constructing a main diagonal and its projections on its faces for each hexahedral element. The local bisection of the resulting elements is recursively carried out by using ideas of the longest edge and the newest vertex bisection methods. Details about the local refinement technique implemented in ALBERTA for two and three dimensions can be analyzed in [5]. 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 meshes 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 algorithms for tetrahedral meshes [6, 7].

2 Automatic Mesh Generator

In this section, we present the main ideas which have been introduced in the mesh generation procedure. In section 2.1, 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 2.2, we continue with the presentation of different strategies to obtain an adapted mesh which can approximate the surface boundaries of the domain within a given precision. We construct the mesh of the domain by projecting the boundary nodes from a plane face to the true boundary surface and by relocating the inner nodes. These two steps are summarized in sections 2.3 and 2.4, respectively. Finally, in section 2.5 we present a procedure to optimize the resulting mesh.

2.1 Initial Coarse Mesh

In order to understand the idea of the proposed mesh generator, it is convenient to first consider a domain of which the boundary can be projected on the faces of a cube. A second case is to consider a parallelepiped instead of a cube. In this last case, an automatic decomposition of the parallelepiped into cubes can be carried out. At present, we have implemented in ALBERTA these two cases. Nevertheless, in the input data we could define an object outline with connected cubes and/or parallelepiped, such that the boundary of the domain is obtained by a one-to-one projection from the boundary faces of the object outline to the true boundary surface. Once the decomposition in cubes is done, we build an initial coarse tetrahedral mesh by splitting all cubes into six tetrahedra [5]. For this purpose, it is necessary to define a main diagonal on each cube and the projections on its faces, see Figure 4(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 number of recursive global bisections [5] which is necessary to fix a uniform element size in the whole initial mesh.

The same technique can be applied by considering a decomposition of the object outline into hexahedra instead of cubes. In this case, the recursive local refinement technique [5] introduced in ALBERTA may produce poor quality elements and, consequently, degenerate meshes. In this paper, as a first approach, we have used a decomposition of the object outline into cubes.

2.2 Local Refined Mesh

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 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 boundary faces of cubes, such that the resulting triangulation after node projection on the true boundary surface is a good approximation of this boundary surface of the domain. 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 [8]. We remark that the true surface may be given by an analytical or a discrete function, such that each point of a cube face 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 a boundary face of cubes. 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 cube faces; see for example Figure 4(b). Once all these nodes are *virtually* projected on the true surface, the application of the derefinement criterion developed in [8], with a given derefinement parameter ε, defines different adaptive triangulations for each face of the cube. We remark that the derefinement criterion fixes which nodes, placed on cube faces, can not be eliminated in the derefinement process in order to obtain a good approach of the true surface. Specifically, a node can not be eliminated if the distance between its virtual position on the true surface and the middle point of its *surrounding edge* is greater than ε. Then, for conformity reasons other nodes of the 3-D triangulation can not be removed. Besides, all node belonging to the coarse initial mesh continues in all the levels of the derefined sequence of nested meshes. As the derefinement criterion in ALBERTA is associated to elements, we mark for derefinement all tetrahedra containing a node which can be eliminated. 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 analyze elements with two sons and if the node, that was introduced by their father's bisection, verifies the derefinement condition, then we mark its two sons for derefinement.

The second strategy only works with a local refinement algorithm. In this case, the idea is to apply a recursive refinement on all tetrahedra containing a face placed on a boundary cube face and, at the same time, verifying that the distance between the points of the virtual triangle defined by the projection of its nodes on the true surface and the corresponding points on the true surface is greater than ε.

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. For example, this situation can occur when we have surfaces defined by a discrete function with a very high resolution. Nevertheless, the user can control the number of recursive bisections.

On the other hand, the problem of the second strategy is to determine for each tetrahedron face, placed on a boundary face of cubes, if it 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 son faces. Suppose, for example that true surface is given by a discrete function. Then, the subdivision criterion stops for a particular face when all the surface discretization points on this face have been analyzed and all of them verify the approximation criterion.

2.3 Projection on Boundary Surfaces

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 the projection of new nodes on complex surfaces.

In this last case, the code does not work properly, as it is only prepared to manage *valid* meshes.

For this reason, a new strategy must be developed in the mesh generator. The projection is really done only when we have defined the local refined mesh by using one of the methods proposed in the previous section. Then, the nodes placed on the cube faces are projected on their corresponding boundary surfaces, maintaining the position of the inner nodes of the domain. We have remarked that any one-to-one projection can be defined: orthogonal, spherical, cylindrical, etc.

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

2.4 Relocation of Inner Nodes

There would be several strategies for defining a new 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. 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.

2.5 Mesh Optimization: Untangling and Smoothing

An efficient procedure is necessary to optimize the pre-existing 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 [10]; 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 $N(v)$ [9]. 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 [9], 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 [11], 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 [12].

In this work we have applied, for simultaneous smoothing and untangling of the mesh by moving their inner nodes, the proposed modification [11] to one objective function derived from an *algebraic mesh quality metric* studied in [13], but it would also be possible to apply it to other objective functions which have barriers like those presented in [14].

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 [15] and [16]. 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 Applications

The performance of our new mesh generator is shown in the following three applications. The first corresponds to a domain defined over a complex terrain, the second to a sphere and the third to a cube with all faces deformed by Gaussian functions.

3.1 Domain Over Complex Terrain

In the last few years, we have developed a tetrahedral mesh generator that approximates the orography of complex terrains with a given precision [17, 18]. To do so, we only have digital terrain information. Our domain is limited on its lower part by the terrain and on its upper part by a horizontal plane placed at a height at which the magnitudes under study may be considered steady. The lateral walls are formed by four vertical planes. The generated mesh could be used for numerical simulation of environmental phenomena, such as wind field adjustment [19], fire propagation or atmospheric pollution [20]. The following procedures are mainly involved in this automatic mesh generation: a Delaunay triangulation method [21, 22], a 2-D refinement/derefinement algorithm [8] and a simultaneous untangling and smoothing algorithm [11]. Besides, we have recently developed a new method for quality improvement of surface triangulations, by using optimal local projections [15, 16], which can be introduced in the mesh generator.

Alternatively to this strategy, the automatic mesh generator proposed in this paper can be used for the same purpose. As a practical application we have considered a rectangular area in *Isla de La Palma* (Canary Islands) of 22×16 km. The upper boundary of the domain has been placed at $h = 6$ km. To define the topography we use a digitalization of the area where heights are defined over a uniform grid with a spacing step of 200 m in directions x and y. We start from a parallelepiped of $22 \times 16 \times 6$ km initially subdivided into $11 \times 8 \times 3$ cubes with edge sizes of 2 km. Each cube is subdivided into six tetrahedra by using the subdivision proposed in [5], see Figure 4(a). This discretization is used to define the uniform initial triangulation τ_1 of the parallelepiped. We refine it 18 times by constructing a recursive bisection of all tetrahedra which contain a face placed on the lower face of the parallelepiped. If we applied 6 global refinements by using the 4-T Rivara's algorithm [23] instead of previous recursive bisections, the resultant 2-D triangulation on the lower face of the parallelepiped would be the same.

Once the orography is virtually interpolated on this local refined mesh, the derefinement condition, introduced in [8], is applied with a derefinement parameter of $\varepsilon = 25$ m. Then, we make an orthogonal projection on the terrain of the adaptive triangulation obtained on the lower face of the parallelepiped. Besides, we relocate the other nodes vertically by using a proportional criterion. The adapted mesh has 65370 tetrahedra and 15263 nodes, see Figure 1(a), and it nears the terrain surface with an error less than $\varepsilon = 25$ m.

This mesh has 115 inverted tetrahedra, its average quality measure is $\overline{q}_\kappa = 0.68$ and its minimum quality is 0.091, see reference [11] and Figure 2. The node distribution is hardly modified after five steps of the optimization process by using our modified objective function. We remark that we have not relocated those nodes placed on the terrain during this optimization process.

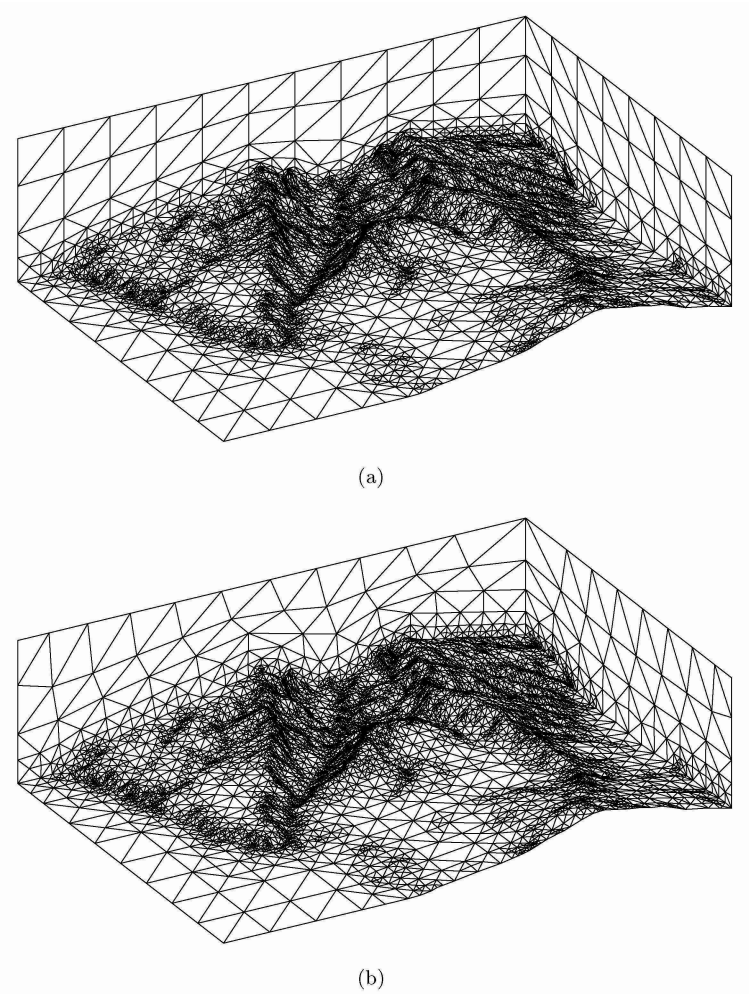


Fig. 1. Detail of *Isla de La Palma* (Canary Island): (a) initial mesh and (b) resulting mesh after five steps of the optimization process

The evolution of the mesh quality during the optimization process is represented in Figure 2. This measure tends to stagnate quickly. The quality curves corresponding to the second and fifth optimization steps are close. The average quality measure increases to $\overline{q}_\kappa = 0.75$. After this optimization process, the worst quality measure of the optimized mesh tetrahedra is 0.34. Finally, we remark that the number of parameters necessary to define the resulting mesh is quite low, as well as the computational cost. The total CPU time for the initial mesh and its optimization is less than 1 minute on an Intel Pentium M processor, 2.26 GHz and 2 Gb RAM memory. In particular, the computational cost of five iterations of the simultaneous untangling and smoothing

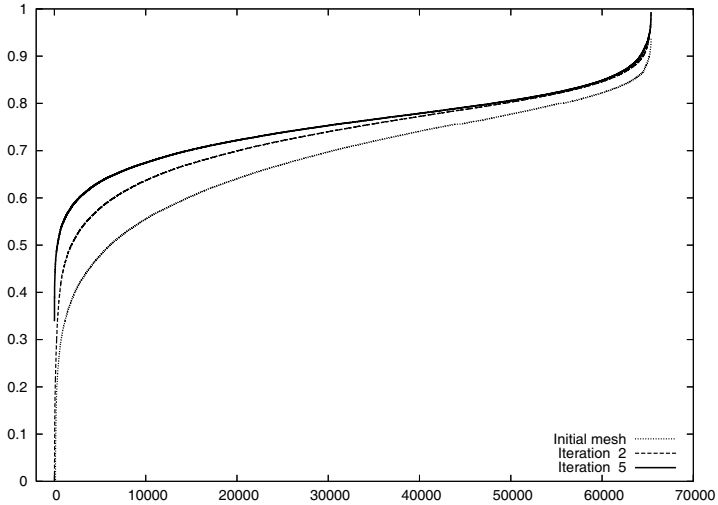


Fig. 2. Quality curves for the initial and optimized meshes after two and five iterations for the domain defined in *Isla de La Palma* (Canary Island)

procedure is about half a minute. At the first iteration of this optimization process the mesh is untangled.

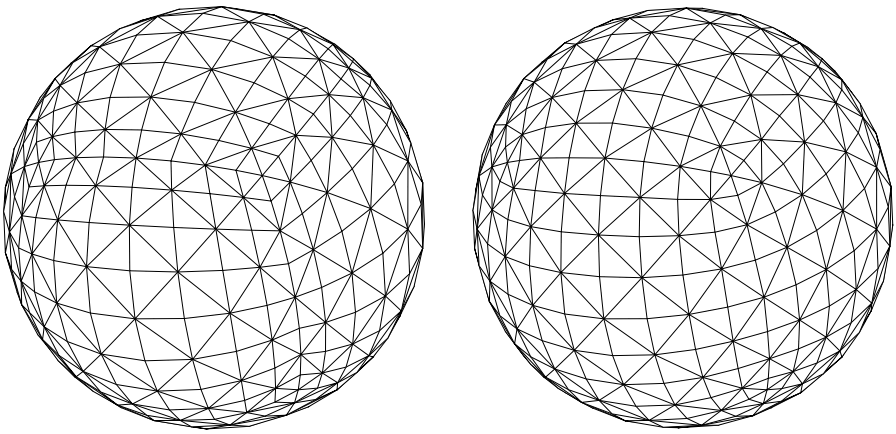
3.2 Sphere

We consider now a 3-D spherical domain as a first application where all cube faces are radial projected on a curve surface. We start from a cube divided into six tetrahedra, see Figure 4(a), and we refine 9 times constructing a recursive bisection of all tetrahedra which contain a face placed on a face of the cube. The resulting mesh contains 577 nodes and 2016 tetrahedra. Then, we make a radial projection on the spherical surface of triangulations defined on the cube faces. Besides, we relocate the inner nodes radially by using a proportional criterion. A view of the resulting mesh can be seen in Figure 3(a).

No inverted elements appear in this process and high quality elements are produced. Its average quality measure is $\overline{q}_\kappa = 0.71$ and its minimum quality is 0.48. If we use the tetrahedral mesh optimization presented in [11] by only relocating inner nodes of the domain, the mesh quality is improved with a minimum value of 0.55 and an average $\overline{q}_\kappa = 0.73$. We remark that the improvement is not so significant after ten iterations, since the initial mesh has good quality. The CPU time for constructing the initial mesh is approximately 0.2 seconds and for its smoothing process is 0.5 seconds on a Intel Pentium M processor, 2.26 GHz and 2 Gb RAM memory.

In this application we also show the smoothing possibility of the surface triangulation. So, we use the technique proposed in [15, 16] to improve the quality of the spherical surface triangulation presented in Figure 3(a). This

surface mesh contains 386 nodes and 640 triangles with an average quality of 0.85 and a minimum one of 0.65. After five iterations of surface smoothing process we obtain the mesh shown in Figure 3(b) which has an average quality of surface triangles of 0.86 and a minimum one of 0.78. If we keep the boundary nodes in their new positions and we optimize again the mesh by only moving inner nodes, then the tetrahedral mesh results with an average quality measure $\overline{q}_\kappa = 0.72$ and its minimum quality is 0.52. It can be observed that these values are between those obtained for the initial mesh and its optimization. Nevertheless, the difference is not significant due to the regularity of the initial mesh. The procedure proposed in this paragraph could be interesting when we start from meshes with poor quality surface triangulations.



(a) (b)

Fig. 3. Surface triangulation of the sphere: (a) initial mesh and (b) resulting mesh after five steps of the surface optimization process

3.3 Domain with Gaussian Surfaces

As a last application we present a unit cube domain of which faces are orthogonal projected on different surfaces defined by Gaussian functions. We start from an initial coarse triangulation composed by 8 nodes and 6 tetrahedra. This subdivision is proposed in [5], see Figure 4(a). We refine this triangulation τ_1 by 18 recursive bisections of all the tetrahedra which contain a face placed on a cube face, resulting a mesh with 48703 nodes and 198672 tetrahedra, see Figure 4(b). Once the boundary surface information was virtually interpolated on this local refined mesh, the derefinement condition [8] was applied with a derefinement parameter $\varepsilon = 0.00001$.

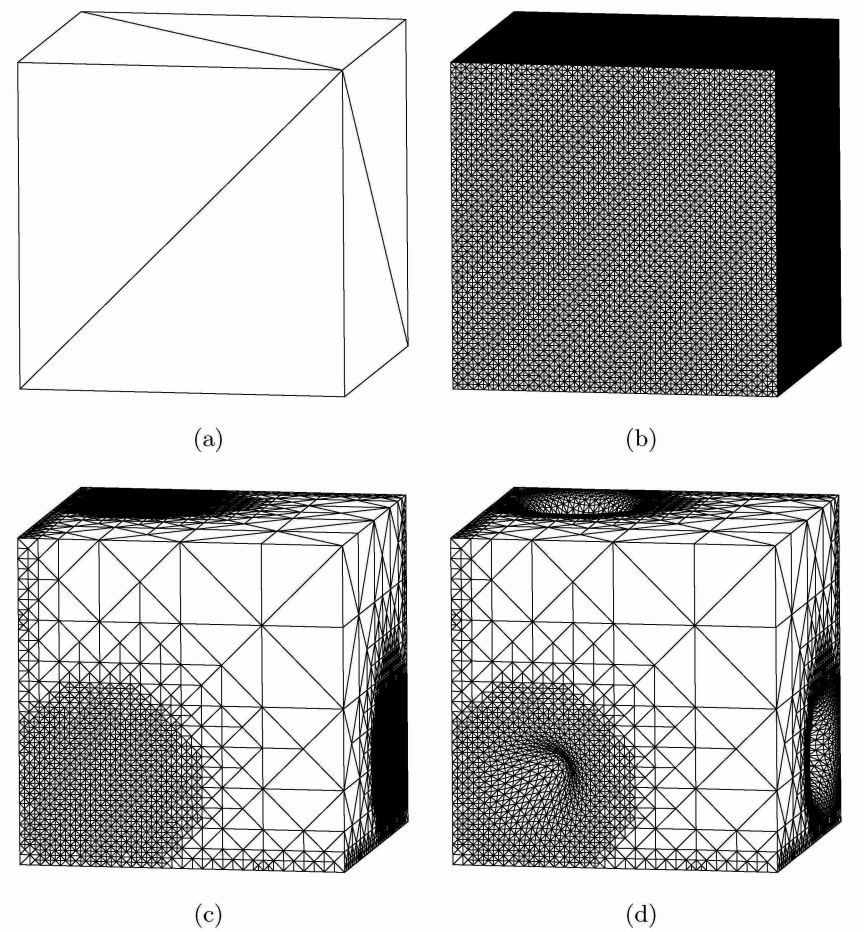


Fig. 4. Main stages of the mesh generator for the domain with Gaussian surfaces on all its faces: (a) initial mesh, (b) global face refinement, (c) local face refinement after applying the derefinement procedure and (d) projection on boundary surfaces

In Figure 4(c) the corresponding mesh is presented and it contains 23520 nodes and 60672 tetrahedra. Then, we make an orthogonal projection on the domain boundary surface of the adaptive triangulation obtained previously over the cube faces. Besides, we relocate the inner nodes by using a proportional criterion along the Cartesian directions. The resulting mesh is shown in Figure 4(d) and it initially has 240 inverted tetrahedra with average quality measure $\overline{q}_\kappa = 0.64$, see Figure 6. An inner view of the surface triangulation may be observed in Figure 5. The evolution of the mesh quality during the optimization process, by applying the simultaneous untangling and smoothing procedure [11] to the inner nodes of the domain, is represented in Figure 6. The quality curves corresponding to the second and tenth optimization steps

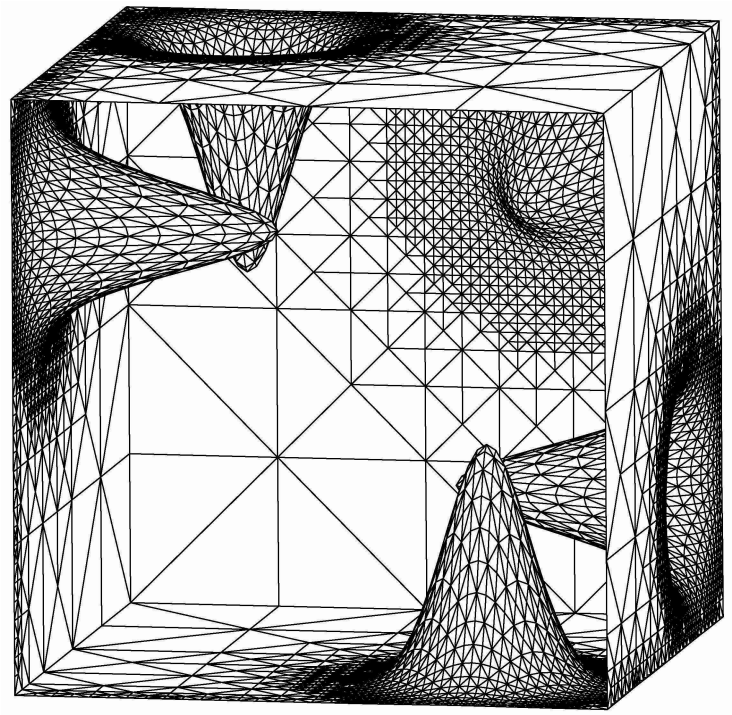


Fig. 5. Inner view of the surface triangulation for the domain with gaussian surfaces

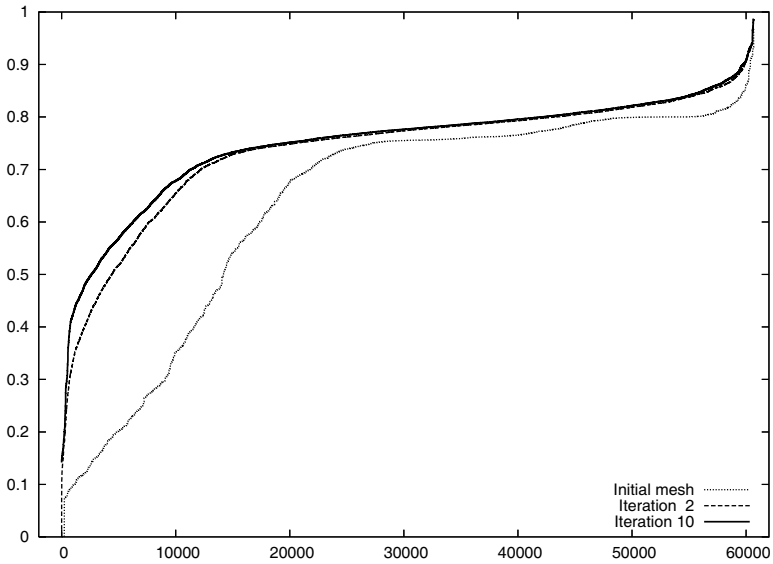


Fig. 6. Quality curves for the initial and optimized meshes after two and ten iterations for the domain with gaussian surfaces

are very close. The average quality measure increases to $\overline{q}_\kappa = 0.75$ and the minimum value improves to 0.14. Finally, we remark that the final mesh is generated in less than one minute on a Intel Pentium M processor, 2.26 GHz and 2 Gb RAM memory. In particular, the computational cost of five iterations of the simultaneous untangling and smoothing procedure is about half a minute. We observe that at the second iteration of this optimization process the mesh is untangled.

4 Conclusions and Future Research

The proposed mesh generator is an efficient method for creating tetrahedral meshes on domains with boundary faces projectable on faces of cubes and it is used as pre-processor for ALBERTA. We remark that it requires a minimum user intervention and has a low computational cost. The main ideas presented in this paper for automatic mesh generation could be used for different codes which work with other tetrahedral or hexahedral local refinement/derefinement algorithms. With these ideas, more complex domains could be meshed by decomposing its *outline* into a set of connected cubes or hexahedra. 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.

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 (see http://www.dca.iusiani.ulpgc.es/proyecto0507). Besides, we thanks to the authors of ALBERTA [3] for the code availability in internet [2] and for their suggestions.

References

1. Carey GF (2006) Comp Meth Appl Mech Eng 195:214–235
2. ALBERTA – An Adaptive Hierarchical Finite Element Toolbox, http://www.alberta-fem.de/
3. Schmidt A, Siebert KG (2005) Design of Adaptive Finite Element Software: The Finite Element Toolbox ALBERTA. Lect N Comp Sci Eng 42. Springer, Berlin Heidelberg New York

4. Mitchell WF (1989) ACM Trans Math Soft 15:326–347
5. Kossaczky I (1994) J Comp Appl Math 55:275–288
6. Löhner R, Baum JD (1992) Int J Num Meth Fluids 14:1407–1419
7. González-Yuste JM, Montenegro R, Escobar JM, Montero G, Rodríguez E (2004) Adv Eng Soft 35:693–702
8. Ferragut L, Montenegro R, Plaza A (1994) Comm Num Meth Eng 10:403–412
9. Freitag LA, Knupp PM (2002) Int J Num Meth Eng 53:1377–1391
10. Freitag LA, Plassmann P (2000) Int J Num Meth Eng 49:109–125
11. Escobar JM, Rodríguez E, Montenegro R, Montero G, González-Yuste JM (2003) Comp Meth Appl Mech Eng 192:2775–2787
12. Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear Programing: Theory and Algorithms. John Wiley and Sons, Inc. New York.
13. Knupp PM (2001) SIAM J Sci Comp 23:193–218
14. Knupp PM (2000) Int J Num Meth Eng 48:1165–1185
15. Escobar JM, Montero G, Montenegro R, Rodríguez E (2006) Int J Num Meth Eng 66:740–760
16. Montenegro R, Escobar JM, Montero G, Rodríguez E (2005) Quality improvement of surface triangulations. In: Hanks BW (ed) Proceedings of the 14th International Meshing Roundtable, 469–484. Springer, Berlin Heidelberg New York
17. Montenegro R, Montero G, Escobar JM, Rodríguez E, González-Yuste JM (2002) Lect N Comp Sci 2329:335–344
18. Montenegro R, Montero G, Escobar JM, Rodríguez E (2002) Neural, Parallel & Scientific Computation 10:57–76
19. Montero G, Rodríguez E, Montenegro R, Escobar JM, González-Yuste JM (2005) Adv Eng Soft 36:3–10
20. Montero G, Montenegro R, Escobar JM, Rodríguez E, González-Yuste JM (2004) Lect N Comp Sci 3037:642–645
21. George PL, Hecht F, Saltel E (1991) Comp Meth Appl Mech Eng 92:269–288
22. Escobar JM, Montenegro R (1996) Adv Eng Soft 27:27–39
23. Rivara MC (1987) Int J Num Meth Eng 24:1343–1354