ELSEVIER

Applied Numerical Mathematics 32 (2000) 195–218

APPLIED NUMERICAL MATHEMATICS

www.elsevier.nl/locate/apnum

Local refinement of simplicial grids based on the skeleton ☆

A. Plaza [a,1], G.F. Carey [b,*]

[a] *Department of Mathematics, University of Las Palmas de Gran Canaria, Las Palmas, Spain*

[b] *Texas Institute for Computational and Applied Mathematics (TICAM), ASE/EM Department, University of Texas at Austin, Austin, TX 78712-1085, USA*

Abstract

In this paper we present a novel approach to the development of a class of local simplicial refinement strategies. The algorithm in two dimensions first subdivides certain edges. Then each triangle, if refined, is subdivided in two, three or four subelements depending on the previous division of its edges. Similarly, in three dimensions the algorithm begins by subdividing the two-dimensional triangulation composed by the faces of the tetrahedra (the skeleton) and then subdividing each tetrahedron in a compatible manner with the division of the faces. The complexity of the algorithm is linear in the number of added nodes. The algorithm is fully automatic and has been implemented to achieve global as well as local refinements. The numerical results obtained appear to confirm that the measure of degeneracy of subtetrahedra is bounded, and converges asymptotically to a fixed value when the refinement proceeds.

Keywords: Grid refinement; 3D bisection; Tetrahedra; Adaptivity

1. Introduction

Both mesh refinement and mesh redistribution are popular techniques for improving a mesh. These concepts and related ideas are discussed in [11]. In the present work we confine our attention to the local mesh refinement problem and, in particular, to an approach that is based on refining the skeleton of a simplex mesh. 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 under uniform refinement. As is well known, there are two main steps in local adaptive refinement [11]: the refinement of a subset of elements based on local error indicators [1,2,18], and the consistent transition between refined and unrefined cells. We refer to the latter as 'mesh conformity' [26,35]. The elements that offer

☆ This research has been supported by DGICYES grant number PR95-280, WES grant number NRC-CR-97-0002, and ASCI grant number B347883.

* Corresponding author. E-mail: carey@cfdlab.ae.utexas.edu

[1] E-mail: aplaza@dma.ulpgc.es

0168-9274/00/$20.00
PII: S0168-9274(99)00022-7

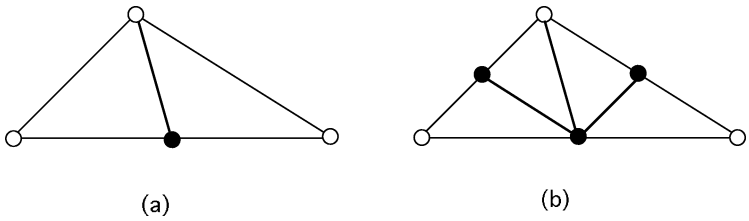


Fig. 1. 4-T division of Rivara.

the simplest choice in any dimension are the simplices: triangles in two dimensions, tetrahedra in three dimensions and their analogs in even higher dimensions. Several different refinement and improvement techniques for two- and three-dimensional triangulations are now available. In what follows we review some of the main refinement algorithms in two and three dimensions.

In two dimensions, Bank and Sherman [3] refine locally by subdividing certain triangles into four similar subtriangles by connecting the midpoints of the sides of the 'parent' triangle. The conformity of the mesh is ensured by appropriate subdivision of adjacent elements. Some schemes are based on edge bisection, e.g., the 4-T algorithm of Rivara [38]: first the longest edge is bisected, and the midedge point is connected to the vertex opposite, and then the newly formed vertex is used to subdivide the initial triangle in four as shown in Fig. 1. In this algorithm, the angles of the triangles in a resulting locally refined grid are uniformly bounded away from 0 and π. Additional refinement of adjacent triangles is again necessary to ensure the conformity of the mesh, but this refinement is also made based on bisecting the longest edge in the Rivara algorithms.

Another related method is the 'newest vertex bisection' introduced by Mitchell in [28]. The triangle edge to be bisected is identified 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. It is worth noting that the newest vertex bisection is equivalent to the 4-T Rivara algorithm in the case in which the bisected edge coincides with the longest-edge.

There are other schemes which introduce different transition elements to interface refined and unrefined regions. Other techniques use constraints on the approximation rather than subdividing adjacent neighbor elements (e.g., see Carey [10] for an early study). We will not consider these alternative approaches here.

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. Bey [9] first connects the edges of each face triangle as in the two-dimensional Bank refinement, then cuts off four subtetrahedra at the corners which are similar to the original one. Then Bey's algorithm cuts the interior octahedron into four more subtetrahedra. Since this algorithm considers only the number of nodes at the midpoints of the edges, not the relative position of these nodes, for partial refinement of a tetrahedron there are $2^6 = 64$ possibilities. To handle the transition from refined area to non refined area he considers four patterns that cover 25 of the 64 possibilities. The remaining 39 possibilities are refined to 8 subtetrahedra (see [9] for details).

Methods based on bisection can also be devised easily to subdivide each tetrahedron in eight, but the primary stage consists in bisecting the tetrahedron in two. 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 to avoid incompatibilities. The algorithm presented by Rivara and Levin [41] is based on longest edge bisection. Mathematical proofs of the non-degeneracy of the grids created by this scheme have not yet been developed, although experiments suggest this property holds. Liu and Joe [25,26] 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. The bisection edges are chosen without computation. This order depends on the assignation of types to the new edges. In this sense it can be said that the QLRB algorithm is the generalization to 3D of the Michell algorithm. In addition, a shape measure is introduced. The number of similarity classes is proved to be bounded and therefore the meshes cannot degenerate.

Several similar studies are reported in the literature. For instance, a recursive approach is proposed by Kossaczký [23]. This algorithm imposes certain restrictions and preprocessing in the initial mesh. The 3D algorithm is equivalent to that given in [5]. Maubach [27] 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 cannot be applied for a general tetrahedral grid. An additional closure refinement is needed to avoid incompatibilities. Recently, Mukherjee [30] has presented an algorithm equivalent to [5,25], and proves the equivalence with [27].

The algorithm presented in the present study is also based on bisection. Although we show similar behavior to those cited above [5,23,25–27,30] the point of view is quite different since the three-dimensional approach is now based on the two dimensional one applied to the skeleton of the triangulation. The two dimensional version is equivalent to the 4-T Rivara algorithm, and the 3D one is the generalization to three dimensions of the 4-T algorithm. 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 based on their length. Moreover, because of the underlying spatial recursion approach, these ideas can be extended to obtain local refinement algorithms in higher dimensional spaces for such problems as relativity or Boltzmann computations.

The outline of the treatment is as follows: Next we introduce some basic notation and definitions. Then in Section 2 we present the 2D version of the algorithm, and generalize this in Section 3 to the 3D case. A complexity analysis follows in Section 4. Some numerical experiments and mesh examples to demonstrate the refinement algorithm and to assess non degeneracy of the mesh conclude the study.

1.1. Notation and definitions

Definition 1.1 (m-simplex). Let $V = \{X_1, X_2, \ldots, X_{m+1}\}$ be a set of $m+1$ points in $\mathbb{R}^n$ $(1 \leqslant m \leqslant n)$ such that

$$\{\overrightarrow{X_1X_i}\colon 2 \leqslant i \leqslant m+1\}$$

is a linearly independent set of vectors in $\mathbb{R}^n$. Then the closed convex hull of V denoted by $S = \langle V\rangle = \langle X_1, X_2, \ldots, X_{m+1}\rangle$ is called an m-dimensional simplex or an m-simplex in $\mathbb{R}^n$, while the points $X_1, \ldots, X_{m+1}$ are called *vertices of S* [22]. In this way a (closed) *tetrahedron t* is defined by an unordered quadruple $\langle X_1, X_2, X_3, X_4\rangle$ of noncoplanar vertices. An *edge* of the tetrahedron is represented by an unordered pair $\langle X_i, X_j\rangle$ of distinct vertices, and a *face f* is defined by an unordered triple $\langle X_i, X_j, X_k\rangle$ of distinct vertices. It is worth mentioning that the terms 'tetrahedron', 'face', and 'edge' do not denote the finite sets of vertices, but rather the convex hulls of those vertices. Vertices, edges, faces, and tetrahedra

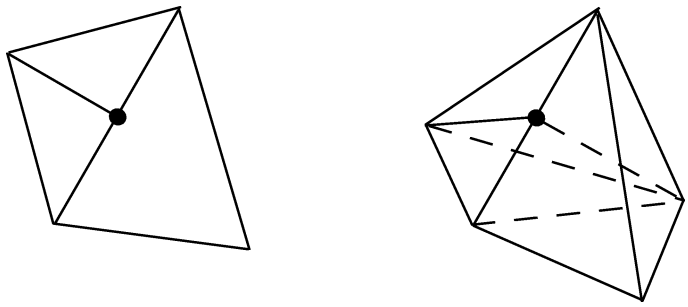


Fig. 2. Non-conforming node in 2 and in 3 dimensions.

are also referred to as 0-, 1-, 2-, and 3-*simplices*, respectively. We refer to any (k)-simplex contained in a tetrahedron t as a (k)-simplex of t.

In general, given an m-simplex $S = \langle X_1, X_2, \ldots, X_{m+1}\rangle$, a (k)-simplex obtained from S on deleting $m-k$ vertices is called a (k)-face f of S.

Definition 1.2 (Conforming simplicial mesh). Let Ω be a bounded set in $\mathbb{R}^n$ with non-empty interior, $\overset{\circ}{\Omega} \neq \emptyset$, and polygonal boundary $\partial\Omega$, and consider a partition of Ω into a set $\tau = \{t_1, \ldots, t_n\}$ of simplices with

(i) $\Omega = \bigcup t_i$;
(ii) $\overset{\circ}{t_i} \cap \overset{\circ}{t_j} = \emptyset$ if $i \neq j$;
(iii) $\overset{\circ}{t_i} \neq \emptyset$; and such that;
(iv) any adjacent simplex elements share an entire face or edge or a common vertex, i.e., there are no *non-conforming* nodes in τ [5,37] (see Fig. 2).

Under these conditions we will say that τ is a conforming simplicial mesh for Ω (a conforming triangulation in two dimensions, and a conforming tessellation in three dimensions, also called a 3D conforming triangulation).

Definition 1.3 (Skeleton). Let τ be an n-simplicial mesh. The set $\mathrm{skt}(\tau) = \{f\colon\ f$ is an $(n-1)$-face of some $t \in \tau\}$ will be called *the skeleton* or $(n-1)$-skeleton of τ [7]. 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. It should be noted however, that the skeleton can be understood as a new triangulation of the set comprised of all the $(n-1)$-faces of the initial triangulation. Note that if τ is a 3-dimensional conforming triangulation in $\mathbb{R}^3$, $\mathrm{skt}(\tau)$ is a 2-dimensional triangulation embedded in $\mathbb{R}^3$ or a $2\frac{1}{2}D$ triangulation [19].

The above definition of the skeleton can be applied in a recursive way. That is, the set $\mathrm{skt}(\mathrm{skt}(\tau)) = \{f\colon\ f$ is an $(n-2)$-face of some $s \in \mathrm{skt}(\tau)\}$. In fact, if τ is an n-simplicial mesh, $\mathrm{skt}(\mathrm{skt}(\tau))$ 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 [20,36], or the skeleton of arrangements [14], but to the topological idea of skeleton of a Euclidean complex [31].

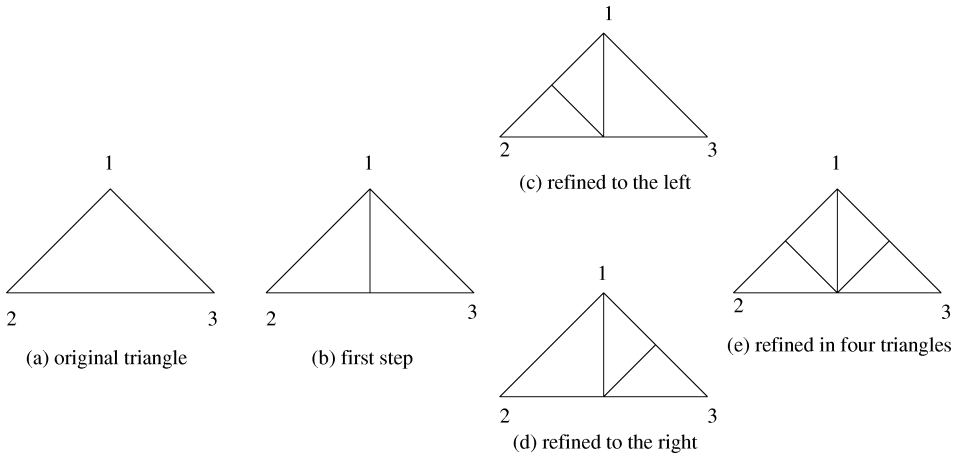


Fig. 3. Different divisions by the 2D algorithm.

Definition 1.4 (Edge bisection). Let $S = \langle X_1, X_2, \ldots, X_{m+1}\rangle$ be an m-simplex in $\mathbb{R}^n$, with edge $\langle X_j, X_k\rangle$ having midpoint $A = (X_j + X_k)/2$. Then two new simplices

$$S_1 = \langle X_1, \ldots, X_{j-1}, A, X_{j+1}, \ldots, X_k, \ldots, X_{m+1}\rangle,$$

$$S_2 = \langle X_1, \ldots, X_j, \ldots, X_{k-1}, A, X_{k+1}, \ldots, X_{m+1}\rangle$$

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 [22,37]. Frequently edge $\langle X_j, X_k\rangle$ is chosen as the longest edge of S [7,26,41].

Definition 1.5 (Configuration). Let $S = \langle X_1, X_2, \ldots, X_{m+1}\rangle$ be an m-simplex in $\mathbb{R}^n$, and suppose that S is to be refined by bisection of edges following some order. This ordered subset of edges of S suitable for local refinement will be called a configuration. Note that since we are using edge bisection, choosing a set of edges is equivalent to selecting the set of midpoint nodes that will arise from the refinement.

As an example consider the configurations corresponding to 'partial' and 'complete' refinement of a single triangle in Fig. 3. Let the longest edge be numbered 1, as the opposite vertex in the figure, and the other edges numbered 2 and 3. Then we obtain the four refinement configurations shown and these can be identified as indicated in Table 1. Here the configuration is specified by the list of edges in parenthesis, in the order in which they are taken for bisection.

Similarly, in three dimensions, with the six edges numbered locally, the list (1, 6, 2, 3) would represent a configuration, for refining a tetrahedron, in which edges number 1, 6, 2, and 3 have been chosen for bisection in that order. If there is no confusion the commas can be ommited, so, for example, the previous configuration can be also written as (1623). For more details regarding the number of possible divisions or configurations obtained in local and complete refining of a tetrahedron see the Appendix.

Lemma 1.1. *The bisection of an n-simplex S by some selected edge induces the bisection of all k-simplices in S with $1 \leqslant k \leqslant n$ that contain the selected edge.*

Table 1
Possible configurations obtained by triangle subdivision of n edges. Integer sequences in bold indicate distinct configurations and # is the number of distinct (bold) configurations in each case

n	Configurations	#
1	(1)	1
2	(1,2)	
	(1,3)	2
3	(1,2,3)	1

Proof. Consider Definition 1.1 (m-simplex and $(m-1)$-simplex) and Definition 1.4 (edge bisection). □

Lemma 1.2. *Consider the case $n = 3$. The bisection of a tetrahedron t by several edges in some specified order, induces the bisection of all the faces of t that contain any of the selected edges, and the bisection of the faces occurs in the same order as the bisection of the tetrahedron.*

Proof. Note that an order in the edges of the tetrahedron induces another order (a relative order) in the edges inside each triangular face. Lemma 1.1 gives then the proof. □

In the present scheme we seek to define a 3D refinement strategy 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 are based on a certain classification of the edges defining the successive bisections. Note that from the last definition and from the lemmas, 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, as in the 4-T Rivara algorithm, this order is based on the length of the edges.

2. The 2D case

2.1. The 4-T Rivara algorithm

For clarity of exposition it is useful first, to recall the (non-recursive) 4-T refinement algorithm of Rivara applied to a triangle t_0 belonging to some initial triangulation τ [39,41]:

The 4-T Rivara algorithm

/* Input variables: t_0, triangle to be refined, and τ, triangular mesh
Output variables: new mesh τ
Internal variables: P, Q, nodes; t_1, t_2, t^*, triangles */
Perform the longest-edge bisection of t_0
/* 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 */
If $P \neq Q$ **then** join points P and Q
$P = Q$
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
$P = Q$
End While
End For
End.

The performance of the 4-T algorithm of Rivara is indicated for a simple example in Fig. 4, in which (a) is the original triangulation, (b) shows the longest edge bisection of t, and (c) and (d) show the extension for conformity. (Fig. 4(d) shows the final triangulation in which subtriangles t_1 and t_2 of t have also been subdivided.) For comparison, (b1) and (c1) show the edge bisections in our new algorithm. Clearly, it reproduces the triangulation in (d). Further details follow in Section 2.2 next.

2.2. The two-dimensional skeleton based refinement (2D-SBR) algorithm

Recalling Figs. 4(b1), (c1) and (d) the algorithm proceeds by first bisecting edges and then connecting to triangulate. The 2D-SBR algorithm proceeds as follows:

The 2D-SBR Algorithm

/* Input variables: t_0, triangle to be refined, and τ, triangular mesh
Output variables: new mesh τ
Internal variables: P, Q, nodes; t, t^*, triangles */
/* 1. *Edge subdivision and assuring conformity* */
Perform the bisection of three edges of t_0
For each new midpoint P generated **do**
While P is not the longest-edge midpoint of the neighbor triangle t^* **do**
Perform the bisection of the longest-edge of t^*
/* let Q be the point generated */
$P = Q$
End While
End For

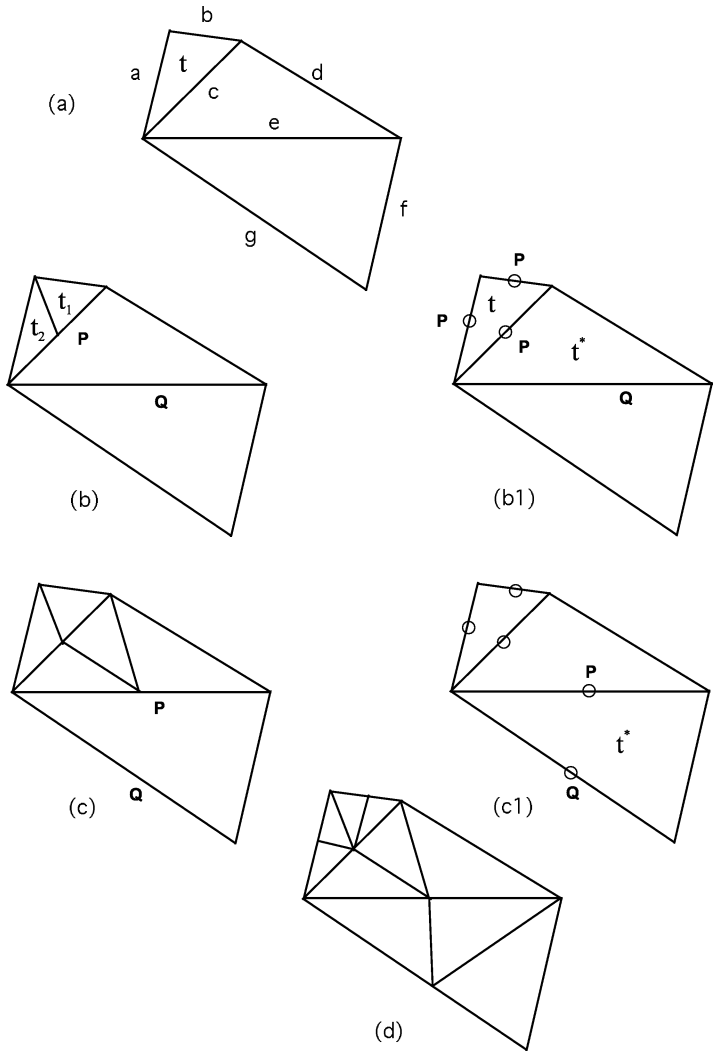


Fig. 4. Refinement and conformity.

/* 2. *Local subdivision of each involved triangle* */
For each involved $t \in \tau$ **do**
    Perform the suitable subdivision of t
End For
End.

Our 2D algorithm was first introduced by Plaza et al. [35], modifying the version proposed in [15]. It differs from Rivara's approach in the following points: (i) when the triangles are taken for evaluation of the refinement condition, the edges for bisection are subdivided, and (ii) after the conformity has been ensured, each triangle is divided following a suitable pattern. As shown previously there are only four possible configurations.

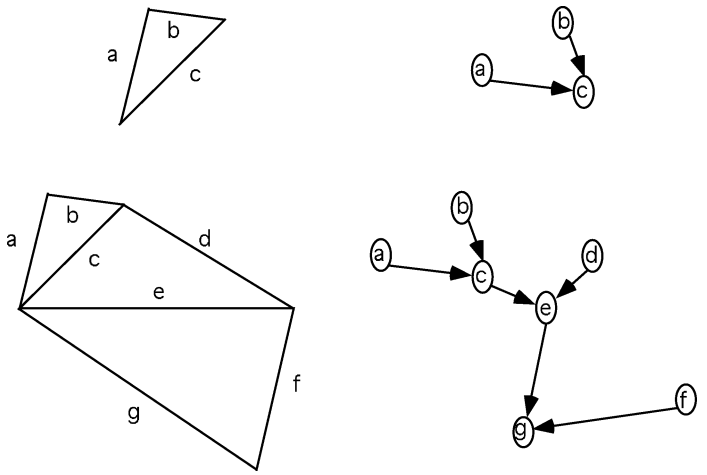


Fig. 5. Oriented graph representing the classification of the edges of a triangle, 2D mesh and associated graph.

Theorem 2.1. *The 2D-SBR algorithm is equivalent to the 4-T Rivara algorithm.*

Proof. Note that step 1 of the 2D-SBR algorithm, edge subdivision and assuring conformity, makes implicit use of the longest-edge propagation path concept, introduced and discussed in [33,40]. This concept is also equivalent to the associated graph to the triangulation (see below), introduced in [34]. The first step also assures the conformity of the refined mesh in the same way as the 4-T algorithm.

It is clear, finally, that the possible patterns from dividing a particular triangle are the same in both algorithms, and these patterns depend on the number of bisected edges. □

Because the longest edge of a triangle plays an important role in these algorithms, for convenience, the edges of each triangle can be classified 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 Fig. 5, can be associated with each triangle. The arrows indicate the dependency of the edges when refining to enforce conformity. This classification of the edges is local to each triangle because an edge can be the longest one in one triangle but not the longest one in the neighbor triangle.

3. The 3D-SBP algorithm

3.1. Definitions

We introduce here some definitions and notations for the 3D case.

Definition 3.1 (Surrounding edge). Let $T^n = \{\tau_1 < \tau_2 < \cdots < \tau_n\}$ be a sequence of triangular grids obtained by edge bisection in $\mathbb{R}^n$, and let P be a node not belonging to the coarsest mesh τ_1. Then the edge e in which node P is at the midpoint is called the surrounding edge of P, and we write $\mathrm{srr}(P) = e$. It is clear that if node P is a new-node in level k, its surrounding edge has been created in level j for some $j < k$.

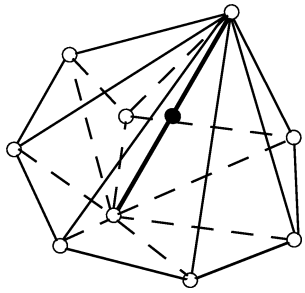


Fig. 6. Hull relative to the tessellation.

Definition 3.2 (Supporting face). To each edge e we assign one of the faces f having e as one of its edges. Face f will be called the supporting face of e, and we write $\mathrm{spp}(e) = f$.

Definition 3.3 (Hull of an edge). Let τ be an n-dimensional simplicial mesh. The hull of edge e is the set of simplices $h(e) = \{S \in \tau\colon\ e \in S\}$. That is, $h(e)$ is comprised of all the simplices that share the same edge e. Note that in general $h(e)$ is not a convex set. It is worth noting that if the information of the surrounding edge of each new node and the supporting face for each edge is kept, it is relatively easy to get the hull of an edge.

3.2. Outline of the refinement algorithm

Let τ be a three-dimensional tetrahedral grid and t_0 be a tetrahedron in the grid to be refined.

The 3D-SBR algorithm

/* Input variables: t_0, tetrahedron to be refined, and τ, 3D triangular mesh
Output variables: new mesh τ
Internal variables: L, set of nodes; N_e, P, $P_{e^\star}$, nodes; e, $e^\star$, edges; f, face; t, tetrahedron */
/* 1. *Edge subdivision* */
$L = \{\emptyset\}$
For each edge $e \in t_0$ **do**
Bisect e producing *new-node* N_e
$L = L \cup N_e$
End For
/* 2. *The conformity is ensured* */
While $L \neq \emptyset$ **do**
/* Let P be a node from L with surrounding edge e_P */
For each tetrahedron $t \in h(e_P)$ **do**
For each non-conforming face f of t **do**
Bisect the longest-edge $e^\star$ of f producing *new-node* $P_{e^\star}$ at the midpoint of $e^\star$
$L = L \cup P_{e^\star}$
End For

End For
$L = L - P$
End While
/* 3. *The subdivision of the skeleton is performed* */
For each triangular face $f \in \mathrm{skt}(\tau)$ to be subdivided **do**
Subdivide f
End For
/* 4. *The subdivision of the tetrahedra is performed* */
For each tetrahedron $t \in \tau$ to be subdivided **do**
Subdivide t
End For
End.

The iterative application of the previous algorithm yields a sequence of nested tetrahedral meshes $T^n = \{\tau_1 < \tau_2 < \cdots < \tau_n\}$, where τ_1 represents the initial mesh and τ_n the finest mesh in the sequence.

3.3. Properties

The salient feature in the 3D-SBR 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 the 2-dimensional algorithm for simplicial subdivision applied to the skeleton of the 3D triangulation.

As an example, Fig. 7 shows the application of the algorithm to a very simple initial mesh comprised of only 2 tetrahedra (Fig. 7(a)). We assume that tetrahedron $\langle 1, 2, 3, 4\rangle$ is to be refined based on some error indicator (Fig. 7(b)) and the adjacent one is to be refined to satisfy conformity (see Fig. 7(c), in which nodes N1 and N2 are introduced). Fig. 7(d) presents the division of the 2D skeleton and Fig. 7(e) the final mesh in which the new elements have been defined.

In the final stage, step 4 of the algorithm, only the local subdivision of each involved 3-simplex of the original mesh must be made.

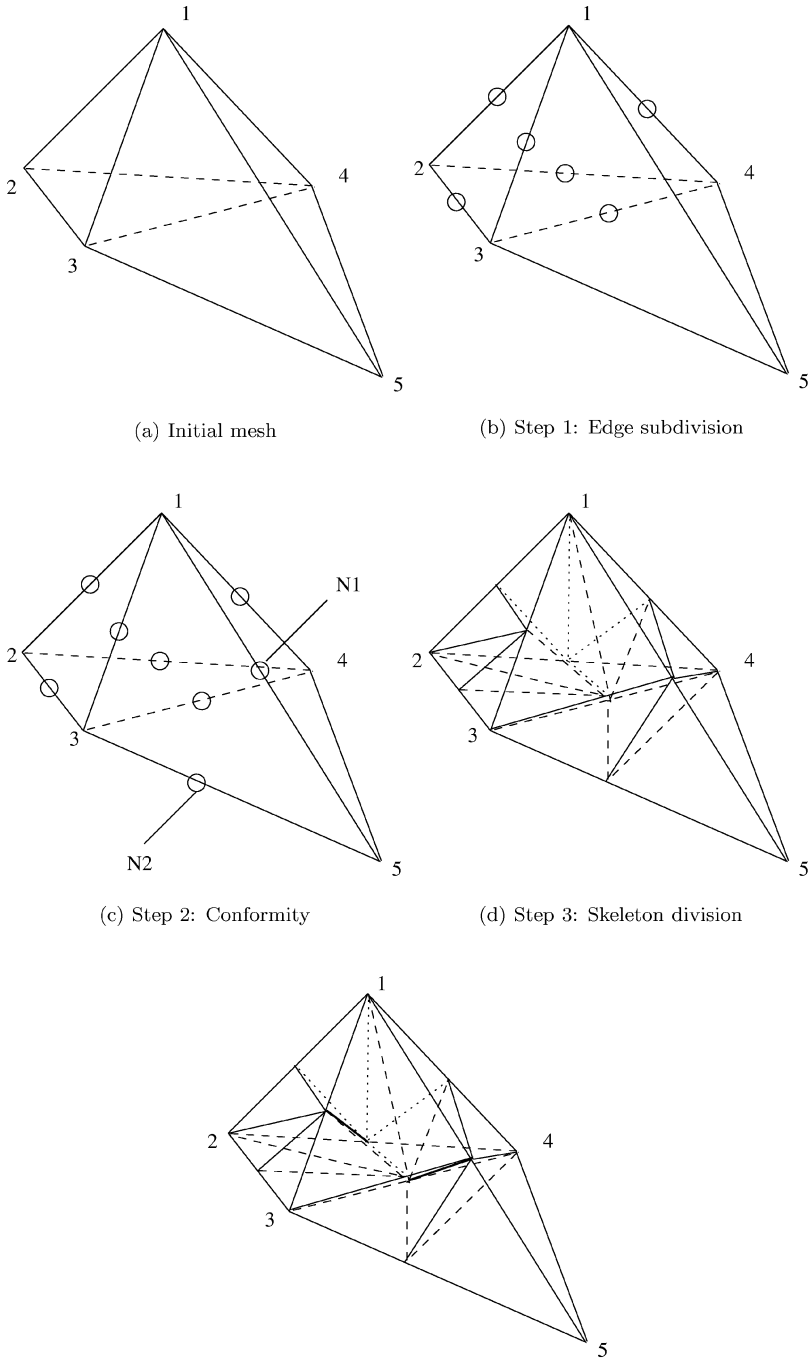
Theorem 3.1. *The 3D-SBR algorithm is finite.*

Proof. This result follows directly from the fact that the corresponding two dimensional 4-T algorithm [38] is finite, and the number of possible configurations for subdividing a 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 [41]. □

Theorem 3.2. *The subdivision of the faces is compatible 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 2D algorithm to refine the skeleton. By the earlier lemmas the refinement at the boundary can also be achieved by subdivision of the tetrahedron following the order of its bisection edges. □



(a) Initial mesh

(b) Step 1: Edge subdivision

(c) Step 2: Conformity

(d) Step 3: Skeleton division

(e) Final mesh

Fig. 7. Example of application of the 3D-SBR algorithm. The initial mesh is given in (a) and the edges of one tetrahedron are bisected in (b). New edges in the neighbor tetrahedron are bisected in (c) and the surface triangles (skeleton) are refined in (d). The interior of each tetrahedron is defined in (e) to yield the final mesh.

Corollary 3.1. *The meshes obtained by application of the previous algorithm are conforming.*

4. Complexity of the algorithm

Consider the 3D case. First step, *edge subdivison*, has a complexity of O(1).

Assuring the conformity, step 2, has a complexity of $O(N_a)$ since it is accomplished locally, for each node in the set L as the procedure below shows. N_a is the number of added nodes at refinement. Note that this task can be performed locally by means of vectors *surrounding edge* srr*(N)*, and *supporting face* spp*(e)*, defined in Section 3.

Procedure Find-hull

/* Input variables: N, node
Output variables: $h(e)$, list of tetrahedra
Internal variables: e, edge; f, f^*, faces; t_1, t_2, t^*, tetrahedra */
$e = \mathrm{srr}(N)$
$h(e) = \{\emptyset\}$
$f = \mathrm{spp}(e)$
/* Let t_1 and t_2 be the two tetrahedra sharing face f */
While $t_1 \neq t_2$ **do**
$h(e) = h(e) \cup t_1$
/* Let f^* be the face in t_1 sharing edge e, with $f^* \neq f$
and t^* the neighbor tetrahedron of t_1 through f^* */
$t_1 = t^*$
$f = f^*$
End While
$h(e) = h(e) \cup t_1$
End

In the previous procedure, for simplicity, it has been supposed that the edge e is an internal edge to the domain Ω. The cases in which edge e is *external*, that is e belongs to the boundary of Ω, or the supporting face of e, $f = \mathrm{spp}(e)$, is in the boundary of Ω, can be studied in an analogous way.

Following the 3D-SBR algorithm, steps 3 and 4 present a complexity of $O(N_f)$ and $O(N_t)$, respectively, where N_f is the number of involved faces, and N_t the number of involved tetrahedra. Hence, the complexity of the algorithm can be estimated by $O(N_a) + O(N_f) + O(N_t)$.

In the case of an initial triangulation in which the number of faces and the number of tetrahedra are of the same order as the number of nodes, linear complexity in the number of nodes follows. Note, that this is the general case for the meshes used in finite element applications (see, for example, [8, p. 75] and [13]). The relation between the numbers of topological entities in real meshes is studied in detail in [6]. When the global refinement proceeds n times, both the number of vertices and the number of tetrahedra, tend to be of the same order. This same order of magnitude is implied by the result: $\lim_{n\to\infty}(4T_n/N_n) = 24$ (see [42]), where T_n is the number of tetrahedra after n global refinements, and N_n is the number of nodes after n global refinements. Numerical experiments also indicate that in practice the algorithm performs like a linear complexity algorithm [32].

5. Numerical examples

Here we present some refinement 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. Several measures of mesh quality (or degeneracy) are possible and we summarize some of these bellow:

Whitehead [44] introduced the *relative thickness* of a simplex S:

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

where the *'radius'* $\widetilde{r}(S)$ is defined here as the distance from the centroid of S to its boundary (that is the minimum distance from the centroid to the faces of the simplex) and $d(S)$ is the diameter of S (that is, the length of its longest edge).

Stynes [43] introduced

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

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

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

where n is the dimension of the simplex S.

Other authors (see, for example, [12]) have used

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

where $r(S)$ is now the length of the radius of the inscribed sphere and $R(S)$ is the radius of the circumsphere.

For tetrahedron t and each vertex $P \in t$, Rivara and Levin [41] use the measure Φ_P associated with the solid angle at P:

$$\Phi_P = \sin^{-1}\left(1 - \cos^2\alpha_P - \cos^2\beta_P - \cos^2\gamma_P + 2\,\cos\alpha_P\,\cos\beta_P\,\cos\gamma_P\right)^{1/2}, \tag{5}$$

where α_P, β_P, γ_P are the associated corner angles, to define $\Phi_t = \min\{\Phi_P\colon\ P \in t\}$. Moreover, for each conforming mesh, τ, they assign the number $\Phi_\tau = \min\{\Phi_t\colon\ \ t \in \tau\}$ 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. The relation between Φ_P and the solid angle Ω_P at P, can be established as follows. From [24] we get

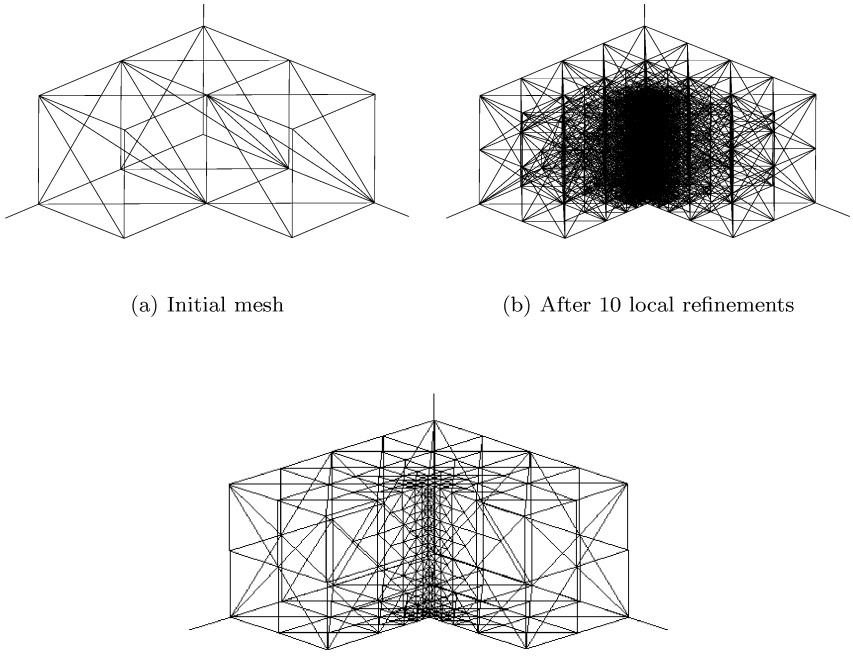
$$\sin(\Omega_P/2) = \frac{(1 - \cos^2\alpha_P - \cos^2\beta_P - \cos^2\gamma_P + 2\,\cos\alpha_P\,\cos\beta_P\,\cos\gamma_P)^{1/2}}{4\cos(\alpha_P/2)\cos(\beta_P/2)\cos(\gamma_P/2)}. \tag{6}$$

So,

$$\Omega_P = 2\,\sin^{-1}\frac{\sin(\Phi_P)}{4\cos(\alpha_P/2)\cos(\beta_P/2)\cos(\gamma_P/2)}. \tag{7}$$

Table 2
Problems 1–4

P1			P2			P3			P4		
$\eta = 0.8846$			$\eta = 0.8399$			$\eta = 0.2835$			$\eta = 1.0000$		
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	1.0	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	$2\sqrt{2}$



(a) Initial mesh

(b) After 10 local refinements

(c) Detail of the refinement on the surface

Fig. 8. Local refinement at a re-entrant corner.

Liu and Joe [25,26] introduce the estimate

$$\eta(t) = \frac{3\sqrt[3]{\lambda_1\lambda_2\lambda_3}}{\lambda_1 + \lambda_2 + \lambda_3}, \tag{8}$$

where λ_i are the eigenvalues of the matrix $A(R, M) = (MR^{-1})^{\mathrm{T}} MR^{-1}$ 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

Table 3
Comparison of problems 1–4

Problem 1									
	Longest edge bisection			QLRB			3D-SBR		
Level	NTET	$\eta_{\min}$	$\Omega_{\min}$	NTET	$\eta_{\min}$	$\Omega_{\min}$	NTET	$\eta_{\min}$	$\Omega_{\min}$
1	1	0.885	0.356	1	0.885	0.356	1	0.885	0.356
2	8	0.682	0.189	8	0.682	0.187	8	0.682	0.187
3	86	0.479	0.079	64	0.663	0.164	64	0.571	0.142
Problem 2									
	Longest edge bisection			QLRB			3D-SBR		
Level	NTET	$\eta_{\min}$	$\Omega_{\min}$	NTET	$\eta_{\min}$	$\Omega_{\min}$	NTET	$\eta_{\min}$	$\Omega_{\min}$
1	1	0.840	0.339	1	0.840	0.339	1	0.840	0.339
2	8	0.657	0.169	8	0.504	0.091	8	0.504	0.091
3	86	0.458	0.070	64	0.504	0.091	64	0.504	0.091
Problem 3									
	Longest edge bisection			QLRB			3D-SBR		
Level	NTET	$\eta_{\min}$	$\Omega_{\min}$	NTET	$\eta_{\min}$	$\Omega_{\min}$	NTET	$\eta_{\min}$	$\Omega_{\min}$
1	1	0.284	0.047	1	0.284	0.047	1	0.284	0.047
2	8	0.181	0.024	8	0.181	0.024	8	0.181	0.024
3	192	0.165	0.014	64	0.170	0.022	64	0.163	0.014
Problem 4									
	Longest edge bisection			QLRB			3D-SBR		
Level	NTET	$\eta_{\min}$	$\Omega_{\min}$	NTET	$\eta_{\min}$	$\Omega_{\min}$	NTET	$\eta_{\min}$	$\Omega_{\min}$
1	1	1.00	0.551	1	1.00	0.551	1	1.00	0.551
2	8	0.546	0.132	8	0.546	0.132	8	0.546	0.132
3	306	0.458	0.070	64	0.429	0.070	64	0.429	0.070

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

$$\eta(t) = \frac{12\ (3\ \text{volume})^{2/3}}{\sum_{i=1}^{6} l_i^2}\ , \tag{9}$$

where l_i means the length of the edge i.

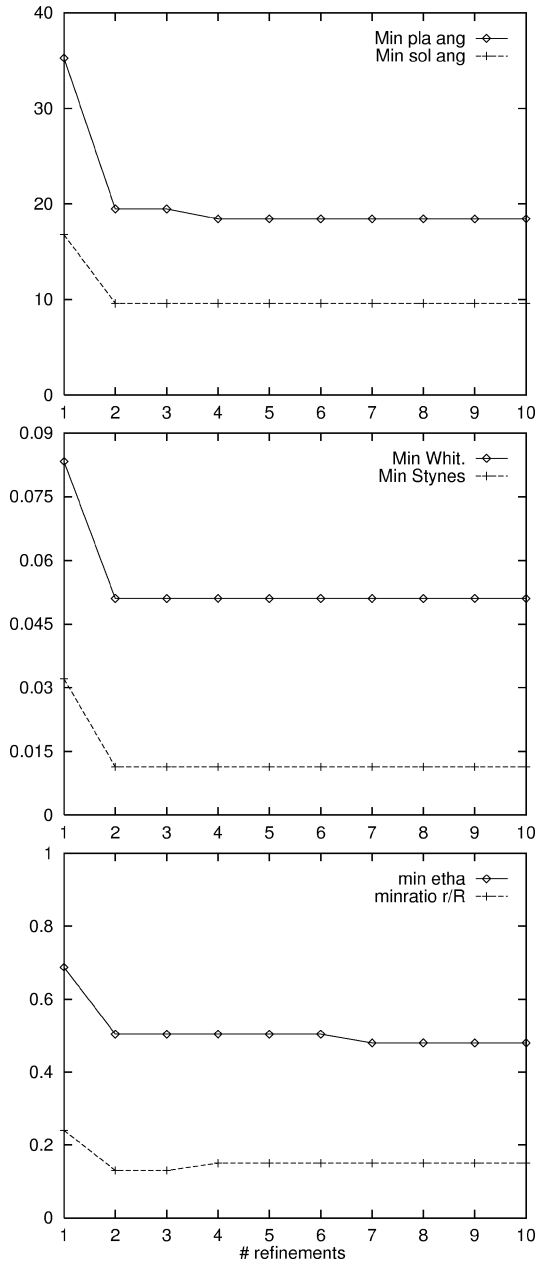


Fig. 9. Evolution of shape measures for example of Fig. 8.

We compute these parameters in our test 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.

In Table 2 the first four problems in [41] are listed in terms of the coordinates of their four vertices. They are also considered in [26]. P1 and P2 are well-shaped tetrahedra; P3 is a poorly-shaped tetrahedron and P4 is the regular tetrahedron.

The left part of Table 3 shows the results reproduced from (8) and (7) based on the longest edge bisection [41]. The middle part of the Table 3 is for QLRB in [26]. Note that the results using our algorithm in the right column are comparable to those by the longest edge bisection of Rivara and Levin [41] and the QLRB algorithm of Liu and Joe [26].

Another test case is presented in Fig. 8. The figure shows local refinements of a 3D domain in which a singularity on the interior corner edge is assumed. We suppose here an error function given by the equation

$$\text{Error} = \frac{1}{0.0001 + d^2}, \tag{10}$$

where d is the distance to the reentrant corner. Then we get an error per element as usual, by integrating the function Error in each tetrahedron t. Here we suppose a linear approximation of the function Error at the vertices in each tetrahedron, so that the error per element, $E(t)$, is equal to the volume of t, $\text{vol}(t)$, multiplied by the value of the function at the centroid of the element B_t:

$$E(t) = \text{vol}(t) \cdot \text{Error}(B_t), \tag{11}$$

where B_t is the baricenter of t. Since $E(t)$ is weighted by the volume of the element, Eq. (11) includes the scale of the mesh in the vecinity of the singularity. The mesh of Fig. 8(b) contains 1365 nodes and 6657 tetrahedra.

The evolution of the shape measurements through the sequence of 10 refinements for this test case can be observed in Fig. 9. 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). Better results in shape measure could be obtained by combination with some appropriate mesh improvement techniques like swapping or smoothing [17] or local transformations [21].

6. Conclusions

The ideas presented here provide a framework for studying local refinement in 2D and 3D and extend the ideas developed by Rivara and other researchers. In fact, the 3-dimensional refinement is deduced from the application of a similar 2-dimensional scheme to the skeleton. The algorithm complexity is linear by enforcing the conformity locally as the new nodes are introduced. Since the data structure is similar to that used in [16], implementation of the corresponding derefinement algorithm and of the multigrid method are relatively simple. Moreover, these algorithms can be applied to any initial triangular or tetrahedral mesh without any preprocessing.

The results confirm that the measure of degeneracy is bounded, and converges asymptotically to a fixed value. Even though a mathematical proof of this property is still not available, the algorithm developed in this paper promises to be a useful and practical tool for the refinement of tetrahedral grids. Moreover, as in the 2D case, it is also possible to develop *corresponding algorithms* for the derefinement of 3D meshes created by means of the refinement algorithm. This is an important feature for dealing with time dependent problems.

The basic idea for extending a two-dimensional algorithm to a three-dimensional one using the skeleton may be generalized and applied in dimension n and with other refinement algorithms. However,

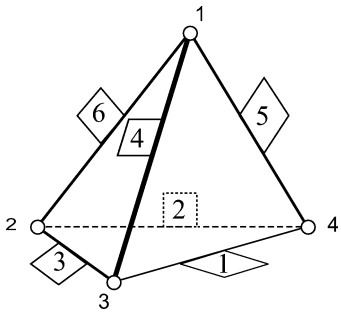


Fig. 10. Tetrahedron in canonical position and local numeration of the vertices and edges.

it should be noted that, in a general n-dimensional simplex we need n indices to classify the edges by their length, and this classification may not be very easy to implement. Furthermore, as n grows the number of possible configurations arising from refinement grows as well.

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

Appendix. Tetrahedral refinement configurations

Let us consider the generic tetrahedron with nodes and edges numbered as in Fig. 10 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 [26], Bánsch [5] or Mukherjee [30], we assign to each edge of the 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 edge of two faces (faces 1 and 2 sharing the edge). The longest edge of each one of the other two faces (faces 3 and 4) is labeled edge type 2, and the rest of the edges are type 3. Edge type 1 is the reference edge of the tetrahedron.

Theorem A.1. *There are* 90 *possible configurations obtained by local or global refinement, including no-division.*

Proof. Note that each configuration different from simple bisection by the longest edge of the tetrahedron is determined by the number and relative position of the introduced midpoint nodes. Nodes other than the midpoint of the longest edge of the tetrahedron will appear on faces 3 and 4. In other words, to count the number of configurations we can ask for the number of possibilities of division for the pair of faces 3 and 4 and add 2 more: single bisection and no-bisection of the initial tetrahedron.

Recall that the first step is based on refinement of the triangular faces in the skeleton. As seen in Fig. 3 there are 4 possible divisions of a triangular face once the longest edge has been determined. Since the longest edge for face 3 and for face 4 is not predetermined and there are three edges we have $3 \times 4 = 12$ different possibilities, and keeping in mind the single bisection option, there are 13 possibilities for each triangular face. Since faces 2 and 3 share edge number 6, we have to distinguish if edge 6 is subdivided (8 cases) or not (5 cases). So the total number of configurations can be counted as

Edge # 6 divided	$8 \times 8 =$	64
Edge # 6 not divided	$5 \times 5 =$	25
No division		1
TOTAL #		90

□

However, many of the configurations above will be equivalent within a rotation. We now introduce a classification procedure to delineate the equivalent configurations. In fact, the edges on each tetrahedron can be consistently classified using the order of the edges of each triangular face. In this way, the refinement of the skeleton by means of our 2D algorithm generates the trace of the 3D refinement obtained by bisection, following the order of the edges chosen for subdivision.

We distinguish three types of tetrahedra based on the relative position of their edge-types. First, we consider the tetrahedra obtained by bisecting the type 1 edge, second by the edge–or edges–type 2, and so on. Note that edge bisection generates more than two more tetrahedra if and only if the edge is common to two previous tetrahedra. That is, if and only if the edge bisected is opposite the longest one. Hence the type of this edge is important in order to classify the tetrahedra. The classification scheme for tetrahedron t proceeds as follows:

Procedure Classification
/* Input variables: t tetrahedron
Output variables: type */
If edge 6 has type 3 **then**
t has type 1
Else 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 Fig. 11. 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 with a type-1 tetrahedron in Fig. 11(a) implies that the edge opposite to the longest one is type-3 -as marked in bold. Similarly, in Fig. 11(b) a type-2 tetrahedron, a single type-2 edge is opposite to the reference edge. 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. 11(c).

The simplest configuration obtained by *local* or *partial* 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* or *full* refinement of a tetrahedron means that a node is 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 Fig. 12(a) be rotated $\pi\ radians$ 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

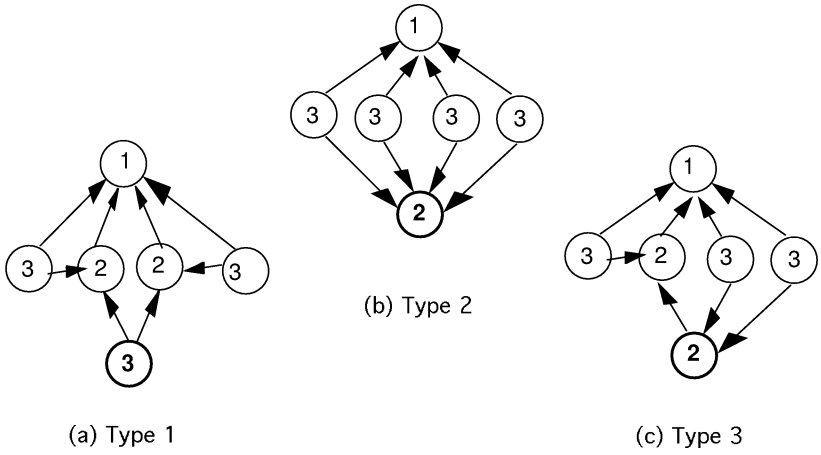


Fig. 11. Oriented graphs representing the different classes of tetrahedra.

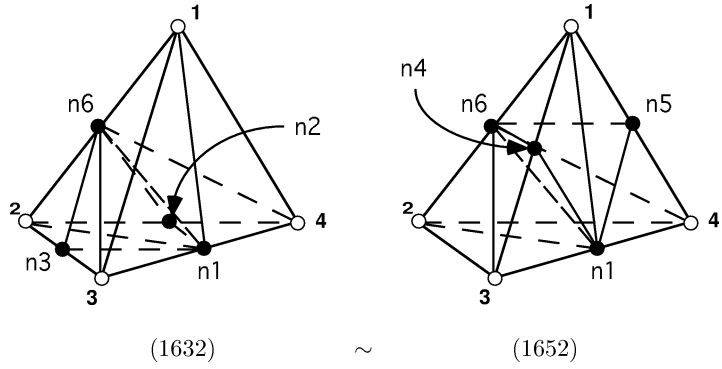


Fig. 12. Two equivalent configurations.

has changed. This is the unique rotation that preserves the canonical position and is still distinct from the identity [29]. 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 $(ei\colon\ i = 1, \ldots, 6)$ we can write: $r(e1) = e1$, $r(e2) = e4$, $r(e3) = e5$ and $r(e6) = e6$.

Next, let us represent each configuration by means of integer sequences of the form $(i \ldots l)$ where the edges that are subdivided are listed in order according to their types, from type 1 to type 3. For instance, the configuration of Fig. 12(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 opposite the reference edge (1), the tetrahedron has type 2 in which 4 nodes are added.) For comparison, the configuration (1645) is shown in Fig. 12(b).

In the set of integer sequences

$$\mathrm{IS} = \{\sigma, \text{such that } \sigma = (i_1, \ldots, i_k),\ \text{where}\ \ k \leqslant 6,\ i_j \in \{1, 2, \ldots, 6\},\ i_1 = 1 \text{ and } i_n \neq i_m \text{ if } n \neq m\}$$

we define an equivalence relation in this way:

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

Table 4
Possible configurations obtained by tetrahedron subdivision 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	**(1,2)**∼(1,4)	**(1,6)**	(1,2)∼(1,4)	
	(1,3)∼(1,5)		(1,3)∼(1,5)	3
3	**(1,2,5)**∼(1,4,3)	**(1,6,2)**∼(1,6,4)	(1,2,5)∼(1,4,3)	
	(1,3,4)∼(1,5,2)	**(1,6,3)**∼(1,6,5)	(134)∼(152)	
	(1,2,3)∼(1,4,5)		**(1,2,6)**∼(1,4,6)	
	(1,2,4)		**(1,3,6)**∼(1,5,6)	
	(1,3,5)			9
4	**(1,2,3,6)**∼(1,4,5,6)	**(1,6,5,4)**∼(1,6,3,2)	**(1,2,6,5)**∼(1,4,6,3)	
	(1,2,3,4)∼(1,4,5,2)	**(1,6,2,5)**∼(1,6,4,3)	**(1,3,6,4)**∼(1,5,6,2)	
	(1,2,3,5)∼(1,4,5,3)	**(1,6,2,4)**	**(1,2,6,3)**∼(1,4,6,5)	
	(1,3,5,6)	**(1,6,3,5)**	**(1,2,6,4)**∼(1,4,6,2)	
	(1,3,5,4)∼(1,5,3,2)		**(1,3,6,2)**∼(1,5,6,4)	
	(1,4,2,6)		**(1,3,6,5)**∼(1,5,6,3)	
	(1,4,2,3)∼(1,2,4,5)			17
5	**(1,2,3,6,4)**∼(1,4,5,6,2)	**(1,6,2,3,4)**∼(1,6,4,5,2)	**(1,2,6,5,3)**∼(1,4,6,3,5)	
	(1,2,3,6,5)∼(1,4,5,6,3)	**(1,6,2,3,5)**∼(1,6,4,5,3)	**(1,2,6,5,4)**∼(1,4,6,3,2)	
	(1,2,3,4,5)∼(1,4,5,2,3)		**(1,3,6,4,5)**∼(1,5,6,2,3)	
	(1,2,4,6,3)∼(1,4,2,6,5)		**(1,3,6,4,2)**∼(1,5,6,2,4)	
	(1,3,5,2,4)		**(1,2,6,3,4)**∼(1,4,6,5,2)	
	(1,3,5,6,2)∼(1,5,3,6,4)		**(1,3,6,2,5)**∼(1,5,6,4,3)	
	(1,4,2,3,5)			15
6	**(1,2,3,4,5,6)**∼(1,4,5,2,3,6)	**(1,6,2,3,4,5)**	**(1,2,6,3,4,5)**∼(1,4,6,5,2,3)	
	(1,2,4,3,5,6)		**(1,3,6,2,4,5)**∼(1,5,6,4,2,3)	
	(1,3,5,2,4,6)			6
#	25	10	16	**51**

(2) If $\sigma_1 = \sigma_2$ except for the order of edges with the same type non-connected in the associated graph, then $\sigma_1 \sim \sigma_2$. This concerns type-3 tetrahedra, in which there is a relation between the two type-2 edges.

It follows immediately that these conditions define an equivalence relation in IS.

Theorem A.2. *There are* 51 *different possible configurations obtained by local or global refinement, excluding rotations.*

Proof. The proof of the theorem follows simply by counting all the equivalence classes which are identified in Table 4. There, the cases in bold are the distinct ones and the other cases are equivalent by rotation to bold cases. The total number of cases in Table 4 is 89 of which 51 are distinct. □

References

[1] I. Babuška, W.C. Rheinboldt, Error estimates for adaptive finite element computations, SIAM J. Numer. Anal. 15 (1978) 736–754.
[2] I. Babuška, W.C. Rheinboldt, A posteriori error analysis of finite element solutions for one-dimensional problems, SIAM J. Numer. Anal. 18 (1981) 565–589.
[3] R.E. Bank, A.H. Sherman, The use of adaptive grid refinement for badly behaved elliptic partial differential equations, in: R.Vichnevetsky and R.S. Stepleman (Eds.), Advances in Computer Methods for Partial Differential Equations III, IMACS, 1979, 33–39.
[4] R.E. Bank, J. Xu, An algorithm for coarsening unstructured meshes, Numer. Math. 73 (1996) 1–36.
[5] E. Bänsch, Local mesh refinement in 2 and 3 dimensions, Impact Comput. Sci. Engrg. 3 (1991) 181–191.
[6] M.W. Beall, M.S. Shephard, A general topology-based data structure, Internat. J. Numer. Methods Engrg. 40 (1997) 1573–1596.
[7] M. Berger, Geometry, Springer, Berlin, 1987.
[8] M. Bern, D. Eppstein, Mesh generation and optimal triangulation, in: D.-Z. Du and F.K. Hwang (Eds), Computing in Euclidean Geometry, World Scientific, 1992, 23–90.
[9] J. Bey, Tetrahedral grid refinement, Computing 55 (1995) 355–378.
[10] G.F. Carey, A mesh refinement scheme for finite element computations, Comput. Methods Appl. Mech. Engrg. 7 (1) (1976) 93–105.
[11] G.F. Carey, Computational Grids: Generation, Refinement and Solution Strategies, Taylor and Francis, 1997.
[12] P. Conti, N. Hitschfeld, W. Fichtner, Ω–an octree-based mixed element grid allocator for the simulation of complex 3D device structures, IEEE Trans. Comput.-Aided Design 10 (1991) 1231–1241.
[13] R.A. Dwyer, Higher-dimensional Voronoi diagrams in linear expected time, in: Proc. 5th Annual ACM Symp. on Computational Geometry (1989) 326–333.
[14] H. Edelsbrunner, Algorithms in Combinatorial Geometry, Springer, Berlin, 1987.
[15] L. Ferragut, NEPTUNO, A system of adaptive finite element methods, (in FORTRAN), Dept. Mat. Aplic. y Met. Inf., ETSI Minas, Madrid, 1987.
[16] L. Ferragut, R. Montenegro, A. Plaza, Efficient refinement/derefinement algorithm of nested meshes to solve evolution problems, Comm. Numer. Methods Engrg. 10 (1994) 403–412.
[17] L.A. Freitag, C. Ollivier-Gooch, Tetrahedral mesh improvement using swapping and smoothing, Internat J. Numer. Methods. Engrg. 40 (1997) 3979–4002.
[18] J.P.S.R. Gago, D.W. Kelly, O.C. Zienkiewicz, A posteriori error analysis and adaptive processes in the finite element method. Part II: Adaptive mesh refinement, Internat. J. Numer. Methods Engrg. 19 (1983) 1621–1656.
[19] H. George, Automatic Mesh Generation, Wiley, New York, 1991.

[20] C.M. Hoffmann, How to construct the skeleton of CSG objects, in: 4th IMA Conference on Mathematics of Surfaces, Bath, England, 1990.
[21] B. Joe, Construction of three-dimensional improved quality triangulations using local transformations, SIAM J. Sci. Comput. 16 (1995) 1292–1307.
[22] B. Kearfott, A proof of convergence and an error bound for the method of bisection in $\mathbb{R}^n$, Math. Comp. 32 (1978) 1147–1153.
[23] I. Kossaczký, A recursive approach to local mesh refinement in two and three dimensions, J. Comput. Appl. Math. 55 (1994) 275–288.
[24] A. Liu, B. Joe, Relationship between tetrahedron shape measures, BIT 34 (1994) 268–287.
[25] A. Liu, B. Joe, On the shape of tetrahedra from bisection, Math. Comp. 63 (1994) 141–154.
[26] A. Liu, B. Joe, Quality local refinement of tetrahedral meshes based on bisection, SIAM J. Sci. Comput. 16 (1995) 1269–1291.
[27] J.M. Maubach, Local bisection refinement for n-simplicial grids generated by reflection, SIAM J. Sci. Statist. Comput. 16 (1995) 210–227.
[28] W.F. Mitchell, Optimal multilevel iterative methods for adaptive grids, SIAM J. Sci. Statist. Comput. 13 (1992) 146–167.
[29] J.M. Montesinos, Classical Tessellations and Three-Manifolds, Springer, Berlin, 1987.
[30] A. Mukherjee, An adaptive finite element code for elliptic boundary value problems in three dimensions with applications in numerical relativity, Ph.D. Thesis, Pennsylvania State University, University Park, PA, 1996.
[31] J.R. Munkres, Elementary Differential Topology, 2nd Edition, Princeton Univ. Press, 1966.
[32] A. Plaza, G.F. Carey, A new refinement algorithm for tetrahedral grids based on skeleton, TICAM Technical Report 96-54, November 1996.
[33] A. Plaza, G.F. Carey, New mathematical tools and techniques for the refinement and/or improvement of unstructured triangulations, in: Proceedings 5th International Meshing Roundtable '96, Pittsburgh, October 10–11, 1996, 77–86.
[34] A. Plaza, G.F. Carey, About local refinement of tetrahedral grids based on bisection, in: Proceedings 5th International Meshing Roundtable '96, Pittsburgh, October 10–11, 1996, 123–136.
[35] A. Plaza, R. Montenegro, L. Ferragut, An improved derefinement algorithm of nested meshes, in: M. Papadrakakis (Ed.), Advances in Post and Preprocessing for Finite Element Technology, Civil-Comp Ltd., 1994, 175–180.
[36] M. Price, C. Stops, G. Butlin, A medial object toolkit for meshing and other applications, in: 4th International Meshing Roundtable, Albuquerque, NM, 1995.
[37] M.C. Rivara, Mesh refinement based on the generalized bisection of simplices, SIAM J. Numer. Anal. 2 (1984) 604–613.
[38] M.C. Rivara, A grid generator based on 4-triangles conforming mesh refinement algorithms, Internat. J. Numer. Methods Engrg. 24 (1987) 1343–1354.
[39] M.C. Rivara, Selective refinement/derefinement algorithms for sequences of nested triangulations, Internat J. Numer. Methods Engrg. 28 (1989) 2889–2906.
[40] M.C. Rivara, New longest-edge algorithms for the refinement and/or improvement of unstructured triangulations, Internat J. Numer. Methods Engrg. 40 (1997) 3313–3324.
[41] M.C. Rivara, C. Levin, A 3-d refinement algorithm suitable for adaptive and multi-grid techniques, J. Comput. Appl. Math. 8 (1992) 281–290.
[42] M.C. Rivara, A. Plaza, Mesh refinement/derefinement based on the 8-tetrahedra longest-edge partition, Math. Comp., submitted.
[43] M. Stynes, On faster convergence of the bisection method for all triangles, Math. Comp. 35 (1980) 1995–1201.
[44] J.H.C. Whitehead, On C^1-complexes, Ann. Math. 41 (1940) 809–824.