SIAM J. SCI. COMPUT.
Vol. 22, No. 2, pp. 431–448

LOCALLY ADAPTED TETRAHEDRAL MESHES USING BISECTION*

DOUGLAS N. ARNOLD†, ARUP MUKHERJEE‡, AND LUC POULY§

Abstract. We present an algorithm for the construction of locally adapted conformal tetrahedral meshes. The algorithm is based on bisection of tetrahedra. A new data structure is introduced, which simplifies both the selection of the refinement edge of a tetrahedron and the recursive refinement to conformity of a mesh once some tetrahedra have been bisected. We prove that repeated application of the algorithm leads to only finitely many tetrahedral shapes up to similarity, and we bound the amount of additional refinement that is needed to achieve conformity. Numerical examples of the effectiveness of the algorithm are presented.

Key words. bisection, tetrahedral meshes, adaptive refinement, similarity classes, finite elements

AMS subject classification. 65N50

PII. S1064827597323373

1. Introduction. The generation of locally adapted conforming tetrahedral meshes is an important component of many modern algorithms, for example, in the finite element solution of partial differential equations. Typically, such meshes are produced by starting with a coarse tetrahedral mesh, selecting certain elements for refinement, somehow refining those elements and others as necessary to maintain conformity, and then possibly repeating this process one (or more than one) time. In this paper we present a simple algorithm for this purpose, and we analyze its behavior. In particular, we consider the question of the shape of tetrahedra that may arise from repeated application of our algorithm and show in section 4 that only a fixed finite number of dissimilar tetrahedra ever arise. A fortiori, the tetrahedral shape cannot degenerate as the mesh is refined in the sense that all the solid angles of all the descendants remain bounded below by a positive constant depending on the tetrahedra in the initial mesh.

The basic refinement step in our algorithm is tetrahedral bisection as in Figure 1. When bisecting a tetrahedron, a particular edge—called the *refinement edge*—is selected for the new vertex. As new tetrahedra are constructed in the course of generating an adapted mesh, their refinement edges must be selected carefully; otherwise element shapes may degenerate. A key aspect of any bisection algorithm is the selection of the refinement edge.

To refine a given conforming mesh we first bisect those tetrahedra that have been selected for refinement. This will usually lead to a nonconforming mesh (a mesh in which neighboring elements do not meet face-to-face). We then apply a recursive procedure to further refine until a conforming mesh is produced. Since it is not obvious

*Received by the editors June 23, 1997; accepted for publication November 24, 1998; published electronically July 13, 2000.
http://www.siam.org/journals/sisc/22-2/32337.html
†Department of Mathematics, Penn State University, University Park, PA 16802 (dna@math.psu.edu). The work of this author was supported by NSF grants DMS-9500672 and DMS-9870399.
‡Department of Mathematics-Hill Center, Rutgers—The State University of New Jersey, Piscataway, NJ 08854-8019 (arup@math.rutgers.edu).
§Department of Mathematics, Swiss Federal Institute of Technology, CH-1015 Lausanne, Switzerland (Luc.Pouly@elca.ch). Current address: ELCA Informatique SA, CH-1000 Lausanne 13, Switzerland. The work of this author was supported by the Swiss National Research Foundation.

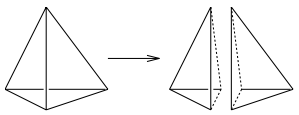

FIG. 1. *Bisection of a single tetrahedron.*

that this procedure will terminate in finitely many steps, in section 3 we provide a rigorous proof that this is the case and establish bounds on the number of steps.

Besides bisection, tetrahedra may be subdivided by octasection. This approach has been studied by Zhang [13] and Ong [10] to obtain uniformly refined meshes. However, octasection cannot be used alone to produce locally adapted conforming meshes. Motivated by work in two dimensions, where local quadrisection has been successfully combined with bisection to obtain conformity (cf., for example, Bank's code PLTMG [2]), Bey [4] has studied the use of local octasection combined with a variety of supplemental refinement strategies which are used to obtain conformity. By contrast, bisection can be used alone to create locally refined conforming meshes, which adds to the simplicity of its implementation. A further advantage of bisection over octasection is that it potentially allows finer control over mesh size. If an error indicator flags an element for refinement, with bisection the element can be divided to reduce its volume by a factor of 2, 4, ... , as required, while with octasection it is not possible to reduce element volume by a factor less than 8.

A number of other authors have proposed bisection-based algorithms for the refinement of tetrahedral meshes. In the pioneering paper [3], Bänsch presented an algorithm for local tetrahedral mesh refinement and discussed element shape degeneration. Although our algorithm appears quite different from Bänsch's, it is essentially equivalent, as we shall discuss in section 2. Another paper which influenced our work is that of Maubach [7]. Maubach considered the question of assigning refinement edges to successive bisections of a single simplex in an arbitrary number of dimensions. His algorithm cannot be easily applied to generate conforming adapted meshes, except for quite special initial meshes. For successive refinement of a single tetrahedron, we establish the precise relationship between our method and his in section 4. We show that his method generates only finitely many similarity classes of simplices (in n dimensions), and from this we deduce the same result for our algorithm. Liu and Joe [6] also study local refinement by bisection. Their algorithm, which is relatively complicated to state and to analyze, is in fact closely related to Bänsch's and therefore to ours. The relationship is discussed in section 2. Liu and Joe [5] prove that their algorithm generates only finitely many similarity classes, although their bound exceeds our sharp one by a large factor. A quite different approach to tetrahedral bisection has been pursued by Rivara [11] and Rivara and Levin [12]. They always use the longest edge of a tetrahedron as the refinement edge. In two dimensions, this approach leads to a finite number of similarity classes, although the number may be arbitrarily large, depending on the initial triangle [1]. As far as we know, the question of finiteness of similarity classes or even of element degeneration for longest edge bisection remains open in three dimensions.

A new aspect of our work is a data structure, which we name *marked tetrahedron*, used to store a geometric tetrahedron together with information necessary to choose its refinement edge and that of its descendants. This data structure is small—it contains just a little additional information beyond the vertices of the tetrahedron—

and it allows us to describe the bisection algorithm simply. Moreover, the marked tetrahedron data structure is useful for insuring mesh conformity. Any conforming tetrahedral mesh can be marked to yield a conforming mesh of marked tetrahedra, and therefore our algorithm does not require any restriction on the initial mesh.

2. Bisection of a single tetrahedron. In this section we describe the marked tetrahedron data structure and present the algorithm `BisectTet`. `BisectTet` bisects a marked tetrahedron by introducing a new vertex at the midpoint of the refinement edge and joining it to the two vertices of the original tetrahedron that do not lie on the refinement edge. It also marks the children (for use in further refinement).

To define a *marked tetrahedron* we introduce some terminology. For a tetrahedron τ, let $\mathcal{V}(\tau)$, $\mathcal{E}(\tau)$, and $\mathcal{F}(\tau)$ denote the set of its vertices, edges, and faces, respectively. For $\varphi \in \mathcal{F}(\tau)$, $\mathcal{E}(\varphi)$ denotes the edges contained in φ. Once a particular edge has been specified as the refinement edge of τ, the two faces that intersect at the refinement edge are called its *refinement faces*. For a marked tetrahedron we specify not only the refinement edge, but also a particular edge of each of the two nonrefinement faces. These are called the *marked edges* of these faces, and we take the refinement edge itself as the marked edges of the two refinement faces. Each marked tetrahedron is also assigned a boolean *flag*. The flag is always unset unless the marked edges of the four faces are all coplanar (we call this a *planar* marked tetrahedron), in which case the flag may or may not be set.

More precisely, a marked tetrahedron τ is a 4-tuple

$$\big(\mathcal{V}(\tau), r_\tau, (m_\varphi)_{\varphi\in\mathcal{F}(\tau)}, f_\tau\big),$$

where

$\mathcal{V}(\tau)$ is a set of four noncoplanar points in $\mathbb{R}^3$, the vertices of τ;
$r_\tau \in \mathcal{E}(\tau)$ is the refinement edge of τ;
$m_\varphi \in \mathcal{E}(\varphi)$ is the marked edge of φ, with $m_\varphi = r_\tau$ if $r_\tau \subset \varphi$;
$f_\tau \in \{0, 1\}$ is the flag, with $f_\tau = 0$ if τ is not planar.

Each marked nonrefinement edge of a marked tetrahedron is either adjacent or opposite to the refinement edge. Thus, we can classify marked tetrahedra into types as follows (cf. Figure 2).

- Type P, planar: the marked edges are coplanar. A type P tetrahedron is further classified as type P_f or type P_u, according to whether its flag is set or not, respectively.
- Type A, adjacent: the marked edges intersect the refinement edge, but are not coplanar.
- Type O, opposite: the marked edges of the nonrefinement faces do not intersect the refinement edge. In this case, a pair of opposite edges are marked in the tetrahedron: one as the refinement edge, and the other as the marked edge of the two nonrefinement faces intersecting there.
- Type M, mixed: the marked edge of just one of the nonrefinement faces intersects the refinement edge.

When a tetrahedron τ is bisected to create children τ_1 and τ_2, a face $\varphi \in \mathcal{F}(\tau_i)$ is called an *inherited face* if $\varphi \in \mathcal{F}(\tau)$, a *cut face* if $\varphi \subsetneqq \varphi'$ for some $\varphi' \in \mathcal{F}(\tau)$, and a *new face* otherwise. Each child has one inherited face, two cut faces, and one new face, which is common to both children. Cf. Figure 3. We are now ready to state `BisectTet`.

ALGORITHM $\{\tau_1, \tau_2\} =$ `BisectTet`(τ)
input: A marked tetrahedron τ

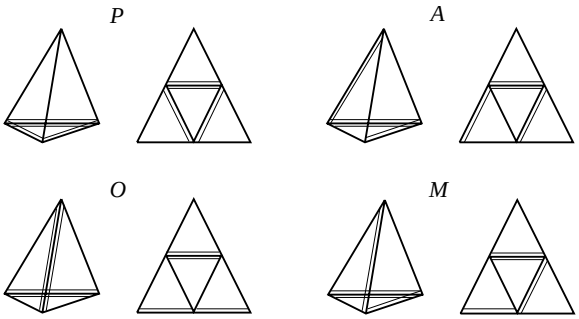

FIG. 2. *The four types of marked tetrahedra: P, A, O, and M. Each marked edge is indicated by a double line and the refinement edge is marked for both faces containing it. Each tetrahedron is shown in three dimensions and cut open and unfolded into two dimensions.*

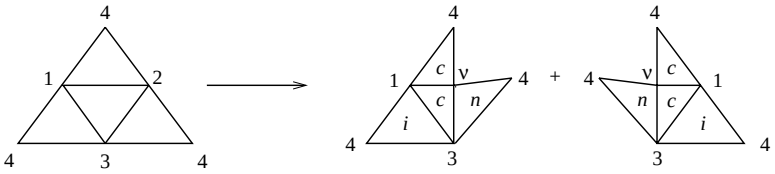

FIG. 3. *Typical bisection of a tetrahedron with the new vertex marked ν, cut faces marked c, inherited faces marked i, and new face marked n.*

output: Marked tetrahedra τ_1 and τ_2

1. Bisect τ by joining the midpoint of its refinement edge to each of the two vertices not lying on the refinement edge. This defines $\mathcal{V}(\tau_i)$ for $i = 1$ and 2.

Mark the faces of the children as follows.

2. The inherited face inherits its marked edge from the parent, and this marked edge is the refinement edge of the child.

3. On the cut faces of the children mark the edge opposite the new vertex with respect to the face.

4. The new face is marked the same way for both children. If the parent is type P_f, the marked edge is the edge connecting the new vertex to the new refinement edge. Otherwise it is the edge opposite the new vertex.

5. The flag is set in the children if and only if the parent is type P_u.

The algorithm is illustrated in Figure 4. Note that the tetrahedra τ_1 and τ_2 output by `BisectTet` will be of the same type. Figure 5 summarizes the relation between the input tetrahedron type and the output tetrahedra type. Note that types M and O are never output.

We close this section by relating the algorithm just described to those of Bänsch [3] and Liu and Joe [5], [6]. The relationship to the work of Maubach [7] will be treated in section 4. Bänsch used a slightly different system for marking a tetrahedron. Namely he associated to each face of the tetrahedron a unique marked edge. He assumes that at least one edge is marked in both faces containing it, and he calls such an edge a *global refinement edge*. This allows the possibility that a tetrahedron has two global refinement edges. Bänsch is not explicit in how such a tetrahedron is bisected. Presumably an arbitrary selection of one of the global refinement edges is made. Assuming such a selection, there is a direct correspondence between Bänsch's marking

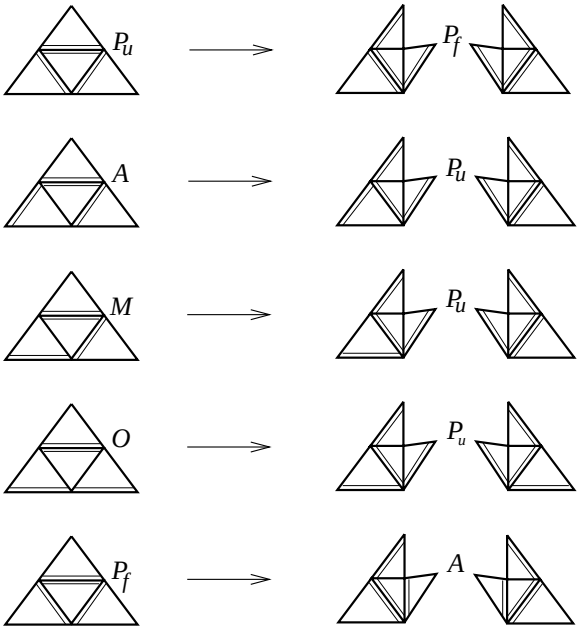

FIG. 4. *Bisection rules for marked tetrahedra.*

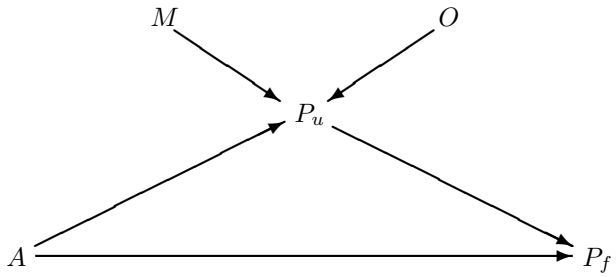

FIG. 5. *The types of tetrahedra produced by* `BisectTet`.

and ours. Bänsch refers to a tetrahedron of type A as red, one of type P as black, and does not explicitly name tetrahedra of type O or M. Our marking also involves a flag which for planar tetrahedra may be either set or unset. Bänsch addresses the same situation by referring to the parent of the tetrahedron: he singles out black tetrahedra with black parents for special treatment. These are precisely our tetrahedra of type P_f. For each type of tetrahedron Bänsch furnishes a picture showing how it should be refined. The algorithm thus described is, via the correspondences just presented, precisely equivalent to `BisectTet`. In this sense, `BisectTet` may be thought of as a convenient reformulation of Bänsch's algorithm.

The relationship to the work of Liu and Joe is less direct. In [5] they present an algorithm for the repeated bisection of an arbitrary tetrahedron. For this they introduce a special tetrahedron with a specific ordering of its vertices. The special tetrahedron and its descendants are always bisected using the longest edge as the refinement edge. They show that, for this particular tetrahedron, the mesh of nth

generation descendants is conforming and contains only one similarity class, which, in fact, depends only on n mod 3. To bisect an arbitrary tetrahedron, they choose an ordering for its vertices and use this to define a unique affine isomorphism with the special tetrahedron. The descendants of the arbitrary tetrahedron are then given as the inverse images of the descendants of the special tetrahedron under this transformation. Liu and Joe prove that the descendants fall into at most 168 distinct similarity classes and give a lower bound for a measure of their shape.

To draw the connection with `BisectTet`, we mark Liu and Joe's special tetrahedron with vertices p_0, p_1, p_2, and p_3, defined in their paper, so that $\overline{p_1p_2}$ is the refinement edge and $\overline{p_1p_3}$ and $\overline{p_2p_3}$ are the other marked edges. This is a planar marking, and we leave the flag unset, so that the special tetrahedron is marked to be type P_u. It can then be verified that applications of `BisectTet` to this marked tetrahedron and its descendants always bisect the longest edge. Indeed, for the first three generations this can be computed explicitly, and it turns out that at the third generation all of the descendants are similar to original special tetrahedron and correspondingly marked, so the bisection of the longest edge continues. Thus, for the special tetrahedron with appropriate marking, Liu and Joe's algorithm coincides with `BisectTet`. For an arbitrary tetrahedron with a selected ordering of its vertices we may use the affine isomorphism of the previous paragraph to transfer the marking from the special tetrahedron, and then we again have equivalence between the algorithm of Liu and Joe and the algorithm `BisectTet`. To summarize, given an arbitrary tetrahedron and an ordering of its vertices, we can determine a marking of this tetrahedron of type P_u, so that our algorithm and Liu and Joe's algorithm for repeated bisection of the tetrahedron are equivalent. In this sense the algorithm of [5] is equivalent to the special case of `BisectTet` applied to a tetrahedron of type P_u and its descendants. As a consequence we may apply our results from section 4 below to deduce that the descendants in fact fall into at most 36 similarity classes.

As remarked by the authors themselves, the question of how the algorithm of [5] can be applied to create a locally refined conforming mesh is far from obvious. This is the subject of [6]. The final algorithm, `QLRB`, which incorporates other algorithms called `BISECT1`, `BISECT2`, `TRANBIS`, `NEWTYPE`, `REFINE`, and `PREBISECT`, is far too complicated to describe here. We are not able to ascertain the precise relationship with `BisectTet` and the algorithm `LocalRefine` presented in the next section, although there are clearly points of similarity. In particular, the tetrahedron classes DD, DSS1, DSS2, and DSS3 introduced in [6] directly correspond to our types O, P, A, and M, respectively. In our analysis of `LocalRefine` in the next section we prove a termination theorem, Theorem 3.1. This is in fact the exact analogue of Theorem 3.1 for `QLRB` of [6].

3. A locally adaptive mesh refinement procedure. In many applications, such as adaptive finite element computations, one wishes to construct a sequence of nested conforming meshes which are adapted to a given criterion. A key step is the construction of a conforming refinement of a given conforming mesh, in which a selected subset of elements has been refined. In this section we describe an algorithm based on `BisectTet` to accomplish this.

Before stating the algorithm we fix some terminology. A *mesh* $\mathcal{T}$ of a domain Ω in $\mathbb{R}^3$ is a set of closed tetrahedra with disjoint interiors and union $\bar{\Omega}$. The *vertex set* of $\mathcal{T}$ is $\mathcal{V}(\mathcal{T}) = \bigcup\{\mathcal{V}(\tau) : \tau \in \mathcal{T}\}$. The *edge set* $\mathcal{E}(\mathcal{T})$ and the *face set* $\mathcal{F}(\mathcal{T})$ are defined similarly. A mesh is *conforming* if the intersection of two distinct tetrahedra is either a common face, a common edge, a common vertex, or empty. If $\tau \in \mathcal{T}$ and

$\nu \in \mathcal{V}(\mathcal{T})$, we say that ν is a *hanging node* of τ if $\nu \in \tau \setminus \mathcal{V}(\tau)$. A mesh is conforming if no tetrahedron in it has a hanging node and every face of every tetrahedron in the mesh either belongs to the boundary or is a face of another tetrahedron in the mesh. A mesh is *marked* if each tetrahedron in it is marked. A marked conforming mesh is *conformingly-marked* if each face has a unique marked edge (that is, when a face is shared by two tetrahedra, the marked edge is the same for both). Given an arbitrary conforming mesh the following marking procedure yields a conformingly-marked mesh. Strictly order the edges of the mesh in an arbitrary but fixed manner, for example, by length with a well-defined tie-breaking rule. Then choose the maximal edge of each tetrahedron as its refinement edge and the maximal edge of each face as its marked edge. Unset the flag on all tetrahedra. (The assumption that the coarse mesh has no flagged tetrahedra will be used in the analysis below.)

We now state the main algorithm of this section.

ALGORITHM $\mathcal{T}' = \texttt{LocalRefine}\ (\mathcal{T}, \mathcal{S})$

input: conformingly-marked mesh $\mathcal{T}$ and $\mathcal{S} \subset \mathcal{T}$

output: conformingly-marked mesh $\mathcal{T}'$

1. $\bar{\mathcal{T}} = \texttt{BisectTets}(\mathcal{T}, \mathcal{S})$
2. $\mathcal{T}' = \texttt{RefineToConformity}(\bar{\mathcal{T}})$

The algorithm in the first step, `BisectTets`, is trivial; we simply bisect each tetrahedron in $\mathcal{S}$:

$$\texttt{BisectTets}(\mathcal{T}, \mathcal{S}) = (\mathcal{T} \setminus \mathcal{S}) \cup \bigcup_{\tau \in \mathcal{S}} \texttt{BisectTet}(\tau).$$

In the second step, we perform further refinement as necessary to obtain a conforming mesh:

ALGORITHM $\mathcal{T}' = \texttt{RefineToConformity}(\mathcal{T})$

input: marked mesh $\mathcal{T}$

output: marked mesh $\mathcal{T}'$ without hanging nodes

1. set $\mathcal{S} = \{\tau \in \mathcal{T} \,|\, \tau \text{ has a hanging node}\}$
2. if $\mathcal{S} \neq \emptyset$ then
   $\bar{\mathcal{T}} = \texttt{BisectTets}(\mathcal{T}, \mathcal{S})$
   $\mathcal{T}' = \texttt{RefineToConformity}(\bar{\mathcal{T}})$
3. else
   $\mathcal{T}' = \mathcal{T}$

The recursion in the Algorithm `RefineToConformity` could conceivably continue forever. Moreover, even if the recursion terminates, the output mesh may not be conforming (a mesh without hanging nodes can nonetheless be nonconforming; cf. Figure 6). However, we shall prove that the recursion does terminate in the application of `RefineToConformity` in Algorithm `LocalRefine`, and that the resulting output mesh is conformingly-marked. Moreover, we shall bound the amount of refinement which can occur before termination. To state this result precisely, we consider an initial marked mesh $\mathcal{T}_0$ and set $\mathcal{Q}_0 = \mathcal{T}_0$, and $\mathcal{Q}_{k+1} = \texttt{BisectTets}(\mathcal{Q}_k, \mathcal{Q}_k)$, $k = 0, 1, \ldots$. Thus $\mathcal{Q}_1$ consists of all children of tetrahedra in the initial mesh, $\mathcal{Q}_2$ all grandchildren, etc. We assign *generation* k to all tetrahedra in $\mathcal{Q}_k$.

THEOREM 3.1. *Let $\mathcal{T}_0$ be a conformingly-marked mesh with no flagged tetrahedra. For $k = 0, 1, \ldots$, choose $\mathcal{S}_k \subset \mathcal{T}_k$ arbitrarily, and set $\mathcal{T}_{k+1} = \texttt{LocalRefine}(\mathcal{T}_k, \mathcal{S}_k)$. Then for each k, the application of* `RefineToConformity` *from within* `LocalRefine` *terminates producing a conformingly-marked mesh, and each tetrahedron in $\mathcal{T}_k$ has a generation of at most $3k$. Moreover, if the maximum generation of a tetrahedron in*

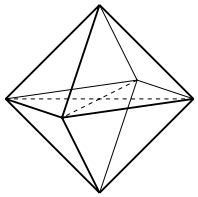

FIG. 6. *A nonconforming mesh without hanging nodes (the barycenter is* not *a vertex of the mesh).*

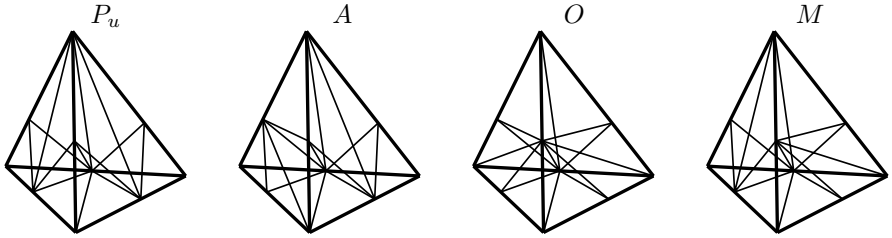

FIG. 7. *The meshes obtained via three applications of* `BisectTet` *beginning with a tetrahedra of type P_u, A, O, and M.*

$\mathcal{T}_k$ is less than $3m$ for some integer m, then the maximum generation of a tetrahedron in $\mathcal{T}_{k+1}$ is less than or equal to $3m$.

For the proof of the theorem, we need a classification of the edges that arise from repeated bisection of an unflagged marked tetrahedron τ. Let $\mathcal{Q}_0^\tau = \{\tau\}$ and define the meshes $\mathcal{Q}_k^\tau$ in analogy to the definition of the $\mathcal{Q}_k$ above (so $\mathcal{Q}_k^\tau$ contains all descendants of τ of generation k). We define $\mathcal{E}_k(\tau) = \mathcal{E}(\mathcal{Q}_{3k}^\tau)$ and refer to these as the edges of generation k. Thus the edges of generation 0 are exactly the edges of τ itself, and, referring to Figure 7, we verify that the edges of generation 1 are precisely the following 25 line segments:

- the line segment connecting the midpoint of the refinement edge to the midpoint of the opposite edge;
- for each face, the line segment connecting the midpoint of the marked edge to the opposite vertex;
- for each face, the two line segments connecting the midpoint of the marked edge to the midpoints of the two nonmarked edges;
- for each edge, its two children.

LEMMA 3.2. *Let τ be an unflagged marked tetrahedron. Then for $k = 1, 2, \ldots$, the mesh $\mathcal{Q}_{3k}^\tau$, consisting of all descendants of τ of generation $3k$, is conformingly-marked. If τ is of type P_u, then all the tetrahedra in $\mathcal{Q}_{3k}^\tau$ are of type P_u, and otherwise all the tetrahedra in $\mathcal{Q}_{3k}^\tau$ are of type A.*

Proof. It is clear from Figure 7 that $\mathcal{Q}_3^\tau$ is conforming. Moreover, the definition of `BisectTet` insures $\mathcal{Q}_3^\tau$ is conformingly-marked (because whenever a face is introduced, it is marked identically in the tetrahedra containing it). From the diagram in Figure 5, we see that tetrahedra in $\mathcal{Q}_3^\tau$ are all either type P_u or type A, depending on whether τ is type P_u. This verifies the lemma in case $k = 1$.

If $\tau' \in \mathcal{Q}_3^\tau$, then the mesh of third generation descendants of τ' is, by the same argument, conformingly-marked and uniformly of type P_u or A. Because $\mathcal{Q}_3^\tau$ is it-

TABLE 1
The types of tetrahedra and generation of edges occurring in the meshes $\mathcal{Q}_k$.

Generations of tetrahedra	Types of tetrahedra		Generations of edges
0	P_u	nonplanar	0
1	P_f	P_u	0,1
2	A	P_f	0,1
3	P_u	A	1
4	P_f	P_u	1,2
5	A	P_f	1,2
6	P_u	A	2
⋮	⋮	⋮	⋮
$3k$	P_u	A	k
$3k+1$	P_f	P_u	$k, k+1$
$3k+2$	A	P_f	$k, k+1$
$3(k+1)$	P_u	A	$k+1$

self conformingly-marked, the mesh obtained by combining all these meshes is again conformingly-marked, verifying the lemma in case $k = 2$. By induction we obtain the result for all k. □

If τ' is a generation $3k$ descendant of an unflagged marked tetrahedron τ, then, by definition, all of the edges of τ' have generation k. The next lemma determines the generations of the edges of descendants of generation $3k+1$ and $3k+2$.

LEMMA 3.3. *Let τ be an unflagged marked tetrahedron and τ' a descendant of τ of generation $3k+1$ or $3k+2$. Then the edges of τ' all have generation k or $k+1$.*

Proof. The tetrahedron τ' is either a child or a grandchild of an unflagged tetrahedron of generation $3k$. From Figure 7, it is easy to see that every edge of a child or a grandchild of an unflagged tetrahedron is either an edge of that tetrahedron or an edge of one of its great grandchildren. Thus each edge of τ' is an edge of a tetrahedron whose generation is either $3k$ or $3k+3$. □

Returning to Theorem 3.1, we easily deduce the following proposition from the preceding lemmas.

PROPOSITION 3.4. *Let $\mathcal{T}_0$ be a conformingly-marked mesh with no flagged tetrahedra, and let $\mathcal{Q}_k$ denote the mesh of all descendants of generation k of tetrahedra in $\mathcal{T}_0$. Then the types of tetrahedra and the generation of edges of occurring in $\mathcal{Q}_k$ are as shown in Table* 1*. Moreover, the meshes $\mathcal{Q}_{3k}$ are conformingly-marked.*

Proof of Theorem 3.1. The proof will proceed in several steps. We use the terminology *descendant mesh of $\mathcal{T}_0$* to refer to any mesh that can arise from $\mathcal{T}_0$ by repeated application of `BisectTet`.

Step 1. If $\mathcal{T}$ is any descendant mesh of $\mathcal{T}_0$ which has maximal tetrahedron generation $3k$, then the application of `BisectTets` in step 2 of `RefineToConformity`($\mathcal{T}$) returns a mesh which again has maximal tetrahedron generation $3k$.

Proof. We refer to Table 1 to see that all the edges of tetrahedra in $\mathcal{T}$ are of most generation k. Now if a tetrahedron in $\mathcal{T}$ has a hanging node, then the edge of the tetrahedron on which the hanging node lies must have generation $k-1$ or less (since its children are also edges in the mesh and so have generation k or less). Hence, again with reference to the table, a tetrahedron with a hanging node has generation strictly less than $3k$. That is, the set $\mathcal{S}$ defined in step 1 of `RefineToConformity` consists of tetrahedra of generation less than $3k$. Consequently the mesh output from `BisectTets` in step 2 of `RefineToConformity`, again has maximal tetrahedron generation $3k$.

Step 2. If $\mathcal{T}$ is any descendant mesh of $\mathcal{T}_0$ which has maximal tetrahedron gener-

ation $3k$, then every mesh constructed in the recursive application of the algorithm `RefineToConformity`($\mathcal{T}$) has maximal tetrahedron generation $3k$ and, moreover, the algorithm terminates.

Proof. Indeed new tetrahedra are only introduced by the application of `BisectTets` in step 2 of `RefineToConformity`, so the generation bound follows from the previous step. Since there are only finitely many tetrahedra of generation $3k$, the algorithm must terminate.

Step 3. Each tetrahedron in $\mathcal{T}_k$ has generation at most $3k$.

Proof. The proof is by induction on k, with the case $k = 0$ being the obvious final step. By the induction hypothesis, $\mathcal{T}_{k-1}$ consists of tetrahedra of generation at most $3k - 3$. Hence, the mesh $\bar{\mathcal{T}}$ output from `BisectTets` in step 1 of `LocalRefine`($\mathcal{T}_{k-1}, \mathcal{S}_{k-1}$) has maximum tetrahedron generation $3k - 2 \le 3k$, and the result follows from the preceding claim.

Step 4. The output mesh is conformingly-marked.

Proof. This follows easily from the fact that the output mesh is a descendant of a conformingly-marked mesh and has no hanging nodes.

Step 5. The last sentence of the theorem follows directly from step 2. ∎

4. Similarity classes. In [7] Maubach presented an algorithm, which we refer to as `BisectSimplex`, for the bisection of an arbitrary n-simplex in $\mathbb{R}^n$. After recalling this algorithm, we shall show that in the special case $n = 3$, it is essentially equivalent to `BisectTet`, when `BisectTet` is restricted to tetrahedra of types A and P as input. Define a *tagged simplex* in $\mathbb{R}^n$ as an ordered pair, $\mathfrak{t} = \big((x_0, x_1, \dots, x_n), d\big)$, where $(x_0, x_1, \dots, x_n)$ is an ordered $(n+1)$-tuple of points in general position in $\mathbb{R}^n$ (the vertices of the simplex), and $d \in \{1, 2, \dots, n\}$ is a *tag* ($\overline{x_0 x_d}$ is the refinement edge of $\mathfrak{t}$). With this terminology, Maubach's algorithm may be stated as follows.

ALGORITHM $\{\mathfrak{t}^{(1)}, \mathfrak{t}^{(2)}\} =$ `BisectSimplex`($\mathfrak{t}$)

input: tagged n-simplex $\mathfrak{t}$

output: tagged n-simplices $\mathfrak{t}^{(1)}$ and $\mathfrak{t}^{(2)}$.

1. set $d' = \begin{cases} d-1, & d > 1 \\ n, & d = 1 \end{cases}$
2. create the new vertex $z = \frac{1}{2}(x_0 + x_d)$
3. set $\mathfrak{t}^{(1)} = \big((x_0, x_1, \dots, x_{d-1}, z, x_{d+1}, ..., x_n), d'\big)$
4. set $\mathfrak{t}^{(2)} = \big((x_1, x_2, \dots, x_d, z, x_{d+1}, ..., x_n), d'\big)$

We now define a correspondence between tagged simplices in $\mathbb{R}^3$ and marked tetrahedra of types P and A, and we show that repeated application of either `BisectTet` or `BisectSimplex` to the same geometric tetrahedron yields the same sequence of descendant tetrahedra. Let $\mathcal{T}_T$ be the set of all tagged 3-simplices and $\mathcal{T}_M$ be the set of all marked tetrahedra; also, let $\mathcal{T}_a \subset \mathcal{T}_M$ denote the set of marked tetrahedra of type A or P. Define a mapping $\mathcal{F} : \mathcal{T}_T \to \mathcal{T}_M$ as follows.

For $\mathfrak{t} = \big((x_0, x_1, x_2, x_3), d\big) \in \mathcal{T}_T$, set $\mathcal{F}(\mathfrak{t}) = (\mathcal{V}(\tau), r_\tau, m_\varphi, f_\tau)$ with $\mathcal{V}(\tau) = \{x_0, x_1, x_2, x_3\}$ and

$$r_\tau = x_0 x_1, \quad m_\varphi = \begin{cases} \overline{x_0 x_3} & \text{if } \varphi = x_0 x_2 x_3, \\ \overline{x_1 x_3} & \text{if } \varphi = x_1 x_2 x_3, \end{cases} \quad f_\tau = 1 \quad \text{if } d = 1;$$

$$r_\tau = x_0x_2, \quad m_\varphi = \begin{cases} \overline{x_0x_1} & \text{if } \varphi = x_0x_1x_3, \\ \overline{x_1x_2} & \text{if } \varphi = x_1x_2x_3, \end{cases} \quad f_\tau = 0 \quad \text{if } d = 2;$$

$$r_\tau = x_0x_3, \quad m_\varphi = \begin{cases} \overline{x_0x_2} & \text{if } \varphi = x_0x_1x_2, \\ \overline{x_1x_3} & \text{if } \varphi = x_1x_2x_3, \end{cases} \quad f_\tau = 0 \quad \text{if } d = 3.$$

Note that $\mathcal{F}(\mathtt{t})$ has type P_f, P_u, or A, when $d = 1$, 2, or 3, respectively. Consequently, $\mathcal{F}(\mathcal{T}_T) \subset \mathcal{T}_a$. In fact, we have the following proposition.

PROPOSITION 4.1. *The mapping $\mathcal{F}$ maps $\mathcal{T}_T$ onto $\mathcal{T}_a$ and is precisely* 2 *to* 1.

Proof. Given a marked tetrahedron $\tau = (\{v_0, v_1, v_2, v_3\}, r_\tau, m_\varphi, f_\tau) \in \mathcal{T}_a$, we define two tagged 3-simplices in its preimage under $\mathcal{F}$ as follows.

If τ has type A, we assume, without loss of generality, that its refinement and nonrefinement edges are $\overline{v_0v_1}$ and $\{\overline{v_0v_2}, \overline{v_1v_3}\}$, respectively. To get the first tagged simplex, set $d = 3$ and choose $x_0 = v_0$ and $x_d = v_1$ (the end points of the refinement edge). Set $x_1 = v_3$ (the second endpoint of the marked nonrefinement edge containing x_d) and $x_2 = v_2$. To obtain the second tagged simplex, we reverse x_0 and x_3 and x_1 and x_2. Thus, the two tagged 3-simplices corresponding to the given marked tetrahedron are $\big((v_0, v_3, v_2, v_1), 3\big)$ and $\big((v_1, v_2, v_3, v_0), 3\big)$. It is straightforward to check that $\mathcal{F}$ maps each of these tagged simplices to τ and that these are the only preimages of τ under $\mathcal{F}$. The argument is similar in the cases of τ of type P_u or P_f. In the P_u case, we take the numbering so that the refinement and marked nonrefinement edges are $\overline{v_0v_1}$ and $\{\overline{v_0v_2}, \overline{v_1v_2}\}$, respectively, and find the two preimages $\big((v_0, v_2, v_1, v_3), 2\big)$ and $\big((v_1, v_2, v_0, v_3), 2\big)$. In the P_f case with refinement and marked nonrefinement edges $\overline{v_0v_1}$ and $\{\overline{v_0v_2}, \overline{v_1v_2}\}$, the preimages are $\big((v_0, v_1, v_3, v_2), 1\big)$ and $\big((v_1, v_0, v_3, v_2), 1\big)$. ∎

The following theorem exhibits the relation between the algorithms `BisectTet` and `BisectSimplex`.

THEOREM 4.2. *The following diagram commutes.*

$$\begin{array}{ccc} \mathcal{T}_T & \xrightarrow{\ \mathcal{F}\ } & \mathcal{T}_M \\ {\scriptstyle \texttt{BisectSimplex}}\downarrow & & \downarrow{\scriptstyle \texttt{BisectTet}} \\ \mathcal{T}_T \times \mathcal{T}_T & \xrightarrow{\ \mathcal{F}\times\mathcal{F}\ } & \mathcal{T}_M \times \mathcal{T}_M \end{array}$$

Proof. Suppose $\mathtt{t} = \big((x_0, x_1, x_2, x_3), 3\big) \in \mathcal{T}_T$, and $\{\mathtt{t}^{(1)}, \mathtt{t}^{(2)}\} \in \mathcal{T}_T \times \mathcal{T}_T$ is produced by `BisectSimplex`($\mathtt{t}$). Then $\mathtt{t}^{(1)} = \big((x_0, x_1, x_2, z), 2\big)$ with $z = (x_0 + x_3)/2$, and so $\mathcal{F}(\mathtt{t}^{(1)})$ yields the marked tetrahedron $(\{x_0, x_1, x_2, z\}, \overline{x_0x_2}, \{\overline{x_0x_1}, \overline{x_1x_2}\}, 0)$. (Here, only the markings for the nonrefinement faces of the tetrahedron are specified in m_φ with the convention that the given edges are marked for the nonrefinement face containing them.) On the other hand, $\mathcal{F}(\mathtt{t}) = (\{x_0, x_1, x_2, x_3\}, \overline{x_0x_3}, \{\overline{x_0x_2}, \overline{x_1x_3}\}, 0)$ which is a tetrahedron of type A, and one of the marked tetrahedra produced by applying `BisectTet` to this tetrahedron is $\mathcal{F}(\mathtt{t}^{(1)})$. A similar verification is easily carried out for the other cases. ∎

Theorem 4.2 implies that if `BisectSimplex` is applied m times to a tagged 3-simplex, the images under the mapping $\mathcal{F}$ of the 2^m descendents are exactly the descendents obtained by applying `BisectTet` m times to the image under $\mathcal{F}$ of the parent. The following corollary is thus immediate.

COROLLARY 4.3. *The 2^m geometric 3-simplices obtained by applying the algorithm* `BisectSimplex` *m times to $\mathtt{t} \in \mathcal{T}_T$ are exactly the same as those obtained by applying* `BisectTet` *m times to $\mathcal{F}(\mathtt{t})$.*

Our next goal is a bound on the number of similarity classes produced by repeated application of algorithm `BisectSimplex`. (Two simplices are said to belong to the same similarity class if one can be transformed to the other via a dilation and a rigid motion.) This will be based on the following technical result from [7]. (Theorem 4.1 of [7] states this result in less generality, but the same proof applies to the statement given here.)

LEMMA 4.4. *Let* $\mathfrak{t}'$ *be a descendent of the tagged* n*-simplex*

$$\mathfrak{t} = \big((0, e_1, e_1 + e_2, \dots, e_1 + e_2 + \cdots + e_n), n\big)$$

obtained via k *applications of* `BisectSimplex`*, where* $\mathcal{B} = \{e_1, e_2, \dots, e_n\}$ *is an arbitrary basis of* $\mathbb{R}^n$*. Define integers* $q \geq 0$ *and* $r \in \{0, \dots, n-1\}$ *by* $k = qn + r$*. Let* $(x_0, x_1, \dots, x_n)$ *be the ordered vertices of* $\mathfrak{t}'$ *and define* $y_i = x_i - x_0$ *for* $i = 1, 2, \dots, n$*. Then, there exist two linear mappings* π *and* R *from* $\mathbb{R}^n$ *to* $\mathbb{R}^n$ *such that* $\{\pi(e_1), \pi(e_2), \dots, \pi(e_n)\}$ *is a permutation of the basis vectors of* $\mathcal{B}$*, and* $R(e_i) = \pm e_i$*,* $1 \leq i \leq n$*, with*

$$y_i = \left(\frac{1}{2}\right)^q \alpha_i R\pi \sum_{j=1}^{i} e_j, \qquad 1 \leq i \leq n,$$

where

$$\alpha_i = \begin{cases} 1, & 1 \leq i \leq n - r, \\ \frac{1}{2}, & n - r + 1 \leq i \leq n. \end{cases}$$

THEOREM 4.5. *The number of similarity classes of* n*-simplices produced by the repeated application of* `BisectSimplex` *is bounded by* $nn!2^{n-2}$*.*

Proof. Without loss of generality we may assume that the initial simplex has tag n, because an arbitrary tagged simplex is a descendant of such a simplex. For example, $\big((x_0, x_1, x_2, \dots, x_n), n-1\big)$ is a child of $\big((x_0, x_1, x_2, \dots, 2x_n - x_0), n\big)$. Moreover we may translate so that the first vertex is at the origin, and thus the initial simplex can be taken to be of the form $\mathfrak{t}$ assumed in the lemma.

From this immediately get an upper bound of $nn!2^n$ similarity classes for the descendants, since there are $n!$ possibilities for the permutations π, 2^n possibilities for the reflections R, and exactly n different vectors α_i. Noting that two different reflections, R and $-R$, give n-simplices in the same similarity class, the bound can be reduced by a factor of 2 to $nn!2^{n-1}$. To prove the theorem it suffices to reduce this by another factor of 2, which we shall do by showing that each n-simplex produced is a translate of another.

Using the notations introduced in Lemma 4.4 and noting that q does not play a role in the count for the number of similarity classes of tetrahedra, we will assume $q = 0$ without any loss of generality. Define the mappings $\hat{\pi} : \mathbb{R}^n \to \mathbb{R}^n$ and $\hat{R} : \mathbb{R}^n \to \mathbb{R}^n$ by

$$\hat{\pi}(e_j) = \begin{cases} \pi(e_{n-r+1-j}), & 1 \leq j \leq n - r, \\ \pi(e_j), & n - r + 1 \leq j \leq n, \end{cases} \tag{1}$$

$$\hat{R}\pi\,(e_j) = \begin{cases} -R\pi(e_j), & 1 \leq j \leq n - r, \\ R\pi(e_j), & n - r + 1 \leq j \leq n, \end{cases} \tag{2}$$

for $j \in \{1, 2, \dots, n\}$. Note that $\hat{\pi}$ is a permutation and $\hat{R}$ a reflection relative to $\mathcal{B}$. The ordered set $(0, y_1, y_2, \dots, y_n)$ represents the vertices of the tagged n-simplex $\mathfrak{t}'$. Denote the ordered set of vertices of another tagged n-simplex $\hat{\mathfrak{t}}$ by $(0, \hat{y}_1, \hat{y}_2, \dots, \hat{y}_n)$ with $\hat{y}_i = \alpha_i \hat{R}\hat{\pi} \sum_{j=1}^{i} e_i$ for $1 \le i \le n$. Let $\Phi : \mathbb{R}^n \to \mathbb{R}^n$ be the translation defined by

$$\Phi(x) = x - R\pi \sum_{j=1}^{n-r} e_j = x - y_{n-r}.$$

Then $\Phi(0) = -R\pi \sum_{j=1}^{n-r} e_j$ and

$$\Phi(y_i) = \begin{cases} -R\pi \sum\limits_{j=i+1}^{n-r} e_j, & 1 \le i < n-r, \\ 0, & i = n-r, \\ -\frac{1}{2} R\pi \sum\limits_{j=1}^{n-r} e_j + \frac{1}{2} R\pi \sum\limits_{j=n-r+1}^{i} e_j, & n-r+1 \le i \le n. \end{cases}$$

Using (1), (2), and the definition of $\hat{y}_i$, we get $\Phi(0) = \hat{y}_{n-r}$ and

$$\Phi(y_i) = \begin{cases} \hat{y}_{n-r-i}, & 1 \le i < n-r, \\ 0, & i = n-r, \\ \hat{y}_i, & n-r+1 \le i \le n. \end{cases}$$

Thus, $\mathfrak{t}'$ is related to $\hat{\mathfrak{t}}$ via the translation Φ as desired. □

For $n = 2$ the bound of four similarity classes furnished by the theorem is easily seen to be sharp. For $n = 3$, the bound is 36. By direct computation on a particular tetrahedron we have verified that the bound of 36 is sharp. (For example, if $\mathfrak{t} = \big((v_0, v_1, v_2, v_3), 3\big)$, where $v_0 = (0, 0, 0)$, $v_1 = (23, 0, 0)$, $v_2 = (7, 0, 11)$, and $v_3 = (17, 5, 33)$, then the upper bound of 36 is attained at the sixth generation.) Maubach [8] has announced a proof that the bound of $nn!2^{n-2}$ is sharp for all n.

In view of Proposition 4.1 and Theorem 4.2, the above result applies to bisection of marked tetrahedra of types A and P: repeated application of `BisectTet` starting from such a marked tetrahedron will produce at most 36 similarity classes. Since one application of `BisectTet` to a tetrahedron of type O or M produces children of type P_u, the repeated bisection of such a tetrahedron will produce at most 72 similarity classes of tetrahedra. In particular, for an arbitrary initial mesh of marked tetrahedra, only finitely many similarity classes will arise in its descendant meshes.

5. Examples. In this section, we give some examples of adapted tetrahedral meshes generated using `LocalRefine`. A coarse initial mesh $\mathcal{T}_0$ is chosen and the meshes $\mathcal{T}_k$ are generated using $\mathcal{T}_k = \texttt{LocalRefine}(\mathcal{T}_{k-1}, \mathcal{S}_{k-1})$ as in section 3 with different criteria being used to determine the sets $\mathcal{S}_k$ in each example.

Example 1. The first example is adapted from Maubach [7]. Let $\mathcal{T}_0$ be the subdivision of the cube $[0, 1]^3$ into six congruent tetrahedra. We choose the longest edge of each face as its marked edge. It can easily be verified that all six tetrahedra are of type A and they belong to the same similarity class. Let

$$\mathcal{H} = \left\{ (x, y, z) \in \mathbb{R}^3 \mid \left(x - \frac{1}{2}\right)^2 + \left(y - \frac{1}{2}\right)^2 + \left(z - \frac{1}{2}\right)^2 = \frac{1}{16} \text{ with } x \ge \frac{1}{2} \right\}$$

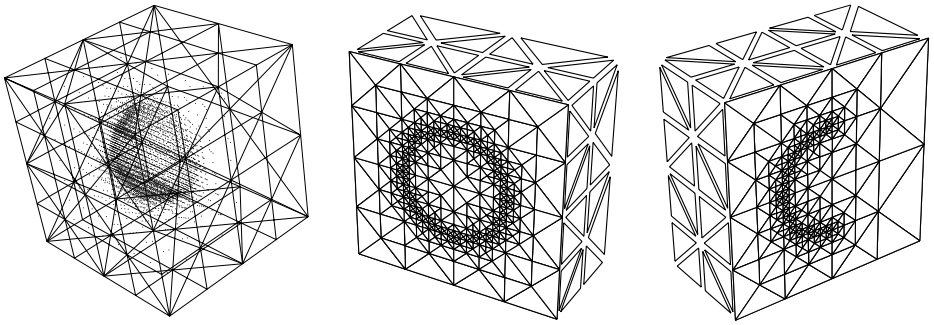

FIG. 8. *The mesh $\mathcal{T}_{16}$ of Example* 1. *The first view shows the vertices of the mesh, the second a cut along the plane $x = 1/2$, and the third a cut along the plane $y = 1/2$.*

be a hemisphere embedded in the cube. We choose $\mathcal{S}_{k-1} = \{\tau \in \mathcal{T}_{k-1} | \tau \cap \mathcal{H} \neq \emptyset\}$, so that we attempt to adapt the meshes to the hemisphere $\mathcal{H}$. Figure 8 shows different views of $\mathcal{T}_{16}$ having $25,448$ tetrahedra. The local adaptivity around $\mathcal{H}$ is clear.

The algorithm `LocalRefine` has been incorporated into a code for solving second order elliptic boundary value problems (cf. [9]). Error estimators are used to determine the sets $\mathcal{S}_k$ in the code, leading to refinement in regions where the gradient of the solution changes rapidly.

Example 2. This example is a test problem for which the exact solution is known to be

$$u_{\text{ex}} = (x^2 - x)(y^2 - y)(z^2 - z)e^{-100[(x-1/4)^2+(y-1/4)^2+(z-1/4)^2]},$$

and $\mathcal{T}_0$ was taken to be a subdivision of $[0,1]^3$ having 96 tetrahedra. The problem was solved using piecewise linear finite elements, and a full multigrid solver based on the sequence of adaptively defined meshes. Note that u_{ex} varies very rapidly near the point $(1/4, 1/4, 1/4)$, and has relatively slow variation in other parts of the domain. This behavior is captured well by the adaptive mesh refinement process, as seen in $\mathcal{T}_6$ pictured in Figure 9.

Figure 10 shows a log-log plot of the errors in energy and L^2 norms against $\mathcal{V}(\mathcal{T})^{-1/3}$, which is an indicator of mesh size for locally refined meshes, using both adaptive and uniform refinement. (Lines with slopes 1 and 2 are shown for easy comparison.) Using uniform refinement, the finest mesh has $68,705$ vertices and a relative percentage energy error of approximately 15.85% while the maximum number of vertices using adaptive refinement are $62,738$ and the corresponding relative percentage energy error is 4.95%.

Example 3. As a final example, we consider a problem that arose in numerical relativity concerning compatible initial metric data for the collision of two black holes. This involves a nonlinear elliptic boundary value problem which we solve with piecewise linear finite elements, an outer Newton iteration, and an inner multigrid iteration based on the sequence of adapted meshes. For details, see [9]. The domain consists of a ball about the origin of radius $64\sqrt{3}$ minus two disjoint balls of radius $\sqrt{3}/2$ and $\sqrt{3}$ centered at $(0, 0, -2\sqrt{3})$ and $(0, 0, 2\sqrt{3})$, respectively. Beginning with the mesh $\mathcal{T}_0$ containing 585 vertices and $2,982$ tetrahedra the code produced an adapted mesh $\mathcal{T}_5$ with $63,133$ vertices and $346,084$ tetrahedra. In Figures 11 and 12, which is a zoom of the previous figure, we show results computed on the intermediate mesh $\mathcal{T}_3$ with

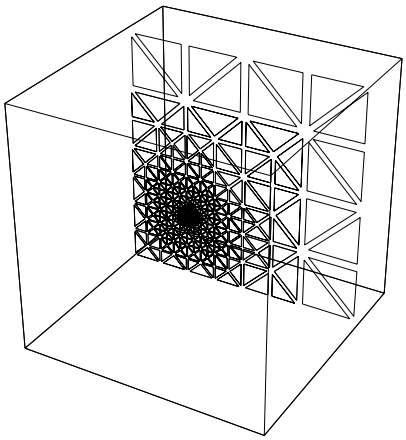

FIG. 9. *Cut along the the plane $x = 1/4$ of $\mathcal{T}_6$ of Example* 2 *(there are* 11,418 *tetrahedra and* 2,116 *vertices in the mesh).*

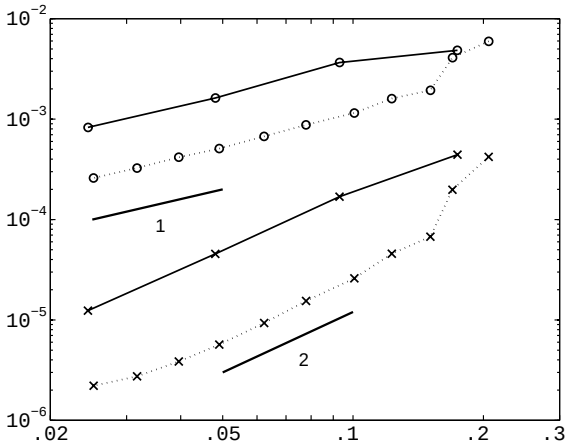

FIG. 10. *The energy (○) and L^2 (×) errors and rates for example* 2. *The solid lines denote the energy (respectively, L^2) errors for uniform refinement while the dotted lines are for adaptive refinement.*

13,899 vertices and 75,300 tetrahedra.

The figures show a contour plot of the solution on the plane $x = y$ (the plot shade is keyed to the value of the solution). This is easier to interpret in the color version, which can be found in [9]. The intersections of the tetrahedra with the plane are shown slightly shrunken to improve visibility. Also shown are the mesh edges which intersect the boundary. Next, Figure 13 gives evidence of the mesh quality in the initial and final meshes. The shape measure used is the ratio of the circumscribed to inscribed spheres, scaled so that an equilateral tetrahedron has shape measure one. The first histogram shows that in the initial mesh more than 80% of elements have measure less than 2, while the worst elements have quality around 3.5. In the final mesh, 72% of the elements have measure less than 2, while 84% have measure less than 2.5, and the worst elements have measure about 5.4.

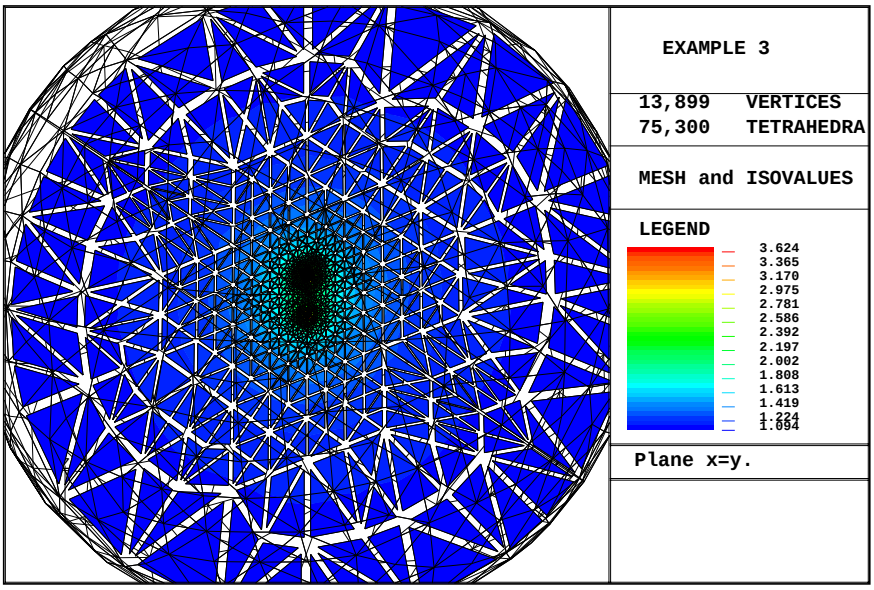

FIG. 11. *Solution to a binary black hole problem.*

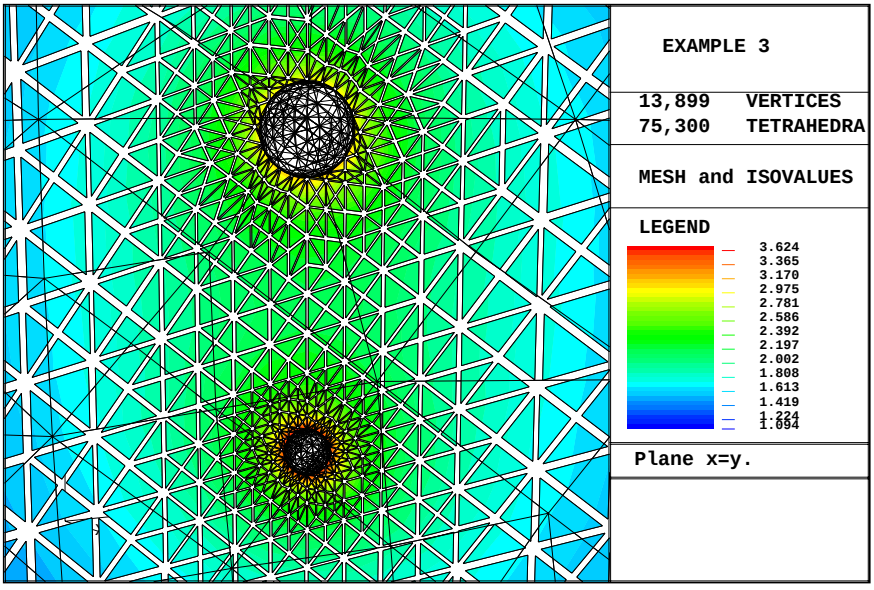

FIG. 12. *Solution to a binary black hole problem, zoom.*

Finally, Figure 14 shows a plot of the total CPU time for the solution of the boundary value problem including the mesh refinement, error estimation, outer iteration, and the multigrid solve on a 1993 DEC 3000 model 500 with a single 150 MHz Alpha processor versus number of mesh vertices. The plot clearly shows that the computation time is very nearly proportional to the number of vertices in the mesh.

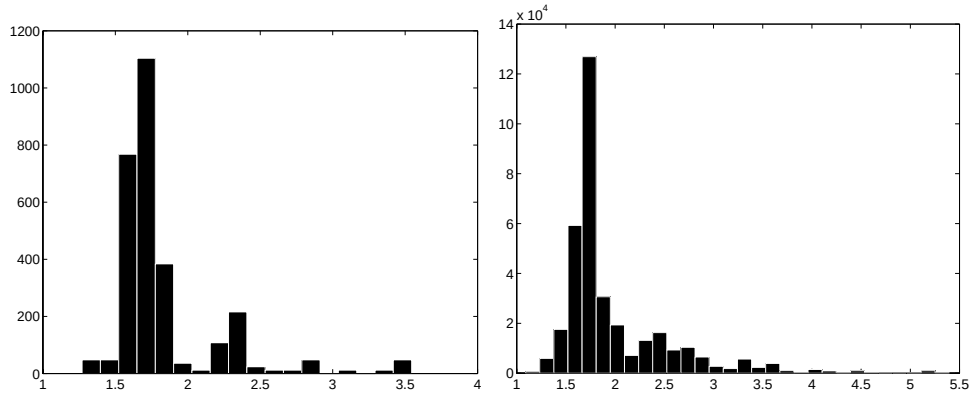

FIG. 13. *Shape measure histograms. Left: coarse mesh with* 2,982 *tetrahedra. Right: fine mesh with* 346,084 *tetrahedra.*

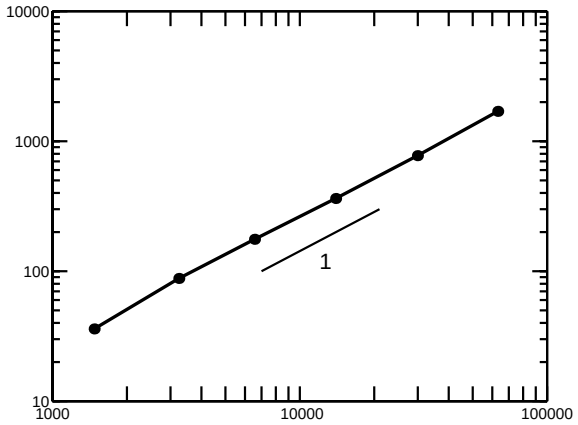

FIG. 14. *CPU seconds, on the y-axis, versus number vertices, on the x-axis, for a binary black hole problem.*

6. Conclusion. In section 2 we introduced the notion of a marked tetrahedron, and presented algorithm `BisectTet` for the bisection of a marked tetrahedron into two children. We discussed its relation which other algorithms in the literature. In section 3 we showed how, beginning with an arbitrary conforming tetrahedral mesh, the algorithm `BisectTet` can be applied to create locally refined conforming tetrahedral meshes, and we bounded the amount of additional refinement that might be required to obtain conformity. In the following section we showed that repeated application of `BisectTet` to a given tetrahedron and its descendants generates at most 36 or 72 dissimilar tetrahedra in all (depending on the marking of the original tetrahedron). In doing so, we showed that for some markings `BisectTet` is, in a certain specific sense, equivalent to the three-dimensional case of Maubach's algorithm for repeated bisection of n-simplices, and for this algorithm we showed that at most $n\,n!2^{n-2}$ dissimilar simplices are created. We closed the paper with numerical examples of the performance of the algorithm.

REFERENCES

[1] A. Adler *On the bisection method for triangles*, Math. Comp., 40 (1983), pp. 571–574.

[2] R. Bank, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations*, Users' Guide 7.0., SIAM, Philadelphia, 1994.

[3] E. Bänsch, *Local mesh refinement in* 2 *and* 3 *dimensions*, Impact Comput. Sci. Engrg., 3 (1991), pp. 181–191.

[4] Jürgen Bey, *Tetrahedral grid refinement*, Computing, 55 (1995), pp. 271–288.

[5] A. Liu and B. Joe, *On the shape of tetrahedra from bisection*, Math. Comp., 63 (1994), pp. 141–154.

[6] A. Liu and B. Joe, *Quality local refinement of tetrahedral meshes based on bisection*, SIAM J. Sci. Comput., 16 (1995), pp. 1269–1291.

[7] J. M. Maubach, *Local bisection refinement for N-simplicial grids generated by reflection*, SIAM J. Sci. Comput., 16 (1995), pp. 210–227.

[8] J. M. Maubach, *The Amount of Similarity Classes Created by Local n-Simplicial Bisection Refinement*, 1996, manuscript.

[9] A. Mukherjee, *An Adaptive Finite Element Code for Elliptic Boundary Value Problems in Three Dimensions with Applications in Numerical Relativity*, Ph.D. thesis, The Pennsylvania State University, University Park, PA, 1996.

[10] M. E. G. Ong, *Hierarchical Basis Preconditioners for Second Order Elliptic Problems in Three Dimensions*, Ph.D. thesis, University of Washington, Seattle, 1989.

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

[12] M. C. Rivara and C. Levin, *A* 3*-D refinement algorithm suitable for adaptive and multi-grid techniques*, Comm. Appl. Numer. Methods., 8 (1992), pp. 281–290.

[13] S. Zhang, *Multi-Level Iterative Techniques*, Ph.D. thesis, The Pennsylvania State University, University Park, PA, 1988.