Journal of Computational Methods in Sciences and Engineering 5 (2005) 1–10
IOS Press

Computational aspects of the refinement of 3D tetrahedral meshes

J.P. Suárez[a,*], P. Abad[a], A. Plaza[b] and M.A. Padrón[c]
[a]*Department of Cartography and Graphic Engineering, University of Las Palmas de Gran Canaria. 35017-Las Palmas de Gran Canaria, Spain*
[b]*Department of Mathematics, University of Las Palmas de Gran Canaria, Spain*
[c]*Department of Civil Engineering, University of Las Palmas de Gran Canaria, Spain*

Abstract. The refinement of tetrahedral meshes is a significant task in many numerical and discretizations methods. The computational aspects for implementing refinement of meshes with complex geometry need to be carefully considered in order to have real-time and optimal results. In this paper we study some computational aspects of a class of tetrahedral refinement algorithms. For local adaptive refinement we give numerical results of the computational propagation cost of a general class of longest edge based refinement and show the implications of the geometry in the global process. Moreover we study the conformity process based on the longest edge bisection and give an algorithm and data structure to efficiently handle tetrahedral refinement.

Keywords: Tetrahedral meshes, refinement, finite element, 65M50, 65N50, 65Y20

1. Introduction and motivation

Unstructured mesh adaptation is clearly recognized as an efficient and powerful method for improving the accuracy of the solution as well as for capturing the behavior of physical phenomena, even if it is not known a priori [1–4]. Hence, three-dimensional complex simulations are now commonly used in many engineering fields. Moreover, reducing the number of nodes (also known as derefinement technique) allows to substantially reduce the CPU time. The refinement problem (considered as the derefinement inverse problem) can be described as any technique involving the insertion of additional vertices in order to produce meshes with higher precision.

Figure 1 shows an example of mechanical component meshed using a Delaunay 3D algorithm. It can be noted the presence of refined areas around the holes of the piece which allow a high precision of such cavities structures.

In general, there are two possible strategies for refining (or derefining) a mesh: (1) local refinement (derefinement), when the process modifies a group of triangles and (2) global (also known as uniform) refinement, when all triangles in the mesh are chosen for refining (derefining). The manipulation of meshes in real-time applications requires particular strategies to adaptively refine or simplify the meshes. For example, the run-time of local refinement algorithms is considerably influenced by the refinement propagation: the extent of secondary refinements induced in neighboring elements by the initiating

*Corresponding author. Tel.: +1 34 928 457268; Fax: +1 34 928 451872; E-mail: jsuarez@dcegi.ulpgc.es.

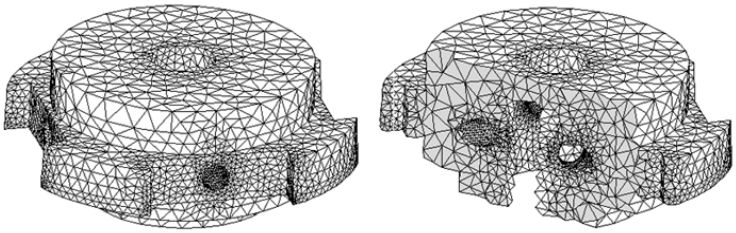

Fig. 1. An example of a mechanical part meshed using a Delaunay algorithm.

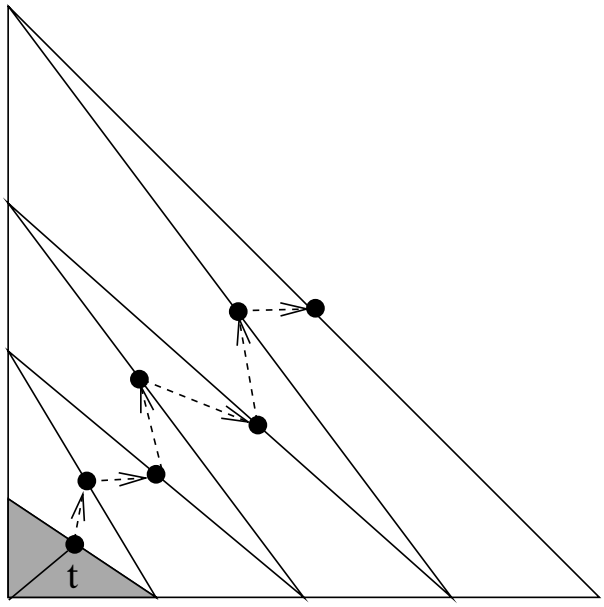

Fig. 2. Longest edge refinement propagation. The dependencies in the propagation when refining t are indicated by arrows.

element. Particularly, in [2] the authors referred as an ominous problem the fact that the propagation could get the worst case, that is: refining a single tetrahedron also implies the refinement of all the tetrahedral in the mesh. Figure 1 shows the worst case of a refinement propagation in 2D. In this sense, it seems necessary to look at the propagation issues in the local refinement of meshes. Other related problem that may affect efficiency in local refinement is the adjacency relationship between elements used in the algorithms. We propose an algorithm and a data structure to handle the tetrahedral adjacencies respecting an edge. An analogous algorithm and data structure but for the bi-dimensional case have been presented in [9]. Tetrahedral adjacencies are a common class of adjacencies structures appearing in local refinement so the proposed algorithm is of value to other similar applications.

2. Two-steps refinement problem

The local refinement of triangular meshes involves two main tasks. The first is the partition of the target triangles or tetrahedra (tets throughput this work) and the second is the propagation or extension to

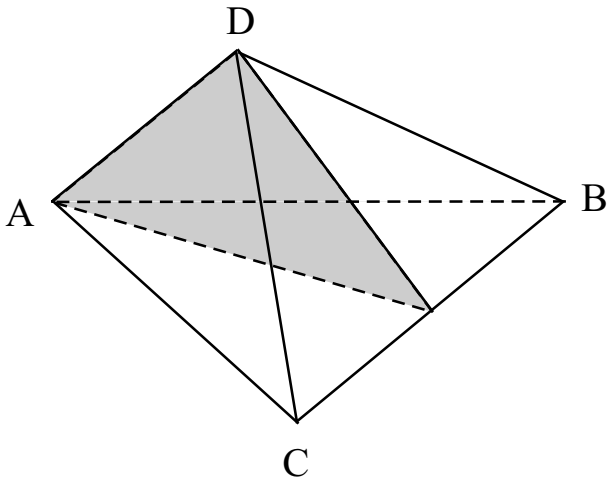

Fig. 3. Longest Edge bisection of a tetrahedron.

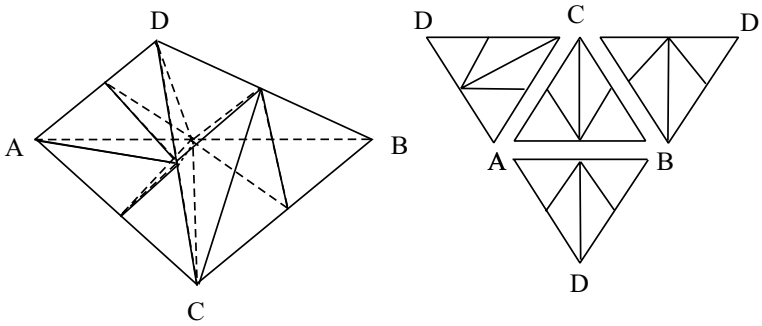

Fig. 4. 8T-LE partition of tet ABCD on the left and the faces unfolded in plane.

preserve the 'cracks' in the resulted mesh. It should be noted that uniform refinement does not consider the propagation as the refinement is uniform in the mesh.

Several approaches for partitioning tets have been studied, see [1] for a short survey. The simplest is Bisection into two sub-tets by the midpoint of one of the edges. If the Longest Edge (LE) is chosen for the bisection, then this is called Longest Edge Bisection, see Fig. 3.

A more sophisticated partition of a tet based on the longest edge is the 8-Tetrahedra Longest Edge partition (8T-LE partition [4]). The partition is defined as follows: (i) LE-bisection of t producing tets t_1,t_2 (ii) Bisection of t_i by the midpoint of the unique edge of t_i which is also the longest edge of a common face of t_i with the original tet t, producing tets t_{ij}, for $i, j = 1, 2$. (iii) Bisection of each t_{ij} by the midpoint of the unique edge corresponding to an edge of the original tetrahedron. The 8T-LE partition subdivides a tet into fifty one different possible configurations, if rotations are excluded [4]. In Fig. 4 we show one of the possible patterns that arise from the 8T-LE partition.

A different tet partition widely used in numerical computations is the Standard partition, also called 3D-Freudenthal-Bey partition [1,5]. The original tet is divided into eight subt-tets by cutting off the four corners by planes trough the midpoints of the edges and subdividing the remaining octahedron by selecting one of three possible interior diagonals, Fig. 5. This interior diagonal has to be chosen carefully

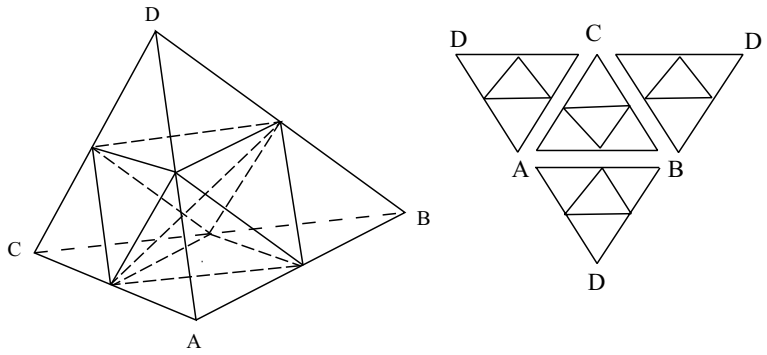

Fig. 5. Standard Partition of a tetrahedron and the faces unfolded in plane.

in order to satisfy the non-degeneracy of the meshes obtained when the partition is recursively applied.

The second task in the local refinement problem is to ensure the conformity of the mesh, also called consistency. This condition means that the intersection of each pair of adjacent tets is either a common edge or an entire face. It is necessary to determine additional irregular patterns, which makes it possible to extend the refinement to neighbor triangles and so, to assure the conformity of the mesh.

The combined use of bisection methods and 8-son (so named as eight sub-tets are generated) based methods have many advantages and they are usually implemented in many finite element codes. We briefly summarize some features:

Remark 1:

1. Bisection methods main advantage is that they result in less local refinements compared to 8-son subdivision because elements are divided into two instead of eight.
2. Bisection methods generate up to $n!2^{n-2}$ (n=3) similarity classes. Standard partition produces at least $\frac{n!}{2}$ similarity classes [1]. For the 8T-LE this feature is still an open problem, however experience indicates that it behaves well in generating new similarity classes.
3. Bisection methods are preferable for constructing smooth meshes (transition between small and large elements should be gradual.

3. The conformity process in Longest-Edge based refinement

First, we provide the definition of the 3D-Longest-Edge Propagation Path [7].

Definition 1. 3D-Longest-Edge Propagation Path, 3D-LEPP The 3D-Longest-Edge Propagation Path of any tetrahedron t is the set of all neighbor tetrahedra (by the longest edge) having respective longest edge greater than or equal to the longest edge of the preceding tet in the path.

If the longest edge bisection is used to refine a given tetrahedron t, then the 3D-Longest-Edge Propagation Path provides the list of tets to be refined. As a consequence, the 3D-Longest-Edge Propagation Path provides the main adjacency list used by the longest edge refinement algorithms. Computing the LEPP in 3D may be a very costly task. Similar definition for the bi-dimensional case is possible, the 2D-LEPP of triangles [9].

Next we introduce the shell of a tet edge that is of help to calculate the LEPP.

Definition 2. Given an edge, a shell is the polyhedron made up of the tets sharing this edge.

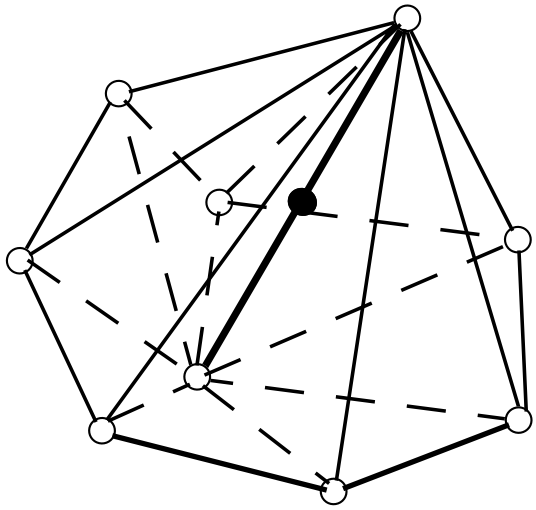

Fig. 6. Shell of tetrahedra sharing an internal edge.

The shell is of relevant interest in Longest-Edge based refinement since it is the geometric place in the mesh that LEPP traverses in the refinement process. It also has been used in combination with Delaunay algorithms, as it is a geometric place where Delaunay point may be inserted to get improved internal dihedral angles [6].

Next algorithms briefly present our proposal for 3D local refinement. For the purpose of this paper we will concentrate on a combined Bisection-Standard based refinement algorithm. The Standard partition is used to refine the target tets and the Longest-Edge Bisection is employed to assure the conformity in the mesh.

Algorithm 3D-Refinement(τ,τ_0)
/* Input: τ tet mesh, τ_0 set of tets to refine
/* Output: τ new tet mesh
Standard-Refine(τ,τ_0)
Assure-Conformity(τ,τ_0)
End

Algorithm Assure-Conformity(τ)
/* Input: τ tet mesh
/* Output: τ new tet mesh
Let L be the set of non-conforming points in τ
While $L \neq \emptyset$ **do**
    Let e be the surrounding edge of p $\in$ L
    S=shell(e)
    For each tet t $\in$ S **do**
        Perform Longest-Edge Bisection of t
        Let p be the new non-conforming point in t
        $L = L \cup p$
    End
End

Table 1
Statistic of refinement propagation in a tube section mesh (Fig. 7)

# Tetrahedra	2016	16128	129024
Propagation mean	21.5154	15.1231	13.8262
Refinement Level	0	1	2

Table 2
Statistic of refinement propagation in a sphere mesh (Fig. 8)

# Tetrahedra	1927	15416	123328
Propagation mean	97.7000	24.1951	13.2069
Refinement Level	0	1	2

The first step in 3D-Refinement algorithm can be usually performed in constant time as long as the algorithm implement pre-computed Standard type partitions. So, the main computational effort of the local algorithms is the propagation of the refinement (Assure-Conformity algorithm). The propagation is a necessary task for avoiding non conforming nodes, that is, the requirement that the intersection of non-disjoint tetrahedra is either a common vertex or a common side, and also for producing gradual and smooth meshes. The main strategy to implement propagation is based on the longest-edge propagation, which consists to repeat the refinement in neighboring elements following the LEPP as long as non-conforming nodes are present.

The algorithm Assure-Conformity performs the refinement propagation to obtain a conforming mesh. The conformity is checked for any non-conforming point in the mesh, and then, tets are properly subdivided. The procedure for assuring the conformity is analogous to that presented in [7], however our version here is not recursive. This makes it possible the translation to a parallel framework. Moreover, it can be seen as other version of that presented in [4]. The improvement respecting that other version is that we use an explicit data structure to handle the shells, as we will see next.

It should be pointed out that Assure-Conformity does not explicitly make use of a LEPP data structure. However, the main while loop in the algorithm traverses tets that continuing belong to the LEPP.

Let us now consider how to store the data of each tet. The pre-processor has to create and describe the mesh with the data structures that are the input for the solver. Firstly, it is necessary to trace the vertices that forms the tets, for example A,B,C and D. The best choice for implementing the bisection is to store the four vertices related to each tet in such that the vertices related to the longest edge are stored in the first two positions.

As shells are basic data structures used in the 3D-Refinement algorithm it is convenient to be stored together with the mesh. A simple and efficient way to store a shell provided that every tet in a mesh is numbered, is as follows:

$$\begin{array}{cccc} e_1 & e_2 & \cdots & e_m \\ t_1 & r_1 & \cdots & s_1 \\ t_2 & r_2 & \cdots & s_2 \\ \vdots & \vdots & \vdots & \vdots \\ t_i & r_j & \cdots & s_k \end{array}$$

where $e_1, e_2, \cdots, e_m$ are the different edges in a mesh, and for each edge, it is followed a path of tets, for example $t_1, t_2, \cdots, t_i$ that are just the tets forming the shell. Note that the order of tets in a shell is not important. The set of tets for each edge has to be closed that means that it must be possible to go trough the path and come back to the element we started. With that adjacency matrix it is easy to see which

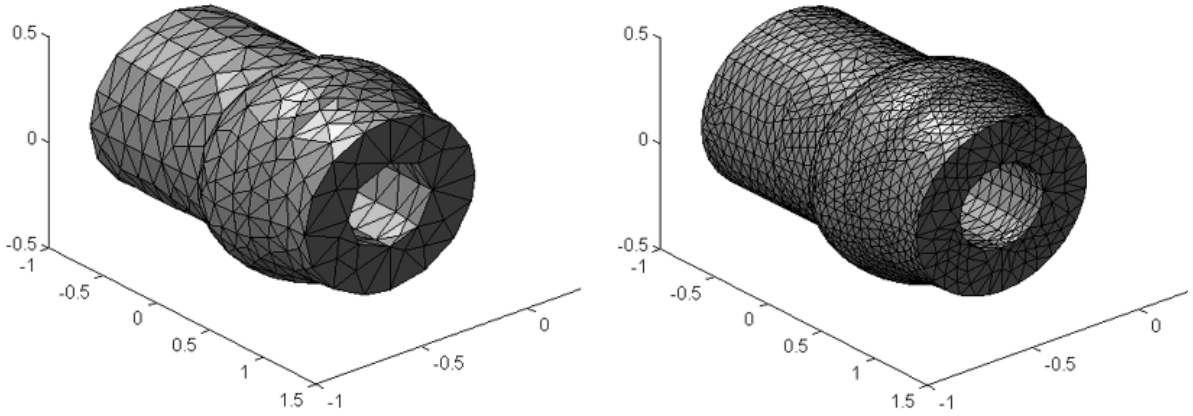

Fig. 7. Initial Delaunay mesh on the left (2016 tets) and uniform Standard refinement on the right (16128 tets.

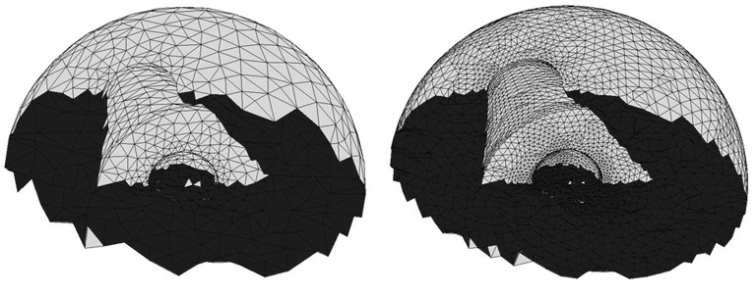

Fig. 8. Test mesh with interior cavity. Uniform Standard refinement on the right.

tets share the same edge in constant time. A possible disadvantage of previous data structure is the size when the mesh is too dense of elements. Next result gives an asymptotic result showing that the size of a shell does not indiscriminately increase when the number of refinements tends to infinity:

Proposition 1. Let τ_n be any tetrahedral mesh with N_n nodes, E_n edges, F_n triangular faces, and T_n tets. Then, the limits of the average of adjacency relations when n, the number of refinements tends to infinity verify:

$$\lim_{n\to\infty} Av\#(Tets\_per\_edge) = \lim_{n\to\infty} Av\#(faces\_per\_edge) = \frac{36}{7}$$

$$\frac{3}{2}\lim_{n\to\infty} Av\#(Tets\_per\_node) = \lim_{n\to\infty} Av\#(faces\_per\_node) = 36$$

A proof of this theorem can be found in [5].

Proposition 2. An asymptotic result for the size complexity of a shell data structure is $E \cdot \frac{36}{7}$, where E is the number of edges refined in the process.

The second part of our study reports some statistical results of the LEPP in refining meshes. We previously give a previous result proved in 2D [8], which states an asymptotic behaviour of the propagation for the Longest-Edge Bisection in 2D partition:

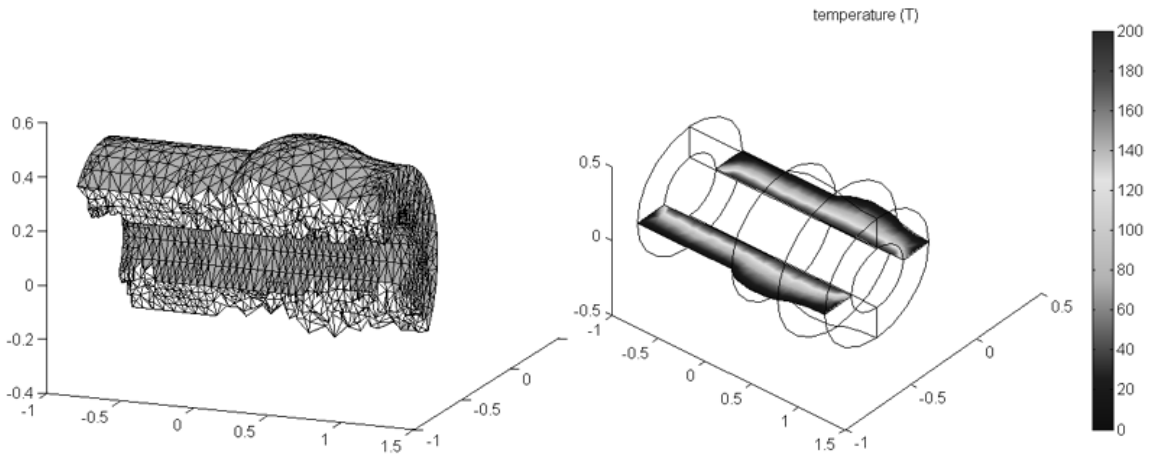

Fig. 9. Tet mesh (69838 tets) and respective solution on the right: (0.1 sg. mesher CPU time, 0.3 sg. solver time.

Proposition 3. The iterative application of the Longest-Edge Bisection to an initial triangular mesh makes the mean of the 2D-Longest-Edge Propagation Path (2D-LEPP) tend to 2 respectively, when the number of refinements applied grows to infinity.

Until now, none theoretical proof respecting the propagation problem in 3D can be found. We provide here a brief numerical study showing that the 3D-LEPP has a statistical mean approaching to a fix constant that is dependent on the partition type used in the refinement and on the initial mesh.

Our experiments numerically show that the element propagation in 3D approaches to a fix constant (between 10 and 15). It seems that the variability of the LEPP counts depends on the tets geometry of the initial mesh and on the partition type applied for refinement. This result emphasizes that Delaunay initial meshes and Standard partition constitutes a good election for the two-steps refinement problem in many of the 3D problems. A theoretical proof of that result is in progress and will be published in a future work.

We consider two initial Delaunay type meshes. First mesh in Fig. 7 is a tube section with an interior cylinder-like path. The second mesh is a sphere domain where an interior cavity is performed, see Fig. 8. Inside the cavity, a salient ellipsoid is located. The refinement around the ellipsoid provides accuracy in the model. For a clear visualization, the sphere has been cut by the middle and hence, the interior cavity and the ellipsoid can be noted (the blue area corresponds to interior tetrahedra).

For both meshes it is performed two levels of uniform refinement following the Standard partition for every tetrahedron, and a propagation following the longest edge, see 3D-Refinement algorithm, is calculated. Tables 1 and 2 summarizes the results on the 3D-LEPP propagation of the meshes. It can be seen as the mean of the propagation gets reduced in the second step of refinement to approximately thirteen, for both experiments. This support our conjecture in affirming that performing local refinement does extent to a few elements in average in the mesh.

4. Application

In this section we present an application of mesh refinement algorithm presented here to solve the 3D Heat Equation on a structural steel domain. We choose a domain like the tube section in Fig. 7. The heat

(200°C) is initially applied in the cylinder-like path inside the tube. The tube is at ambient temperature (18°C). The example illustrates the use of 3D-Refinement algorithm integrated in a simple FEM code programmed in *Matlab* to solve the heat equation in 3D:

$$\rho C \frac{\partial T}{\partial t} + \nabla \cdot (-k \nabla T) = Q \tag{1}$$

An essential step of our adaptive mesh refinement is the estimation of the local error (element-wise). After computing an approximate solution u, it is computed the residual:

$$R(u) = -\nabla \Gamma(u) + F(u) \tag{2}$$

The error is $e = u - u'$, assuming u' is the exact solution. The interpolation error estimate used is:

$$\int_{\omega} D_l h^{p_l+1} |\rho_l| dA \tag{3}$$

where h is the local mesh size, and ρ_l is the order of the finite element. D_l is a norm of the derivatives of order $p_l + 1$.

The solution is obtained for a mesh with 69838 tets after four iterations of local refinements in a time of 0.3 sg., see Fig. 9. The experiment reveals that the computational costs studied in section 3 do not yield a critical time in the overall process, and this is a common behavior for most of the problems modeled with 3D-Refinement algorithm.

5. Conclusions

We have remarked in this work the main source of computational effort by common algorithms for local refinement. These algorithms are key tools for the discretization process needed in many numerical methods as Finite Element Method, Volume Methods etc. We have numerically showed an important result regarding the element propagation and studied the behavior of a very common adjacency of tets in these algorithms. Asymptotically the LEPP does not increase when the number of refinement tends to infinity and so it is not an ominous problem in local refinement (we obtain values between 10 and 15). Furthermore, the shell adjacency to implement the refinement asymptotically tends to 36/7 when the number of refinement tends to infinity. We give an example for Delaunay meshes and using the Standard partition and a Longest-Edge based propagation. This is an useful basis for users and engineers who often uses meshing algorithms for a variety of application problems.

Acknowledgements

This research has been partially supported by Gobierno de Canarias grant PI2003/35 and ULPGC grant number UNI2003/16.

References

[1] J. Bey, Simplicial grid refinement: on Freudenthal's algorithm and the optimal number of congruence classes, *Numer. Math.* **85:1** (2000), 1–29.
[2] M. Bern and P. Plassmann, Mesh generation, in: *Handbook of Computational Geometry*, J.-R. Sack and J. Urrutia, eds, 2000, pp. 291–332.
[3] M.T. Jones and P.E. Plassmann, Parallel algorithms for adaptive mesh refinement, *SIAM J. Scie. Comp.* **18** (1997), 686–708.
[4] A. Plaza and G.F. Carey, Local refinement of simplicial grids based on the skeleton, *App. Num. Math.* **32** (2000), 195–218.
[5] A. Plaza and M.C. Rivara, Asymptotic behavior of the average adjacencies for skeleton-regular triangular and tetrahedral partitions, *J. Comp. Appl. Math.* **140** (2002), 673–693.
[6] M.C. Rivara, N. Hitschfeld and B. Simpson, Terminal-edges Delaunay (small-angle based) algorithm for the quality triangulation problem, *Computer-Aided Design* **33** (2001), 263–273.
[7] M.C. Rivara and C. Levin, A 3D refinement algorithm suitable for adaptive and multigrid techniques, *Comm. Appl. Numer. Methods Eng.* **8** (1992), 281–290.
[8] J.P. Suárez, A. Plaza and G.F. Carey, *Propagation path properties in iterative longest-edge refinement*, In Proceeding 12th International Meshing Roundtable'03. SANDIA Report SAND 2003-3030P, 2003, 79–90.
[9] J.P. Suárez, G.F. Carey and A. Plaza, Graph-based data structures for skeleton-based refinement algorithms, *Comm. Num. Meth. Eng.* **17** (2001), 903–910.