TICAM REPORT 96-54
November 1996

A New Refinement Algorithm for Tetrahedral Grids Based on Skeleton

A. Plaza and G. F. Carey

A NEW REFINEMENT ALGORITHM FOR TETRAHEDRAL GRIDS BASED ON SKELETON *

A. PLAZA † AND G.F. CAREY‡

Abstract. In this paper we present a novel approach to the development of a class of local three-dimensional simplicial refinement strategies based on ideas that have been shown to be effective for two-dimensional refinement. The algorithm presented here begins subdividing the two-dimensional triangulation composed by the faces of the tetrahedra, and then subdividing each tetrahedron in a compatible manner with the first step. The study of the complexity of the algorithm suggests another improved version with a linear complexity with respect to the worst case that is also given. The algorithm is fully atomatic and can be used to achieve local as well global refinements. Moreover, these algorithms can be applied to any initial tetrahedral mesh without any preprocessing. Although mathematical proofs are not provided here, the numerical results obtained seem to confirm that the measure of degeneracy is bounded, and converges asymptotically to a fixed value. The idea for extending a two-dimensional algorithm to a three-dimensional one may possibly be applied in a general dimension n and with other refinement algorithms.

Key words. Grid Refinement, 3D Bisection, Tetrahedra, Adaptivity

AMS subject classification. 51M20, 52B10, 65N50

1. Introduction.

Local refinement is critical to the efficient approximate solution of partial differential equations in many practical applications. Adaptivity of the mesh is particularly important in three-dimensional problems because the problem size and computational cost grow very rapidly as the mesh size is reduced. For example, in certain space-time finite element formulations and in relativity applications, 4-space discretizations may be desired. Similarly, in Boltzmann applications involving momentum space variables, discretizations in high dimensions may be needed.

As is well known, there are two main steps in local adaptive refinement [8]: the refinement of a subset of elements based on local error indicators [1, 2, 13], and the achievement of the conformity of the mesh [17, 25]. The elements that offer the simplest choice in any dimension are the simplices: triangles in two dimensions, tetrahedra in three dimensions and their analog in even higher dimensions. Many different refinements and improvement techniques for two- and three-dimensional triangulations are now available.

For example, Bank and Sherman [3] subdivide a triangle into four similar triangles by connecting the midpoints of the sides. The conformity of the mesh is made by subdivision of elements adjacent to the irregular vertices (nodes) using so-called *green division*. Bisection connects one of the vertices of the triangle to the midpoint of the opposite side. Eg. Rivara [28, 29, 31] presents two algorithms: The 2T algorithm is based on the longest edge bisection. In the 4T algorithm, the vertex opposite to the longest edge is chosen in a first step, and then the newly formed vertex is used to subdivide the initial triangle in four as shown in Figure 1. In these algorithms, the angles of the triangles in a resulting locally refined grid are uniformly bounded

*This research has been supported by DGICYESM grant number PR95-280, grant from Gobierno de Canarias, NSF grant number ECS-9412475, and by ARPA grant number DABT63-96-C-0061.

†Department of Mathematics. University of Las Palmas de Gran Canaria. Spain, and Texas Institute for Computational and Applied Mathematics (TICAM), ASE/EM Dept., University of Texas at Austin. U.S.A. aplaza@dma.ulpgc.es

‡Texas Institute for Computational and Applied Mathematics (TICAM), ASE/EM Dept., University of Texas at Austin. U.S.A. carey@cfdlab.ae.utexas.edu

away from 0 and π. Additional refinement is necessary to ensure the conformity of the mesh, but this refinement is also by the longest edge in the Rivara algorithms.

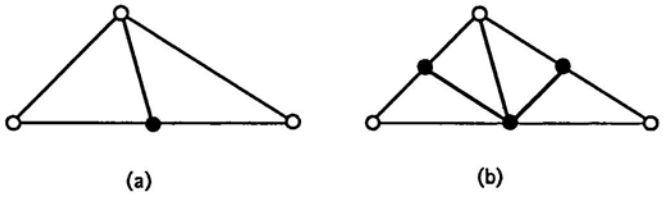


FIG. 1. *4T division of Rivara*

Another bisection method is the newest vertex bisection introduced by Mitchell in [19]. The triangle edge to be bisected is determined without any computation. Only four similarity classes of triangles and only eight distinct angles are created by this method. Hence the important condition of being bounded away from 0 and π is again satisfied. Additional refinement is also necessary in this algorithm to ensure the conformity of the mesh.

In three dimensions there are two main approaches for subdividing a single tetrahedron: octasection and bisection. Octasection methods simultaneously create eight descendants for each tetrahedron. For example Bey [7] first connects the edges of each face triangle as in the two-dimensional Bank refinement, then cuts off four sub-tetrahedra at the corners which are similar to the original one. Finally Bey's algorithm cuts the interior octahedron into four more sub-tetrahedra. Concerning irregular refinement he only considers four patterns of the 64 possibilities, and performs a global refinement in another case. This implies that *a domino effect* may occur in certain situations and excessive refinement can propagate through the mesh.

Methods based on bisection can also subdivide each tetrahedron in eight, but the elemental stage consists in bisecting the tetrahedron in two. E. Bänsch [5], presents an algorithm based on the selection of an edge as a *global refinement edge* in each tetrahedron, but imposes small perturbations of the coordinates of the nodes for avoiding incompatibilities. The algorithm presented by Rivara and Levin [32] is based on longest edge bisection. In this sense their algorithm is a generalization of the 2T algorithm of Rivara. However, it is not known into how many tetrahedra each of the original tetrahedron will be subdivided. Mathematical proofs about the non-degeneracy of the grids created are needed, although experiments suggest this holds. Also based on longest edge bisection is the algorithm by Muthukrishnan *et al.* [22], although all the meshes created are conforming. A. Liu and B. Joe [16, 17], present an algorithm (*QLRB*) similar to that of Bänsch. They classify the tetrahedra in four types and set up the types of edges depending on the type of tetrahedron. In this way they avoid evaluating the length of the edges. In addition, a shape measure is introduced. The number of similarity classes is proved to be bounded and therefore, the meshes can not degenerate.

A recursive approach is proposed by I. Kossaczký [15]. This algorithm imposes certain restrictions and preprocessing in the initial mesh. The 3D algorithm is equivalent to that given in [5]. J. M. Maubach [18] develops an algorithm for n-simplicial grids generated by reflection. Although the algorithm is valid in any dimension and the number of similarity classes is bounded, it can not be applied for a general tetrahedral grid. An additional closure refinement is needed to avoid incompatibilities. Recently, A. Mukherjee [21], has presented an algorithm equivalent to [5, 16], and

proves the equivalence with [18].

Two new algorithms based on bisection will be developed in the present work [23, 24]. Although they show similar behavior to those cited above ([5, 15, 16, 17, 18, 21]) the point of view is quite different and simpler since the three dimensional approach is based on the two dimensional one applied to the skeleton of the triangulation. The algorithm can be applied to any valid initial mesh without any restriction on the shape of the tetrahedra, since it is based on a previous classification of the edges that is local to each tetrahedron, and at the same time global for each triangular face of the mesh. Moreover, these ideas can be extended to obtain local refinement algorithms in higher dimensional spaces.

1.1. Notation and definitions. We introduce here some notations and definitions that will be used in following sections.

DEFINITION 1.1 (m-simplex). Let $V = \{X_0, X_1, \ldots, X_m\}$ be a set of $m+1$ points in R^n $(1 \leq m \leq n)$ such that $\{\overrightarrow{X_0X_i}: 1 \leq i \leq m\}$ is a linearly independent set of vectors in R^n. Then the closed convex hull of V denoted by $S =< V >=< X_0, X_1, \ldots, X_m >$ is called an m-simplex in R^n, while the points $X_0, \ldots, X_m$ are called *vertices of* S [14].

Note, that strictly speaking, it is not necessary to order the vertices of the simplex. We consider only the simplex as a closed set defined as above. Other authors use the order of the vertices [18].

DEFINITION 1.2 (Conforming simplex mesh). Let Ω be a bounded set in R^n with non-empty interior, $\overset{\circ}{\Omega} \neq \emptyset$, and consider a partition of Ω into a set $\tau = \{t_1, \ldots, t_n\}$ of simplices. That is: (I) $\Omega = \cup t_i$; (II) $\overset{\circ}{t_i} \cap \overset{\circ}{t_j} = \emptyset$ if $i \neq j$; (III) $\overset{\circ}{t_i} \neq \emptyset$; such that any adjacent simplex elements share an entire face or edge or a common vertex, i.e. (IV) there are no *non-conforming* nodes in τ (see Figure 2). Under these conditions we will say that τ is a conforming simplex mesh for Ω, a conforming triangulation in two dimensions, and in three dimensions a conforming tessellation or also as a 3D conforming triangulation.

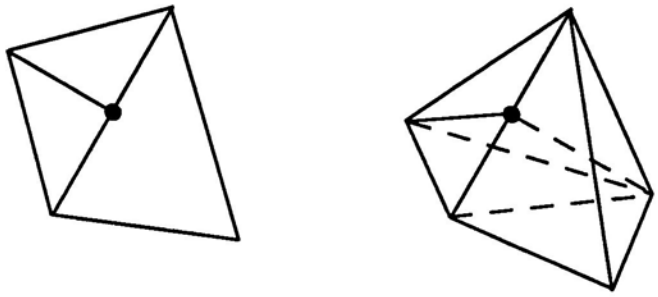


FIG. 2. *Non-conforming node in 2 and in 3 dimensions*

DEFINITION 1.3 (Skeleton). Let Ω be a bounded set in R^n with non-empty interior and τ an n-simplicial mesh of Ω. The set $skt(\tau) = \{\partial(t_i) : t_i \in \tau\}$ will be called *skeleton* or $(n-1)$-skeleton of τ, where the boundary (∂) is taken in the usual topology of R^n [6]. For instance, the skeleton of a triangulation in three dimensions is comprised of the faces of the tetrahedra, in two dimensions the skeleton is the set of edges of the triangles, and in one dimension it is the set of the points (vertices) which define the partition into segments.

The above definition of the skeleton can be applied in a recursive way. That is, the set: $skt(skt((\tau)) = \{\partial(\partial(t_i))$ *for* $t_i \in \tau\}$ where the $\partial(\partial(t_i))$ must be found in the topology of $skt(\tau)$ inherited by inclusion in R^n. In fact, if τ is an n-simplicial mesh, $skt(skt(\tau))$ will be called the $(n-2)$-*skeleton* of τ, and so on. Note that this concept is not related to the *medial object* or *skeleton* used in the literature [27, 12].

DEFINITION 1.4 (Edge bisection). Let $S =< X_0, X_1, \ldots, X_n >$ be an n-simplex in R^n, with edge $< X_k, X_{k'} >$ having midpoint $A = (X_k + X_{k'})/2$. Then two new simplices

$$S_1 =< X_0, \ldots, X_{k-1}, A, X_{k+1}, \ldots X_{k'} \ldots X_n >$$
$$S_2 =< X_0, \ldots, X_k, \ldots, X_{k'-1}, A, X_{k'+1}, \ldots X_n >$$

may be formed such that the interiors are disjoint and $S = S_1 \cup S_2$. This defines a subdivision of S by edge bisection or a simple bisection [14, 28]. Frequently $< X_k, X_{k'} >$ is chosen as the longest edge of S [6, 32, 17]. Then it said that a generalized bisection has been performed.

LEMMA 1.5. *The bisection of an n-simplex S by the longest edge or by some selected edge induces the bisection of all k-simplices in S that contain the selected edge.*

The goal here is to define a 3D refinement algorithm after a 2D refinement algorithm has been applied to the skeleton of the 3D triangulation. Both of the algorithms will be based on bisection. Moreover, both of them are based on a certain classification of the edges defining the successive bisections. Note that from the last definition and from the lemma, if we know the bisection edges, and the order in which they are taken for subdividing the tetrahedron, the subdivision is already defined. In our case, this order is based on the length of the edges.

1.2. The 2D case. The (non-recursive) 4T refinement algorithm of Rivara, to refine a unique triangle t belonging to some initial triangulation τ can be described as follows [30, 32]:

INPUT: $\{t, \tau\}$
Perform the longest edge bisection of t
    (let P be the midpoint generated
    and t_1, t_2 the triangles obtained)
While P is a *non-conforming* side midpoint of the neighbor triangle t^* do
     Perform the longest edge bisection of t^*
         (let Q be the point generated)
     It $P \neq Q$ then join points P and Q
End while
For $i = 1, 2$ do
    Perform the bisection of t_i by the common side of t and t_i
        (let P be the midpoint generated)
    While P is a *non-conforming* side midpoint of the neighbor triangle t^* do
        Perform the longest edge bisection of t^*
            (let Q be the point generated)
        If $P \neq Q$ then join points P and Q
    End while
End for.
OUTPUT: τ^*

In the present work we consider a version of the 4T algorithm proposed by L. Ferragut [10]. This approach will be extended later to refine the skeleton of 3D

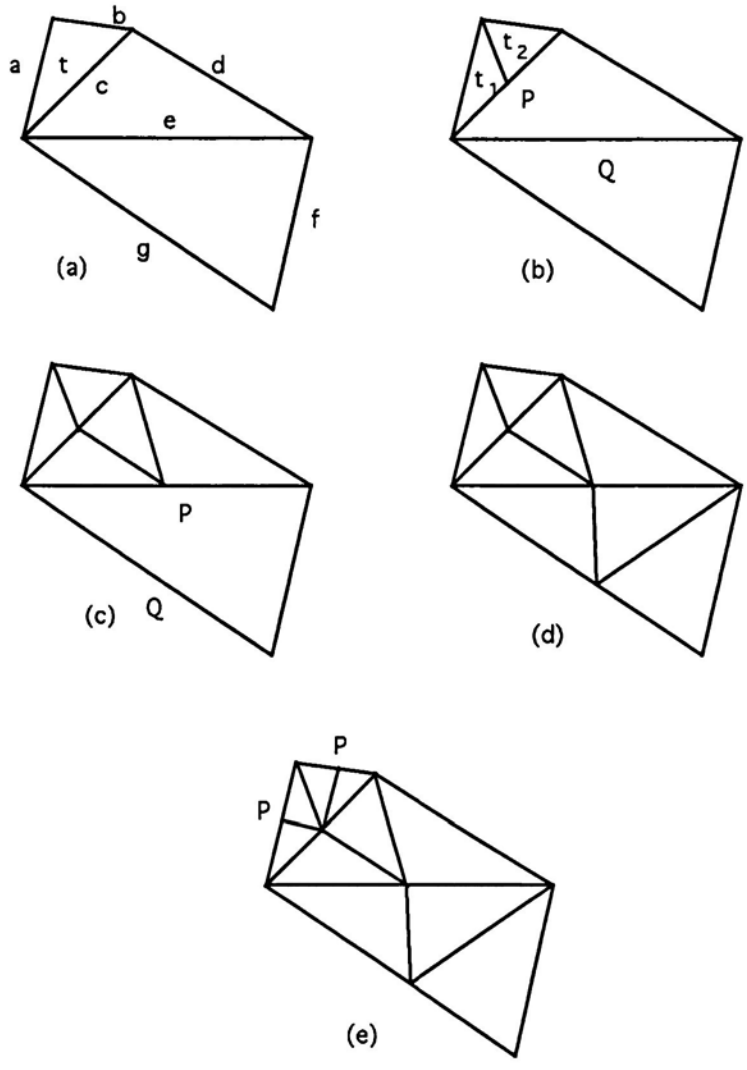


FIG. 3. *Refinement and conformity*

triangulations. Although in the algorithm by Ferragut the evaluation of the refinement condition and the conformity of the new mesh are two separate steps, other variants of the algorithm can be considered [25], in which the conformity is assured locally at the same time as the triangles are evaluated for refinement. This means that when an element t must be refined, the conformity is assured by testing the neighboring element of each edge in which a node is introduced. This version differs from Rivara's approach in the following points: i) when the triangles are taken for evaluation of the refinement condition, the edges in which a node is going to be introduced are marked. ii) after the conformity has been ensured, each triangle is divided following a suitable pattern.

The performance of the 4T algorithm of Rivara is indicated in Figure 3, in which (a) is the original triangulation, (b) the longest edge bisection of t, (c) and (d) the extension for conformity, and (e) the final triangulation in which sub-triangles t_1 and

t_2 have been subdivided.

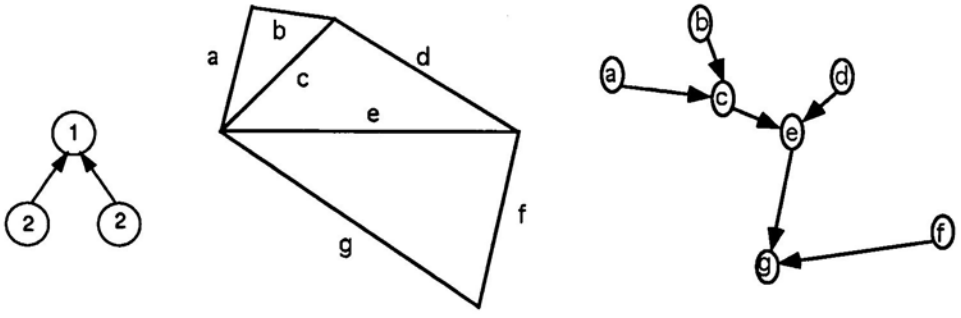


FIG. 4. *Oriented graph representing the classification of the edges of a triangle, 2D mesh and associated graph*

The 4T algorithm has the advantage that the different possibilities for refining a triangle depend only on determining the longest edge. Therefore, we can say that this algorithm classifies the edges of each triangle in two types, the longest one, type 1, and the other two edges, type 2. In this way, an oriented graph, such as that shown in Figure 4 can be associated with each triangle. The arrows indicate the dependency of the edges when refining to enforce conformity. Note that because an edge can be the longest one in one triangle but not the longest edge in the neighboring triangle, this classification is local to each triangle.

Now we show how the edges on each tetrahedron can be consistently classified with the order of the edges of each triangular face. In this way, the refinement of the skeleton by means of our 4T algorithm will be the trace of the 3D refinement obtained by bisection, following the order of the edges chosen for subdivision.

2. Classification of the tetrahedra. Let us consider a generic tetrahedron. We can number its nodes and its edges as in Figure 5 so that edge 3-4 is the longest one. The faces are numbered with the number of the opposite vertex, as usual. Introducing a criterion similar to that of Liu and Joe [17], Bänsch [5] or A. Mukherjee [21] we assign to each edge of a tetrahedron a *type*-label between 1 and 3: the longest edge of a tetrahedron is labeled type 1. Note that this edge is also the longest one of two faces. The longest edge of each one of the other two faces is edge type 2, and the rest of the edges are type 3. Edge type 1 is the reference edge of the tetrahedron.

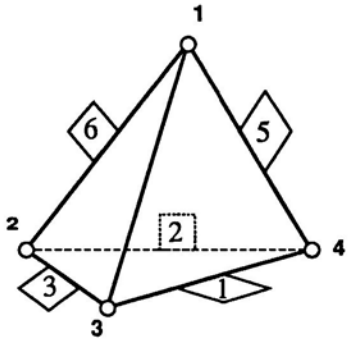


FIG. 5. *Tetrahedron in canonical position and local numeration of the vertices and edges*

Paradoxically, the selection of the longest edge of the faces and the reference edge of each element presents the most difficult case when the face or the element is regular, that is when it has all the edges of the same length. In this case we choose the first edge in which a node for division has been introduced. This detail ensures that the refinement area will not increase unnecessarily in the case of a regular initial mesh. In this way small hypothetical perturbations of the coordinates of the vertices to break ties (see for example [21]) are not necessary.

Now we classify the tetrahedra depending on the relative position of the longest edges. Obviously, we can get different configurations for refining a tetrahedron depending on the number of nodes added and also on the relative position of the nodes.

We distinguish three types of tetrahedra based on the relative position of their edge-types. This is because we will consider the subdivision of the tetrahedra in order, following their edge-type: first, we consider the tetrahedron bisected by the reference edge, second by the edge type 2, and so on.

Procedure Classification(t).
If edge 6 has type 3, then:
    t has type 1
Else if edge 6 = *longest*(face 3)=*longest*(face 4) then:
    t has type 2
Else
    t has type 3
End if

Each type of tetrahedron can be represented by an oriented graph such as that shown in Figure 6. The arrows in the graph indicate the dependency of the edges when refining for conformity. This means that if in a tetrahedron there is one type 3 edge marked for refinement, all of the edges with which it is connected in the graph must also be marked. The graph associated to a tetrahedron type 1 means that the edge opposite to the longest one is type 3 -as marked in bold in Fig. 6(a). In a tetrahedron type 2, the edge opposite to the reference edge is the only edge type 2, see Fig. 6(b). Note that in this case the graph is symmetric, so we can permute types 1 and 2. Finally, in tetrahedra type 3 the opposite edge to the longest one is type 2, but is shorter than the other edge type 2, and both are in the same face. This explains the dependency between the two edges type 2 in Fig. 6(c).

THEOREM 2.1. *There are 51 different possible configurations obtained by local or global refinement, excluding rotations.*

Proof.

Note that the simplest configuration obtained by *local* refinement is that in which only one node is introduced in the midpoint of the longest edge of the tetrahedron. We will denote this configuration as (1). *Global* refinement of a tetrahedron means that a node has been introduced in each one of its edges. We begin by pointing out the equivalence of the possible configurations that are obtained by pure rotation. Let the tetrahedron of Figure 7(a) be rotated π *radians* about an axis through the midpoints of edges 1 and 6. We obtain again the tetrahedron in canonical position, but the local numeration of its edges and nodes has changed. This is the unique rotation that preserves the canonical position and is still distinct from the identity [20]. Observe that if this rotation is denoted r, the relation between the nodes is: $r(1) = 2$; $r(2) = 1$; $r(3) = 4$ and $r(4) = 3$. Similarly for the edges $(ei : \ i = 1, \ldots, 6)$ we can write: $r(e1) = e1$; $r(e2) = e4$; $r(e3) = e5$ and $r(e6) = e6$.

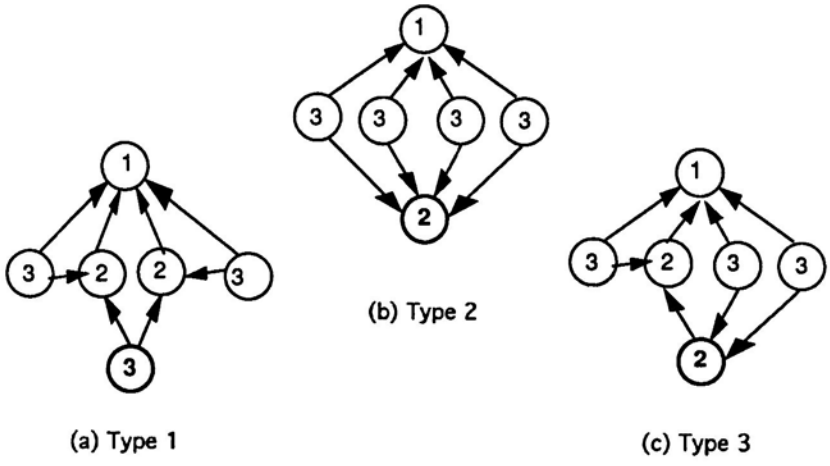


FIG. 6. *Oriented graphs representing the different classes of the tetrahedra*

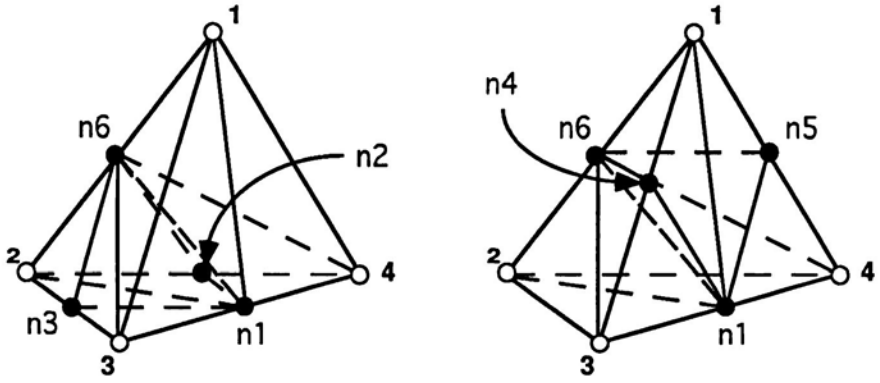


FIG. 7. *Two equivalent configurations*

Next, let us represent each configuration by means of integer sequences of the form $(i \ldots l)$ where the numbers of the edges that are subdivided are listed in order according to their types, from type 1 to type 3. For instance, the configuration of Figure 7(a) is represented as (1623) since it is characterized by midpoint nodes of edges 1, 6, 2 and 3. (Here edge 6 has type 2, since it is the second one in the sequence, and as it is the opposite one to the reference edge (1), the tetrahedron has type 2 in which 4 nodes are added.) For comparison, the configuration (1645) is shown in Figure 7(b).

In the set of integer sequences $IS = \{\sigma,\ \textit{such that}\ \sigma = (i_1, \ldots, i_k),\ \textit{where}\ k \leq 6,\ i_j \in \{1, 2, \ldots, 6\},\ i_1 = 1\ \textit{and}\ i_n \neq i_m\ \textit{if}\ n \neq m\}$ we define an equivalence relation in this way:

1) if $r(\sigma_1) = \sigma_2$, then $\sigma_1 \sim \sigma_2$

2) if $\sigma_1 = \sigma_2$ except for the order of edges with the same type non-connected in the associated graph, then $\sigma_1 \sim \sigma_2$.

Note that the point 2) is concerning about tetrahedra type 3, in which there is a relation between the two edges type 2. It follows immediately to check that the previous conditions define an equivalence relation in IS.

The proof of the theorem follows simply by counting all the equivalence classes

TABLE 1

Possible configurations obtained by tetrahedron subdividing of na edges. Integer sequences in bold indicate distinct configurations and # is the number of distinct (bold) configurations in each case.

na	Tet type 1	Tet type 2	Tet type 3	#
1	**(1)**	(1)	(1)	1
2	**(12)** ~(14) **(13)**~(15)	**(16)**	(12)~(14) (13)~(15)	3
3	**(125)**~(143) **(134)**~(152) **(123)**~(145) **(124)**~(142) **(135)**~(153)	**(162)**~(164) **(163)**~(165)	(125)~(143) (134)~(152) **(126)**~(146) **(136)**~(156)	9
4	**(1236)**~(1456) **(1234)**~(1452) **(1235)**~(1453) **(1356)** **(1354)**~(1532) **(1426)** **(1423)**~(1245)	**(1654)**~(1632) **(1625)**~(1643) **(1624)** **(1635)**	**(1265)**~(1463) **(1364)**~(1562) **(1263)**~(1465) **(1264)**~(1462) **(1362)**~(1564) **(1365)**~(1563)	17
5	**(12364)**~(14562) **(12365)**~(14563) **(12345)**~(14523) **(12463)**~(14265) **(13524)** **(13562)**~(15364) **(14235)**	**(16234)**~(16452) **(16235)**~(16453)	**(12653)**~(14635) **(12654)**~(14632) **(13645)**~(15623) **(13642)**~(15624) **(12634)**~(14652) **(13625)**~(15643)	15
6	**(123456)**~(145236) **(124356)**~(142536) **(135246)**~(153426)	**(162345)**~(164523)	**(126345)**~(146523) **(136245)**~(156423)	6
#	25	10	16	51

which are identified in the Table 1. There, the cases in bold are the distinct ones and the other cases are equivalent by rotation to bold cases. □

3. Outline of the Refinement Algorithm. Let $T^n = \{\tau_1 < \tau_2 < \ldots \tau_n\}$ be a sequence of nested three-dimensional grids, where τ_1 represents the initial mesh and τ_n the finest mesh in the sequence. To refine the sequence, or the finest mesh τ_n, implies generating a new higher-level member in the sequence to obtain: $T^{n+1} = \{\tau_1 < \tau_2 < \ldots < \tau_n < \tau_{n+1}\}$. In practice τ_{n+1} may be constructed by means of the use of a 2-dimensional refinement algorithm applied to the skeleton of τ_n as follows:

INPUT: τ_n
The refinement condition is evaluated:
For each $t \in \tau_n$, do:
1. If t must be refined, then:
   1.1. Each edge is marked to be subdivided.
   1.2. Set the mesh conformity flag, $Flag = 1$.

End for.

The conformity is ensured:
While $Flag = 1$, do:
Set $Flag = 0$.
For each $t \in \tau_n$, do:
2. If t is non-conforming, then:
2.1. Mark the longest edge of each non-conforming face of t
2.2. Procedure *reference-edge(t)*
2.3. Mark the reference-edge.
2.4. Set $Flag = 1$.
End for.
End while.
The subdivision of the mesh is performed:
For each $f \in skt(\tau_n)$, do:
3. Perform the subdivision of f by the 4-T algorithm of Rivara [29].
End for.
For each $t \in \tau_n$, do:
4. Perform the appropriate subdivision of t.
That is, one of the 51 possible configurations.
End for.
OUTPUT: Sequence $T^{n+1} = \{\tau_1 < \tau_2 < \ldots < \tau_n < \tau_{n+1}\}$.

Procedure reference-edge(t).
If there is only one edge e common longest edge for two faces
reference-edge(t)=e
Else if there are two edges, e and $e*$ common longest edges for two faces
if e has to be subdivided reference-edge(t)=e
Else if there are no common longest edge for two faces
reference-edge(t)= e, where e is the first longest edge in which a node has been introduced
Change the longest edge of the neighbor face of e, f
neot =neighbor element of t through f
call reference-edge(*neot*)
End if
End if

The salient feature in the above algorithm is the way in which mesh conformity is assured. It is worth noting that the new 3-dimensional mesh obtained by bisection will be conforming if and only if its 2-dimensional skeleton is conforming. To ensure the conformity of the skeleton, we can use, with minimal changes, step 2.3 above, from the 2-dimensional algorithm for simplicial subdivision. So, if some node is introduced in a tetrahedron, the node on the midpoint of its longest edge is also introduced, as in the 4T algorithm of Rivara. This is congruent with the application of the 4T algorithm at the faces, except in the case of a tetrahedron type 2 in which only one at the midpoint of the edge number 6 is introduced.

As an example, Figure 8 shows the application of the algorithm to a very simple initial mesh comprised of only 2 tetrahedra (Figure 8 a)). Tetrahedron 1-2-3-4 is refined based on the error indicators (Figure 8 b) and the other one is refined to satisfy conformity (see Figure 8 (c), in which nodes $N1$ and $N2$ are introduced).

Figure 8 (d) presents the division of the 2D skeleton and Figure 8 (e) the final mesh in which the new elements have been defined.

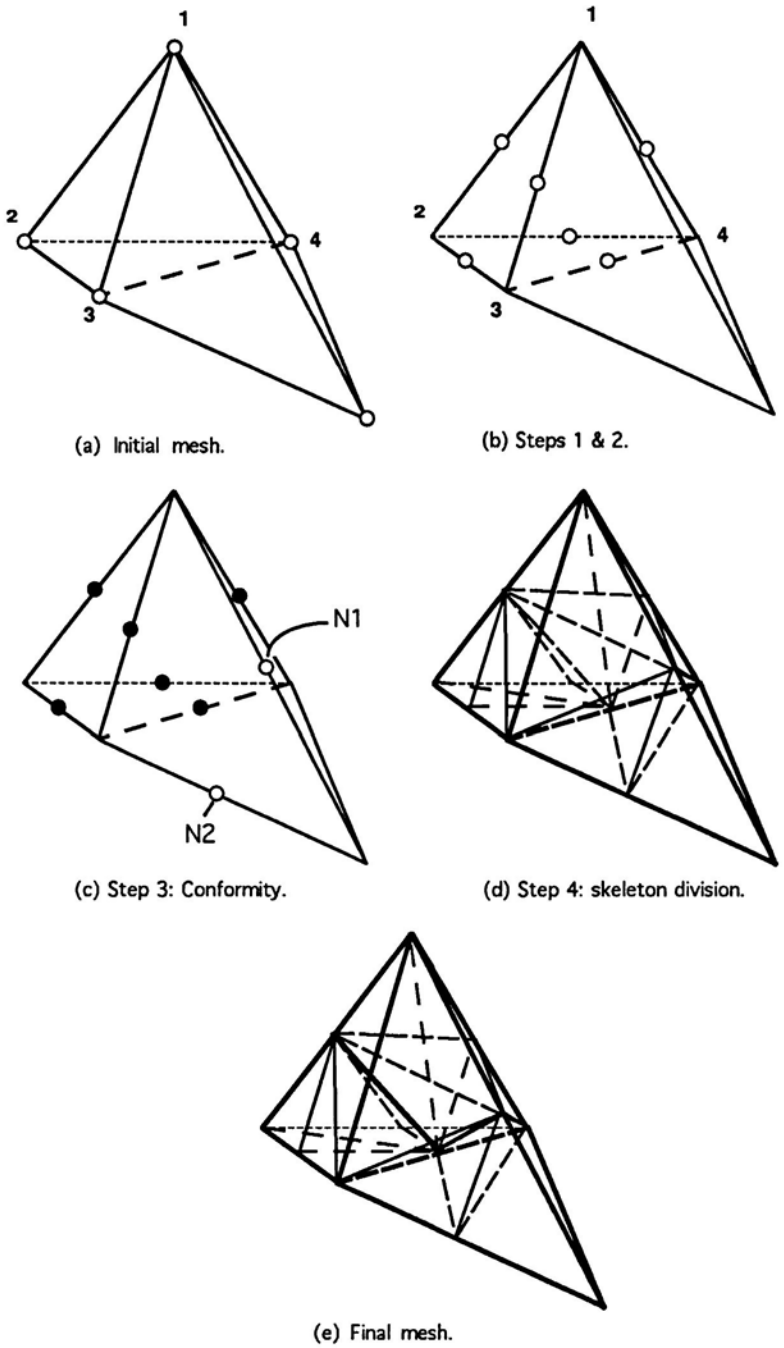


FIG. 8. *Example of application of the 3D algorithm*

As has been noted already, the selection of the longest edge of the faces and elements may complicate matters when the faces or elements are regular. Choosing the

first edge in which a node has been introduced, we are able to assure that incompatibilities between the longest edge of the faces and the longest edge of each tetrahedron do not exist. Figure 9 shows one of these situations. Suppose that the faces are numbered with the number of the opposite vertex as usual. There the longest edge of each face is marked with the number of the corresponding face. This is only possible if the lengths of some edges are equal.

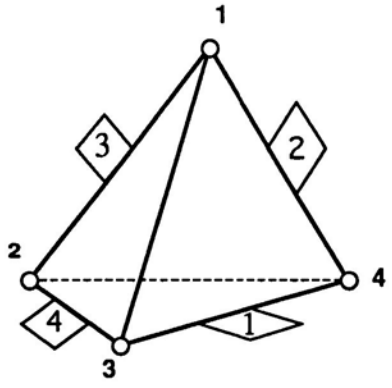


FIG. 9. *Example of non-decidable longest edge of the tetrahedron*

What is, in this case, the longest edge of the tetrahedron? One approach is to change the longest edge of one of the triangular faces such that two faces share the same longest edge. But, if the longest edge of some face is changed, we must go to the adjacent tetrahedron and verify that its longest edge must be changed as well. This implies an iterative process such as Procedure reference-edge.

In the final stage, step 4 of the algorithm, only the local subdivision of each 3-simplex of the original mesh must be made. An iterative method similar to that employed in [17] could be used. As we may implement different subdivisions depending on the number of points introduced and their relative position, it is very important to classify the tetrahedra as has been described.

THEOREM 3.1. *The above algorithm is finite.*

Proof. This result follows directly from the fact that the corresponding two dimensional 4-T algorithm [29] is finite, and the number of possible configurations for subdividing a tetrahedron are also finite. The use of a global criteria in step 2.2 of the algorithm assures that the procedure for selecting the reference-edge of the tetrahedron is also finite.

Geometrically, the conforming process can be viewed as the refinement of a set of polyhedral neighbor sets, where each has its axis longer than the preceding one. This property ensures that the algorithm terminates with a conforming mesh in a finite number of steps, in the same way as the algorithm described in [32]. □

THEOREM 3.2. *The subdivision of the faces is congruent with the internal subdivision of the tetrahedra.*

Proof. Note that all possible situations for refining the boundary of each tetrahedron are based on the relation between the longest edge of each face, since we are using the 4T Rivara algorithm. This refinement at the boundary can also be achieved by subdivision of the tetrahedron following the order of its marked edges, taking in account its type and the corresponding graph. □

COROLLARY 3.3. *The meshes obtained by application of the previous algorithm are conforming.*

The data structure is similar to that used in [11]. With this data structure implementation of the corresponding derefinement algorithm and of the multi-grid method are relatively simple.

Remark: The n-dimensional Algorithm.

The above ideas can be generalized to grids comprised of n- simplices in n dimensions. Although the n-dimensional case must be studied more carefully, we can say that the mesh conformity would be assured from conformity of the immediately lower dimensional skeleton:

$$1\text{-}skt(\tau_m) \subset 2\text{-}skt(\tau_m) \subset \ldots \subset (n-1)\text{-}skt(\tau_m)$$

where k-$skt(\tau_m)$ denotes the k-dimensional skeleton.

The number of possible configurations obtained by subdividing a simplex obviously continues to increase with the dimension just as observed in progression from the triangle to the tetrahedron. Hence, an implementation of the algorithm in which only the ordered list of edges that characterizes each subdivision is taken into account to perform the subdivision, seems to be more appealing.

It can be noted here that in a general n-dimensional simplex we need n indices to classify the edges by their length: Hence for the n-dimensional simplex we have:

- type 1: the longest edge of the n-simplex.
- type k: the longest edge of each $(n-k)$-simplex of the corresponding $(n-k)$-skeleton.
- type n: the remaining edges.

4. Complexity of the algorithm. Consider the 3D case. The number of elements of τ_m has the order of the number of vertices, N, so steps 1 and 2 of the algorithm in Section 2.1 have a complexity of $O(N)$. In assuring the conformity of the skeleton we are not using the 2D refinement algorithm presented in Reference [25] since we have to check each element again if some node is added in step 4.1. This implies as many loops on elements as vertices are added *to assure conformity*, so the complexity is at most $O(N^2)$ for local refinement; however it is $O(N)$ when global refinements are performed. Finally, steps 4 and 5 present a complexity of $O(N)$. Combining these estimates the complexity in 3D is

$$O(N) + O(N^2) + O(N) = O(N^2) \text{ for local refinement,}$$
$$O(N) + O(N) + O(N) = O(N) \text{ for global refinement.}$$

It is worth noting here that the above study provides an upper bound of the number of operations required by the algorithm. Numerical experiments indicate that, in practice, the algorithm performs like a linear complexity algorithm. Nevertheless, we present in the following paragraphs another version of the algorithm that exhibits a linear complexity with respect to the worst case: $O(N)$.

A better complexity will be obtained if the conformity test can be improved to $O(N)$. This can be accomplished at the storage cost of introducing a new vector of elements belonging to the *hull* of each edge *IT[1:A,1:NUME]*, where *NUME* is the actual number of edges in the mesh and A is the maximum number of tetrahedra sharing the same edge. Note that in general the hull $h(E)$ of an edge relative to the triangulation is not a convex set. In addition, we will need the vector of *surronding edges* for each node *IEX[NUMN]*. This vector defines for each node the edge of which it is in the midpoint. As in previous algorithms [25] this vector will also be used when coarsening the mesh. We recall that one of the vectors in our data structure is *IMNODE(NUMN)* for the nodes. In *IMNODE* only the *proper* nodes of each level are kept.

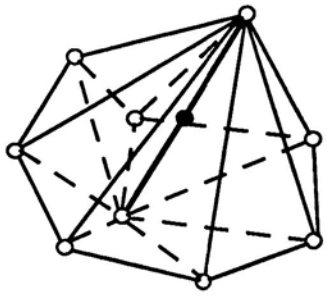


FIG. 10. *Hull of an edge relative to the tessellation*

The previous algorithm can now be modified to assure the associated conformity as follows:

For each *new-node* N let E be its surrounding edge, then:
  For each tetrahedron t in $h(E)$, do:
  1. If t is non-conforming, then:
    1.1. A node is added at the midpoint
      of the longest edge of each non-conforming
      face of t.
    1.2. Add a node to the midpoint of the
      longest edge of t.
    End if.
  End for.
End for.

This procedure to assure the conformity has a complexity of $O(N_a)$, where N_a is the number of added nodes due to refining. Since $\sum N_a < N$ the linear complexity of the algorithm is proved.

5. Quality of the meshes. Many parameters can be chosen for measuring the quality of a mesh in 3 or higher dimensions. Here we refer to such measures as estimates of "shape" of the tetrahedra. For example, Whitehead [34] used the *relative thickness* of a simplex S:

$$\rho(S) = \frac{r(S)}{d(S)} , \tag{1}$$

where $r(S)$ is the *radius* of S, that is, the distance from its centroid to its boundary, and $d(S)$ is the diameter of S, that is, the length of its longest edge. Stynes [33] introduced

$$t(S) = \frac{area(S)}{d^2(S)} \tag{2}$$

for the bisection method applied to triangles. This ratio can be generalized easily to higher dimensions as

$$t(S) = \frac{volume(S)}{d^n(S)} , \tag{3}$$

where n is the dimension of the simplex S. Other authors (see for example, [9]) have used

$$ratio(S) = \frac{r(S)}{R(S)}\ , \tag{4}$$

where $r(S)$ is now the length of the radius of the inscribed sphere and $R(S)$ is the radius of the circumsphere. Similarly, the eccentricity [9] of polyhedra, i.e. the ratio of the lengths of the longest and shortest edge, can be similarly computed to estimate shape quality.

For tetrahedron T and each vertex P of T, Rivara and Levin [32] use the solid angle at P to define

$$\Phi_T = min\{sin^{-1}(1 - cos^2\alpha_P - cos^2\beta_P - cos^2\gamma_P + 2\ cos\alpha_P\ cos\beta_P\ cos\gamma_P)^{1/2}\} \tag{5}$$

Moreover, for each conforming mesh, τ, they assign the number $\Phi_\tau = \min \Phi_T$ to obtain a global measure for the mesh. With these parameters they classify the tetrahedra into four classes and offer some numerical results regarding the evolution of *bad tetrahedra* when certain refinements are performed.

Liu and Joe [16, 17] introduce the estimate

$$\eta(T) = \frac{3\sqrt[3]{\lambda_1\lambda_2\lambda_3}}{\lambda_1 + \lambda_2 + \lambda_3}\ , \tag{6}$$

where λ_i are the eigenvalues of the matrix $A(R, M) = (MR^{-1})^t MR^{-1}$ for matrices M and R associated with a given tetrahedron T and a regular tetrahedron with the same volume as T. The columns of M are the vectors corresponding to the edges joining a vertex to the other vertices of T and R is similarly defined for the reference tetrahedron. They prove that $\eta(T)$ is independent of the order of the vertices and that:

$$\eta(T) = \frac{12\ (volume)^{2/3}}{\sum_{i=1}^{6} l_i^2}\ , \tag{7}$$

where l_i means the length of the edge i.

Our objective in the following numerical experiments is to compare these parameters in representative examples in order to study the change in shape quality of the tetrahedra as successive local refinements are carried out with the algorithm of section 3.

6. Numerical examples. Here we present some refinements case studies to show the behavior of the 3D algorithm. All the calculations were performed on a Sun-4 workstation with the f77 compiler optimization option on.

The first example corresponds to global refinement of a mesh comprised initially of two regular tetrahedra. Figure 11 presents the CPU times versus the number of added nodes in each step of refinement. Five global refinements were performed and the finest mesh has 1785 nodes and 8192 tetrahedra.

The second example is for local refinement towards an edge in a prismatic domain. Here, 40 local refinements were done to obtain 864 nodes and 3878 tetrahedra in the finest mesh. The CPU times are shown in Figure 12. As a final example, Figure 13 illustrates local refinement toward a vertex in a 3D domain. The mesh 13 (c) contains 251 nodes and 1016 tetrahedra. The CPU time increases in the manner shown in Figure 14. Note the slightly quadratic behavior of the first algorithm.

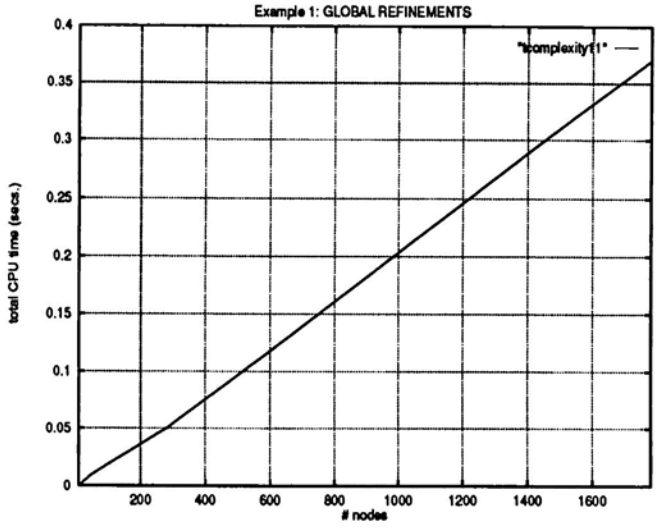


FIG. 11. *Global refinements show linear complexity*

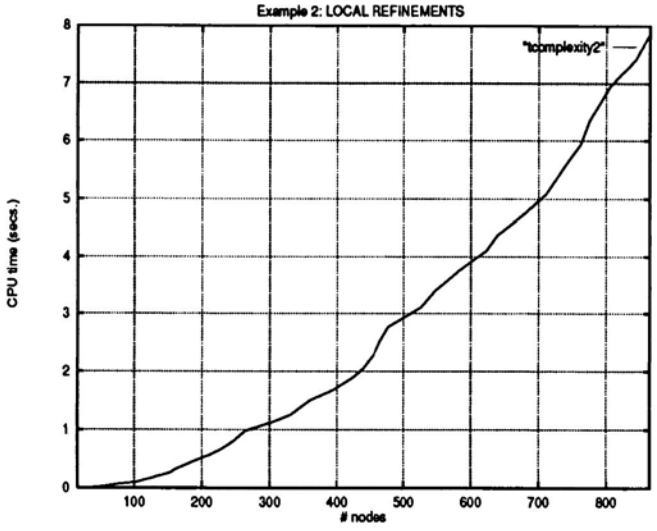
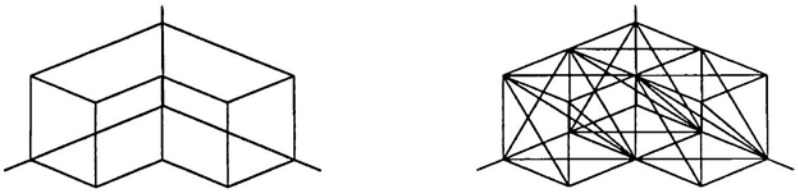
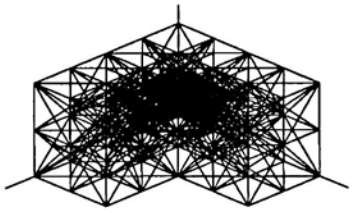


FIG. 12. *Local refinements*

In Table 2 the first four problems in [32] are listed in terms of the coordinates of their four vertices. They are also considered in [17]. P1 and P2 are well-shaped tetrahedra; P3 is a poorly shaped tetrahedron and P4 is the regular tetrahedron.



(a) Domain (b) Initial mesh



(c) After 6 refinements

FIG. 13. *Local refinement at a vertex*

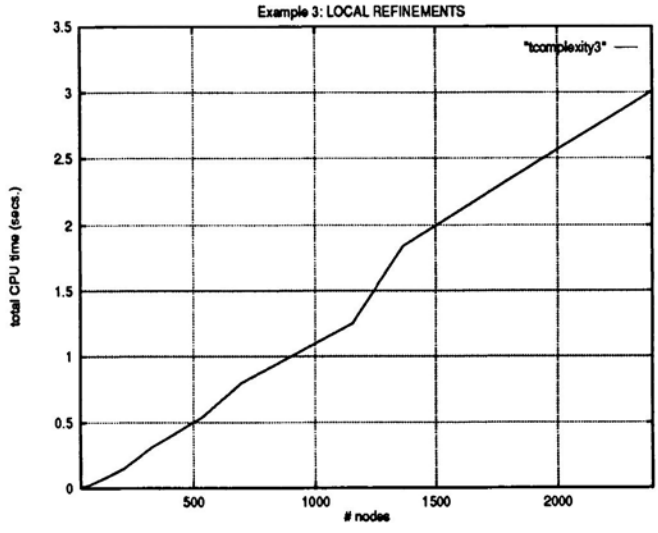


FIG. 14. *Local refinements in an L-shape domain*

TABLE 2
Problems 1 to 4

P1 $\eta = 0.8846$			P2 $\eta = 0.8399$			P3 $\eta = 0.2835$			P4 $\eta = 1.0000$		
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4.0	2.0	2.0	4.0	0.0	0.0	0.5	0.0	0.0	$2\sqrt{3}$	0.0	0.0
1.0	5.0	0.0	0.0	4.0	0.0	1.5	5.0	2.0	$\sqrt{3}$	3.0	0.0
0.5	0.5	5.0	0.0	0.0	4.0	0.5	0.5	5.0	$\sqrt{3}$	1.0	$\sqrt{3}$

The left parts of Table 3 show the results reproduced based on the longest edge bisection [32]. The middle part of the Table 3 are for QLRB in [17]. Note that the results using our algorithm in the right column are comparable to those by the longest edge bisection of Rivara and Levin [32] and the QLRB algorithm of Liu and Joe [17].

TABLE 3
Comparison of problems 1 to 4

Problem 1.

	Longest edge bisection			*QLRB*		*Based on Skeleton*		
level	*NTET*	η_{min}	ϕ_{min}	*NTET*	η_{min}	*NTET*	η_{min}	ϕ_{min}
1	1	.885	31.39	1	.885	1	.885	31.39
2	8	.682	18.19	8	.682	8	.682	18.19
3	86	.479	8.33	64	.663	64	.571	13.84

Problem 2.

	Longest edge bisection			*QLRB*		*Based on Skeleton*		
level	*NTET*	η_{min}	ϕ_{min}	*NTET*	η_{min}	*NTET*	η_{min}	ϕ_{min}
1	1	.840	30.00	1	.840	1	.840	30.00
2	8	.657	16.78	8	.504	8	.504	9.59
3	86	.458	7.42	64	.504	64	.504	9.59

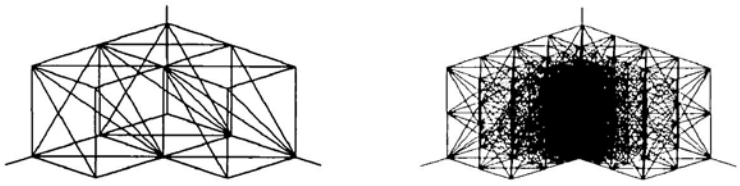
Problem 3.

	Longest edge bisection			*QLRB*		*Based on Skeleton*		
level	*NTET*	η_{min}	ϕ_{min}	*NTET*	η_{min}	*NTET*	η_{min}	ϕ_{min}
1	1	.284	4.28	1	.284	1	.284	4.28
2	8	.181	2.62	8	.181	8	.181	2.62
3	192	.165	1.59	64	.170	64	.163	1.59

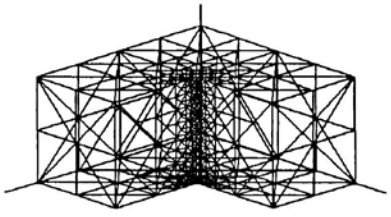
Problem 4.

	Longest edge bisection			*QLRB*		*Based on Skeleton*		
level	*NTET*	η_{min}	ϕ_{min}	*NTET*	η_{min}	*NTET*	η_{min}	ϕ_{min}
1	1	1.00	45.00	1	1.00	1	1.00	45.00
2	8	.546	13.63	8	.546	8	.546	13.63
3	306	.458	7.42	64	.429	64	.429	7.42

Figure 15 shows local refinements of a 3D domain in which a singularity on the interior corner edge is assumed. The mesh 15(b) contains 1365 nodes and 6657 tetrahedra.



(a) Initial mesh (b) After 10 local refinements



(c) Detail of the refinement on the surface

FIG. 15. *Local refinement at a re-entrant corner*

The evolution of the shape measurements through the sequence of 10 refinements can be observed in Figure 16. Here *Min sol ang* is the minimum solid angle, *Min pla ang* is the minimum planar angle, *Min Styn* is the minimum value of (2), *Min Whit* is the minimum of (1),*min etha* is the minimum of (7), and *minratio r/R* is the minimum of the ratio (4).

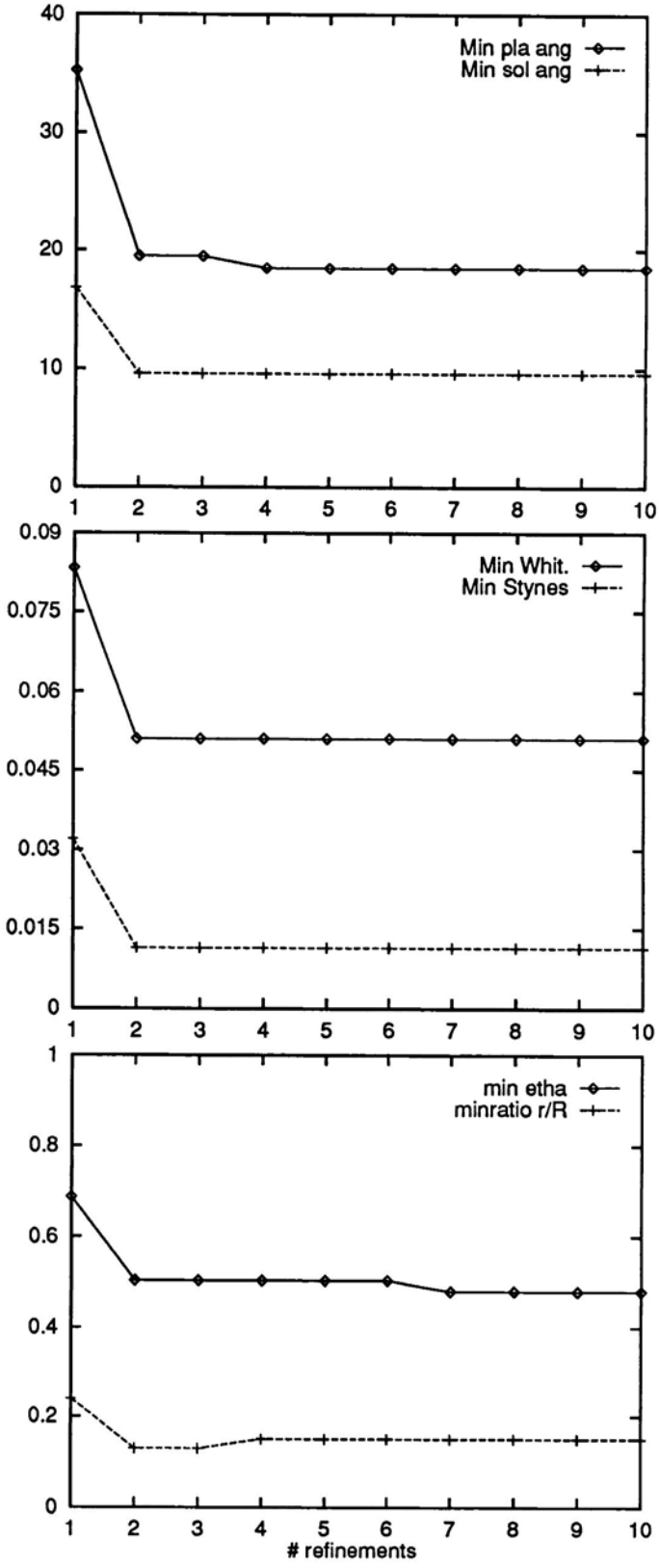


FIG. 16. *Evolution of shape measures for example of Figure 15*

7. Conclusions. The ideas presented here provide a framework for studying local refinement in 3D and extend the ideas developed by Rivara and other researchers. In fact the way in which a 3-dimensional refinement is deduced from the application of the similar 2-dimensional scheme can be recursively generalized to any dimension. The first algorithm presented here has a higher complexity, in terms of an upper bound of the number of operations required, than the one developed by Liu and Joe [17, 16]. This study of complexity suggests an improved version. In this case, the use of a new vector of elements belonging to the hull of each edge reduces the algorithm complexity to linear.

The edges of each tetrahedron are ordered in a compatible way with the order in each face, and this makes it possible to write the algorithms so that only this order has to be taken into account to perform iteratively the subdivision of each tetrahedron.

The results obtained seem to confirm that measure of degeneracy is bounded, and converges asymptotically to a fixed value. Even though we do not have a mathematical proof of this property, the algorithm developed in this paper has shown itself to be in practice a powerful tool for the refinement of tetrahedral grids. Moreover, as in the 2D case, in 3D it is also possible to develop *inverse algorithms* for the derefinement of meshes created by means of the refinement algorithm [26]. This is an important feature for dealing with time dependent problems.

These algorithms based on bisection can be extended in such a way that the refinement algorithm yields in a sequence of refined meshes, which though physically non-nested, still retain an underlying logical refinement structure [4].

REFERENCES

[1] I. Babuška and W. C. Rheinboldt, *Error Estimates for Adaptive Finite Element Computations*, SIAM J. Numer. Anal. 15, (1978), pp 736–754.

[2] I. Babuška and W. C. Rheinboldt, *A Posteriori Error Analysis of Finite Element Solutions for One-Dimensional Problems*, SIAM J. Numer. Anal. 18, (1981), pp 565–589.

[3] R. E.Bank and A. H. Sherman, *The use of adaptive grid refinement for badly behaved elliptic partial differential equations*, in Advances in Computer Methods for Partial Differential Equations III, R.Vichnevetsky and R. S. Stepleman, eds, IMACS, 1979, pp. 33–39.

[4] R. E. Bank and J. Xu, *An algorithm for coarsening unstructured meshes*, Num. Math., 73 (1996), pp. 1–36.

[5] E. Bänsch, *Local mesh refinement in 2 and 3 dimensions*, IMPACT Com. Sci. Eng., 3 (1991), pp. 181–191.

[6] M. Berger, *Geometry*, Springer-Verlag, 1987.

[7] J. Bey, *Tetrahedral grid refinement*, Computing 15 (1995), pp. 355–378.

[8] G. F. Carey, *Computational Grids: Generation, Refinement and Solution Strategies*, Taylor and Francis (in press), 1997.

[9] P. Conti, N. Hitschfeld and W. Fichtner, Ω *-an octree-based mixed element grid allocator for the simulation of complex 3D device structures*, IEEE Trans. Com.-Aided Des., 10 (1991), pp 1231–1241.

[10] L. Ferragut, NEPTUNO, *A system of adaptive finite element method*, (in FORTRAN) Dept. Mat. Aplic. y Met. Inf., ETSI Minas, Madrid, 1987.

[11] L. Ferragut, R. Montenegro and A. Plaza, *Efficient refinement/derefinement algorithm of nested meshes to solve evolution problems*, Comm. Num. Meth. Eng., 10 (1994), pp. 403–412.

[12] C. M. Hoffmann, *How to construct the skeleton of CSG objects*, in 4th IMA Conference on Mathematics of Surfaces, Bath, England, (1990).

[13] J. P. S. R. Gago, D. W. Kelly, and O. C. Zienkiewicz, *A Posteriori Error Analysis and Adaptive Processes in the Finite Element Method. Part II: Adaptive Mesh Refinement*, *IJNME* 19 (1983) pp 1621–1656.

[14] B. Kearfott, *A proof of convergence and an error bound for the method of bisection in* R^n, Math. Comp., 32 (1978), pp. 11147–1153.

[15] I. KOSSACZKÝ, *A recursive approach to local mesh refinement in two and three dimensions*, J. Comp. App. Math., 55 (1994), pp. 275–288.

[16] A. LIU AND B. JOE, *On the shape of tetrahedra from bisection*, Math. Comp., 63 (1994), pp. 141–154.

[17] A. LIU AND B. JOE, *Quality local refinement of tetrahedral meshes based on bisection*, SIAM J. Sci. Comput., 16 (1995), pp. 1269–1291.

[18] J. M. MAUBACH, *Local bisection refinement for n-simplicial grids generated by reflection*, SIAM J. Sci. Stat. Comp., 16 (1995), pp. 210–227.

[19] W. F. MITCHELL, *Optimal multilevel iterative methods for adaptive grids*, SIAM J. Sci. Statist. Comput. 13 (1992), pp. 146–167.

[20] J. M. MONTESINOS, *Classical Tessellations and Three-Manifolds*, Springer-Verlag, 1987.

[21] A. MUKHERJEE, *An adaptive finite element code for elliptic boundary value problems in three dimensions with applications in numerical relativity*, Phd. Thesis, Penn. State University, University Park, PA 16802, 1996.

[22] MUTHUKRISHNAN, E., SHIAKOLAS,P. S., NAMBIAR, R. V., AND LAWRENCE, K. L., *Simple algorithm for Adaptive Refinement of three-dimensional finite element tetrahedral meshes*, AIAA Journal, 33, 1995, pp. 928–932.

[23] A. PLAZA AND G.F. CAREY, *About local refinement of tetrahedral grids based on bisection*, in 5th International Meshing Rountable, Pittsburgh, Pennsylvania, (1996).

[24] A. PLAZA AND G.F. CAREY, *On local refinement of tetrahedral grids*, SIAM J. Sci. Comp., to appear.

[25] A. PLAZA, R. MONTENEGRO AND L. FERRAGUT, *An improved derefinement algorithm of nested meshes*, in Advances in Post and Preprocessing for Finite Element Technology, M. Papadrakakis ed., Civil-Comp Ltd., 1994, pp. 175–180.

[26] A. PLAZA, M. A. PADRÓN, AND G.F. CAREY, *A 3D derefinement algorithm of tetrahedral grids from a 2D one on the skeleton*, in McNU'97 Symposium on Trends in Unstructured Mesh Generation, Evanston, Illynois, (1997).

[27] M. PRICE, C. STOPS AND G. BUTLIN, *A medial object toolkit for meshing and other applications*, in 4th International Meshing Roundtable, Albuquerque, New Mexico, (1995).

[28] M. C. RIVARA, *Mesh refinement based on the generalized bisection of simplices*, SIAM J. Numer. Anal., 2 (1984), pp. 604–613.

[29] M. C. RIVARA, *A grid generator based on 4-triangles conforming mesh refinement algorithms*, Int. J. Num. Meth. Eng., 24 (1987), pp. 1343–1354.

[30] M. C. RIVARA, *Selective refinement/derefinement algorithms for sequences nested triangulations*, Int. J. Num. Meth. Eng., 28 (1989), pp. 2889–2906.

[31] M. C. RIVARA, *Local modification of meshes for adaptive and/or multigrid finite-element methods*, J. Comp. and Appl. Math., 36 (1991), pp. 79–89.

[32] M. C. RIVARA AND C. LEVIN, *A 3-d refinement algorithm suitable for adaptive and multi-grid techniques*, J. Comp. and Appl. Math., 8 (1992), pp. 281–290.

[33] M. STYNES, *On faster convergence of the bisection method for all triangles*, Math. Comp., 35 (1980), pp 1995–1201.

[34] J. H. C. WHITEHEAD, *On C^1-complexes*, Ann. Math., 41 (1940), pp 809–824.