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 bound the amount of additional refinement that is needed to achieve conformity. Numerical examples of the effectiveness of the algorithm are presented.

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 times. In this paper we present a simple algorithm for this purpose, and 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 tetrahedra shape cannot degenerate as the mesh is refined.

The basic refinement step in our algorithm is tetrahedral bisection as in Fig. 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.

![Fig. 1. Bisection of a single tetrahedron.](image-url)
To refine a given conforming mesh we first bisect those tetrahedra that have been selected for refinement. This will usually lead to a non-conforming mesh (a mesh in which neighboring elements don’t meet face-to-face). We then apply a recursive procedure to further refine until a conforming mesh is produced. Since it is not obvious 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 [12] and Ong [9] 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., e.g., Bank’s code PLTMG [1]), Bey [3] has studied the use of local octasection combined with bisection to obtain conformity. However, many different cases arise and the method is difficult to implement and to analyze. We believe that bisection, properly applied, is far preferable in three dimensions. A further advantage of bisection over octasection is that it allows meshes which are more smoothly graded, since element volume changes by a factor of only two in bisection, but eight in octasection.

A number of other authors have proposed bisection based algorithms for the refinement of tetrahedral meshes. In [2], Bänsch developed an algorithm for local tetrahedral mesh refinement and showed that the element shapes produced do not degenerate (although he did not show that the algorithm leads to a finite number of similarity classes). Our algorithm appears quite different from Bänsch’s, being simpler to state and implement, but it is essentially equivalent. Another paper which influenced our work, is that of Maubach [6]. 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 a close relation 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 hence deduce the same result for our algorithm. Liu and Joe [4] 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 so to ours. (Although we don’t prove it here, it can be shown that when properly initialized, and under some restrictions on the initial mesh, all three algorithms will generate the same tetrahedra.) 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 and coworkers [10,11]. They always use the longest edge of a tetrahedron as the refinement edge. Even in two dimensions, this approach does not lead to a finite number of similarity classes, but it is known that it cannot lead to element degeneration [10]. As far as we know, the question 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 as well. 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 \( \tau \) let \( V(\tau) \), \( E(\tau) \), and \( F(\tau) \) denote the set of its vertices, edges, and faces, respectively. For \( \varphi \in F(\tau) \), \( E(\varphi) \) denotes the edges contained in \( \varphi \). Once a particular edge has been specified as the refinement edge of \( \tau \), 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 non-refinement 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 \( \tau \) is a 4-tuple

\[
(V(\tau), r_\tau, (m_\varphi)_{\varphi \in F(\tau)}, f_\tau),
\]

where

- \( V(\tau) \) is a set of four non-coplanar points in \( \mathbb{R}^3 \), the vertices of \( \tau \);
- \( r_\tau \in E(\tau) \) is the refinement edge of \( \tau \);
- \( m_\varphi \in E(\varphi) \) is the marked edge of \( \varphi \), with \( m_\varphi = r_\tau \) if \( r_\tau \subset \varphi \);
- \( f_\tau \in \{0, 1\} \) is the flag, with \( f_\tau = 0 \) if \( \tau \) is non-planar.

Each marked non-refinement 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., Fig. 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.
- **Type A**, adjacent: the marked edges intersect the refinement edge, but are not coplanar.
- **Type O**, opposite: the marked edges of the non-refinement 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 non-refinement faces intersecting there.
- **Type M**, mixed: the marked edge of just one of the non-refinement faces intersects the refinement edge.

When a tetrahedron \( \tau \) is bisected to create children \( \tau_1 \) and \( \tau_2 \), a face \( \varphi \in F(\tau_i) \) is called an *inherited face* if \( \varphi \in F(\tau) \), a *cut face* if \( \varphi \nsubseteq \varphi' \) for some \( \varphi' \in F(\tau) \), 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., Fig. 3. We are now ready to state BisectTet.

**Algorithm** \( \{\tau_1, \tau_2\} = \text{BisectTet}(\tau) \)

**input:** A marked tetrahedron \( \tau \)

**output:** Marked tetrahedra \( \tau_1 \) and \( \tau_2 \)

1. Bisect \( \tau \) by joining the midpoint of its refinement edge to each of the two vertices not lying on the refinement edge. This defines \( V(\tau_i) \) for \( i = 1 \) and \( 2 \).
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 \( \nu \), cut faces marked \( c \), inherited faces marked \( i \), and new face marked \( n \).

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 Fig. 4. Note that the tetrahedra \( \tau_1 \) and \( \tau_2 \) output by \texttt{BisectTet} will be of the same type. Fig. 5 summarizes the relation between the input tetrahedron type and the output tetrahedra type. Note that types \( M \) and \( O \) are never output.

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 \texttt{BisectTet} to accomplish this.

Before stating the algorithm we fix some terminology. A mesh \( \mathcal{T} \) of a domain \( \Omega \) in \( \mathbb{R}^3 \) is a set of closed tetrahedra with disjoint interiors and union \( \overline{\Omega} \). 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 $\nu \in \mathcal{V}(\tau)$, we say that $\nu$ is a *hanging node* of $\tau$ 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, e.g., 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** $T' = \text{LocalRefine}(T, S)$

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

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

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

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

**Algorithm** $T' = \text{RefineToConformity}(T)$

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

The recursion in the algorithm $\text{RefineToConformity}$ 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 non-conforming; cf., Fig. 6). However, we shall prove that the recursion does terminate in the application of $\text{RefineToConformity}$ in algorithm $\text{LocalRefine}$, 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 $T_0$, and set $Q_0 = T_0$, and $Q_{k+1} = \text{BisectTets}(Q_k, Q_k)$, $k = 0, 1, \ldots$. Thus $Q_1$ consists of all children of tetrahedra in the initial mesh, $Q_2$ all grandchildren, etc. We assign generation $k$ to all tetrahedra in $Q_k$.

![Fig. 6. A non-conforming mesh without hanging nodes (the barycenter is not a vertex of the mesh).](image)
**Theorem 3.1.** Let $\mathcal{T}_0$ be a conformingly-marked mesh with no flagged tetrahedra. For $k = 0, 1, \ldots$, choose $S_k \subset \mathcal{T}_k$ arbitrarily, and set $\mathcal{T}_{k+1} = \text{LocalRefine}(\mathcal{T}_k, S_k)$. Then for each $k$, the application of $\text{RefineToConformity}$ from within $\text{LocalRefine}$ terminates producing a conformingly-marked mesh, and each tetrahedron in $\mathcal{T}_k$ has generation at most $3k$. Moreover, if the maximum generation of a tetrahedron in $\mathcal{T}_k$ 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 $\tau$. 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 $\tau$ 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 $\tau$ itself, and, referring to Fig. 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 non-marked edges.
- For each edge, its two children.

![Fig. 7](image_url)

**Fig. 7.** The meshes obtained via three applications of $\text{BisectTet}$ beginning with a tetrahedra of type $P_\mathcal{U}$, $A$, $O$, and $M$.

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

**Proof.** It is clear from Fig. 7 that $\mathcal{Q}_{3}^\tau$ is conforming. Moreover, the definition of $\text{BisectTet}$ 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 Fig. 5, we see that tetrahedra in $\mathcal{Q}_{3}^\tau$ are all either type $P_\mathcal{U}$ or type $A$, depending on whether $\tau$ is type $P_\mathcal{U}$ or not. This verifies the lemma in case $k = 1$.

If $\tau' \in \mathcal{Q}_{3}^\tau$, then the mesh of third generation descendants of $\tau'$ is, by the same argument, conformingly-marked and uniformly of type $P_\mathcal{U}$ or $A$. Because $\mathcal{Q}_{3}^\tau$ is itself 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 \( \tau' \) is a generation 3\( k \) descendant of an unflagged marked tetrahedron \( \tau \), then, by definition, all of the edges of \( \tau' \) have generation \( k \). The next lemma determines the generations of the edges of descendants of generation 3\( k + 1 \) and 3\( k + 2 \).

**Lemma 3.3.** Let \( \tau \) be an unflagged marked tetrahedron and \( \tau' \) a descendant of \( \tau \) of generation 3\( k + 1 \) or 3\( k + 2 \). Then the edges of \( \tau' \) all have generation \( k \) or \( k + 1 \).

**Proof.** The tetrahedron \( \tau' \) is either a child or a grandchild of an unflagged tetrahedron of generation 3\( k \). From Fig. 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 \( \tau' \) is an edge of a tetrahedron whose generation is either 3\( k \) or 3\( k + 3 \). \( \square \)

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

**Proposition 3.4.** Let \( T_0 \) be a conformingly-marked mesh with no flagged tetrahedra and let \( Q_k \) denote the mesh of all descendants of generation \( k \) of tetrahedra in \( T_0 \). Then the types of tetrahedra and the generation of edges of occurring in \( Q_k \) are as shown in Table 1. Moreover, the meshes \( Q_{3k} \) are conformingly-marked.

<table>
<thead>
<tr>
<th>Generations of tetrahedra</th>
<th>Types of tetrahedra</th>
<th>Generations of edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( P_u ) non-planar</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>( P_f ) ( P_u )</td>
<td>0,1</td>
</tr>
<tr>
<td>2</td>
<td>( A ) ( P_f )</td>
<td>0,1</td>
</tr>
<tr>
<td>3</td>
<td>( P_u ) ( A )</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>( P_f ) ( P_u )</td>
<td>1,2</td>
</tr>
<tr>
<td>5</td>
<td>( A ) ( P_f )</td>
<td>1,2</td>
</tr>
<tr>
<td>6</td>
<td>( P_u ) ( A )</td>
<td>2</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots ) ( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>3( k )</td>
<td>( P_u ) ( A )</td>
<td>( k )</td>
</tr>
<tr>
<td>3( k + 1 )</td>
<td>( P_f ) ( P_u )</td>
<td>( k, k + 1 )</td>
</tr>
<tr>
<td>3( k + 2 )</td>
<td>( A ) ( P_f )</td>
<td>( k, k + 1 )</td>
</tr>
<tr>
<td>3( (k + 1) )</td>
<td>( P_u ) ( A )</td>
<td>( k + 1 )</td>
</tr>
</tbody>
</table>

**Proof of Theorem 3.1.** The proof will proceed in several steps. We use the terminology descendant mesh of \( T_0 \) to refer to any mesh that can arise from \( T_0 \) by repeated application of BisectTet.

**Step 1.** If \( T \) is any descendant mesh of \( T_0 \) which has maximal tetrahedron generation 3\( k \), then the application of BisectTets in step 2 of RefineToConformity(\( T \)) returns a mesh which again has maximal tetrahedron generation 3\( k \). **Proof:** We refer to Table 1 to see that all the edges of tetrahedra in \( T \) are of most generation \( k \). Now if a tetrahedron in \( T \) 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 \( S \) defined in step 1 of \textit{RefineToConformity} consists of tetrahedra of generation less than \( 3k \). Consequently the mesh output from \textit{BisectTets} in step 2 of \textit{RefineToConformity}, again has maximal tetrahedron generation \( 3k \).

\textbf{Step 2.} If \( T \) is any descendant mesh of \( T_0 \) which has maximal tetrahedron generation \( 3k \), then every mesh constructed in the recursive application of the algorithm \textit{RefineToConformity}(\( T \)) has maximal tetrahedron generation \( 3k \) and, moreover, the algorithm terminates. \textit{Proof:} Indeed new tetrahedra are only introduced by the application of \textit{BisectTets} in step 2 of \textit{RefineToConformity}, so the generation bound follows from the previous step. Since there are only finitely many tetrahedra of generation \( 3k \), the algorithm must terminate.

\textbf{Step 3.} Each tetrahedron in \( T_k \) has bisection level at most \( 3k \). \textit{Proof:} By induction on \( k \), the case \( k = 0 \) being obvious. By the inductive hypothesis, \( T_{k-1} \) consists of tetrahedra of generation at most \( 3k-3 \). Hence the mesh \( T \) output from \textit{BisectTets} in step 1 of \textit{LocalFracnfity}(\( T_{k-1}, S_{k-1} \)) has maximum tetrahedron generation \( 3k - 2 \leq 3k \), and the result follows from the preceding claim.

\textbf{Step 4.} The output mesh is conformingly-marked. \textit{Proof:} This follows easily from the fact that the output mesh is a descendant of a conformingly-marked mesh and has no hanging nodes.

\textbf{Step 5.} Finally, the last sentence of the theorem follows directly from step 2. \( \square \)

\section{Similarity classes.} In \cite{6} Maubach gave an algorithm, which we refer to as \textit{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 \textit{BisectTet}, when \textit{BisectTet} is restricted to tetrahedra of types \( A \) and \( P \) as input. Define a \textit{tagged simplex} in \( \mathbb{R}^n \) as an ordered pair, \( t = ((x_0, x_1, \ldots, x_n), d) \), where \((x_0, x_1, \ldots, x_n)\) is an ordered \((n + 1)\)-tuple of points in general position in \( \mathbb{R}^n \) (the vertices of the simplex), and \( d \in \{1, 2, \ldots, n\} \), \((\overrightarrow{x_0x_d} \) is the refinement edge of \( t \)). With this terminology, Maubach's algorithm may be stated as follows.

\begin{algorithm}
\textbf{Algorithm} \{\( t^{(1)}, t^{(2)} \)\} = \textit{BisectSimplex}(\( t \))
\begin{enumerate}
\item \textit{input:} tagged \( n \)-simplex \( t \)
\item \textit{output:} tagged \( n \)-simplices \( t^{(1)} \) and \( t^{(2)} \).
\end{enumerate}
\begin{enumerate}
\item set \( d' = \begin{cases} 
  d - 1, & d > 1 \\
  n, & d = 1
\end{cases} \)
\item create the new vertex \( z = \frac{1}{2} (x_0 + x_d) \)
\item set \( t^{(1)} = \left( (x_0, x_1, \ldots, x_{d-1}, z, x_{d+1}, \ldots, x_n), d' \right) \)
\item set \( t^{(2)} = \left( (x_1, x_2, \ldots, x_2, z, x_{d+1}, \ldots, x_n), d' \right) \)
\end{enumerate}
\end{algorithm}

We now define a correspondence between tagged simplices in \( \mathbb{R}^3 \) and marked tetrahedra of types \( P \) and \( A \), and show that repeated application of either \textit{BisectTet} or \textit{BisectSimplex} to the same geometric tetrahedron yields the same sequence of descendant tetrahedra. Let \( \mathcal{J}_T \) be the set of all tagged \( 3 \)-simplices and \( \mathcal{J}_M \) the set of all marked tetrahedra; also, let \( \mathcal{J}_A \subset \mathcal{J}_M \) denote the set of marked tetrahedra of type \( A \) or \( P \). Define a mapping \( \mathcal{F} : \mathcal{J}_T \rightarrow \mathcal{J}_M \) as follows:
For $t = ((x_0, x_1, x_2, x_3), d) \in \mathcal{J}_T$, set $\mathcal{F}(t) = (\mathcal{V}(\tau), r_\tau, m_\varphi, f_\tau)$ with $\mathcal{V}(\tau) = \{x_0, x_1, x_2, x_3\}$ and

$$
\begin{align*}
r_\tau = x_0x_1, & \quad m_\varphi = \begin{cases} 
x_0x_2, & \text{if } \varphi = x_0x_2, \\
x_1x_3, & \text{if } \varphi = x_1x_3,
\end{cases} & & f_\tau = 1, \text{ if } d = 1; \\
r_\tau = x_0x_2, & \quad m_\varphi = \begin{cases} 
x_0x_1, & \text{if } \varphi = x_0x_1, \\
x_1x_3, & \text{if } \varphi = x_1x_3,
\end{cases} & & f_\tau = 0, \text{ if } d = 2; \\
r_\tau = x_0x_3, & \quad m_\varphi = \begin{cases} 
x_0x_2, & \text{if } \varphi = x_0x_2, \\
x_1x_3, & \text{if } \varphi = x_1x_3,
\end{cases} & & f_\tau = 0, \text{ if } d = 3.
\end{align*}
$$

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

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

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

If $\tau$ has type $A$, we assume, without loss of generality, that its refinement and non-refinement edges are $\overrightarrow{v_0v_1}$ and $\{\overrightarrow{v_0v_2}, \overrightarrow{v_1v_3}\}$ respectively. To get the first tagged simplex, set $d = 3$ and choose $x_0 = v_0$ and $x_3 = v_1$ (the end points of the refinement edge). Set $x_1 = v_3$ (the second endpoint of the marked non-refinement edge containing $x_3$), 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 $((v_0, v_3, v_2, v_1), 3)$ and $((v_1, v_2, v_3, v_0), 3)$. It is straightforward to check that $\mathcal{F}$ maps each of these tagged simplices to $\tau$ and that these are the only preimages of $\tau$ under $\mathcal{F}$.

The argument is similar in the cases of $\tau$ of type $P_u$ or $P_f$. In the $P_u$ case, we take the numbering so that the refinement and marked non-refinement edges are $\overrightarrow{v_0v_1}$ and $\{\overrightarrow{v_0v_2}, \overrightarrow{v_1v_3}\}$, respectively, and find the two preimages $((v_0, v_2, v_1, v_3), 2)$ and $((v_1, v_2, v_0, v_3), 2)$. In the $P_f$ case with refinement and marked non-refinement edges $\overrightarrow{v_0v_1}$ and $\{\overrightarrow{v_0v_2}, \overrightarrow{v_1v_3}\}$, the preimages are $((v_0, v_1, v_3, v_2), 1)$ and $((v_1, v_0, v_3, v_2), 1)$. \(\Box\)

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

**Theorem 4.2.** The following diagram commutes.

$$
\begin{array}{ccc}
\mathcal{J}_T & \xrightarrow{\mathcal{F}} & \mathcal{J}_M \\
\text{BisectSimplex} \downarrow & & \downarrow \text{BisectTet} \\
\mathcal{J}_T \times \mathcal{J}_T & \xrightarrow{\mathcal{F} \times \mathcal{F}} & \mathcal{J}_M \times \mathcal{J}_M
\end{array}
$$

**Proof.** Suppose $t = ((x_0, x_1, x_2, x_3), d) \in \mathcal{J}_T$, and $\{t^{(1)}, t^{(2)}\} \in \mathcal{J}_T \times \mathcal{J}_T$ is produced by BisectSimplex$(t)$. Then $t^{(1)} = ((x_0, x_1, x_2, z), 2)$ with $z = (x_0 + x_3)/2$, and so $\mathcal{F}(t^{(1)})$ yields the marked tetrahedron $\{x_0, x_1, x_2, z\}, \overrightarrow{x_0x_2}, \overrightarrow{x_0x_1}, \overrightarrow{x_1x_2}, 0$ (here, only the markings for the non-refinement faces of the tetrahedron are specified in $m_\varphi$ with the convention that the given edges are marked for the non-refinement face containing them). On the other hand, $\mathcal{F}(t) = ((x_0, x_1, x_2, x_3), \overrightarrow{x_0x_2}, \overrightarrow{x_0x_1}, \overrightarrow{x_1x_2}, 0)$ which is a tetrahedron of type $A$, and one of the marked tetrahedra produced by applying
BisectTet to this tetrahedron is $\mathcal{F}(t(1))$. A similar verification is easily carried out for the other cases. \hfill \square

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 $t \in \mathcal{T}$ are exactly the same as those obtained by applying BisectTet $m$ times to $\mathcal{F}(t)$.

Our next goal is a bound on the number of similarity classes produced by repeated application of algorithm BisectSimplex. This will be based on the following technical result from [6]. (Theorem 4.1 of [6] states this result in less generality, but the same proof applies to the statement given here.)

**Lemma 4.4.** Let $t'$ be a descendent of the tagged $n$-simplex

$$t = \left( (0, e_1, e_1 + e_2, \ldots, e_1 + e_2 + \ldots + e_n), d \right)$$

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

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

where

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

An upper bound for the number of similarity classes produced by the repeated application of BisectSimplex follows immediately from this result. As there are $n!$ possibilities for the permutations $\pi$, $2^n$ possibilities for the reflections $R$, and exactly $n$ different vectors $\alpha_i$, the bound is $mn!2^n$. 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 $mn!2^{n-1}$. However, this bound is not sharp. In the simple case $n = 2$, only 4 similarity classes of triangles arise from repeated bisection, but the bound is 8. We now show how to reduce the bound by a further factor of 2 to $mn!2^{n-2}$. For $n = 2$ this bound of 4 similarity classes is obviously sharp. For $n = 3$, the bound is 36, which improves a bound of 168 due to Liu and Joe [3] for an algorithm which can be seen to be equivalent. By direct computation on a particular tetrahedron we have verified that the bound of 36 is sharp. (For example, when $\mathcal{V}(\tau) = \{v_1, v_2, v_3, v_4\}$ where $v_1 = (0, 0, 0)$, $v_2 = (23, 0, 0), v_3 = (7, 0, 11)$, and $v_4 = (17, 5, 33)$, and $\overline{v_1v_2}$ is the refinement edge, the upper bound of 36 is attained at the seventh generation.) Maubach [7] has announced a proof that the bound of $mn!2^{n-2}$ is sharp for all $n$. 
Theorem 4.5. The number of similarity classes of $n$-simplices produced by the repeated application of Bisect-Simplex is bounded by $nn!2^{n-2}$.

Proof. It suffices to show 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

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

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

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

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

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

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

Thus, $\mathbf{t}'$ is related to $\hat{\mathbf{t}}$ via the translation $\Phi$. The theorem now follows from Lemma 4.4 and the discussion following it. \qed

In view of Proposition 4.1 and Theorem 4.2, the above result applies to bisection of marked tetrahedra of types $A$ and $P$. Since one application of BisectTet to a tetrahedron of type $O$ or $M$ produce 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 $T_0$ is chosen and the meshes $T_k$ are generated using $T_k = \text{LocalRefine}(T_{k-1}, S_{k-1})$ as in section 3 with different criteria being used to determine the sets $S_k$ in each example.

Example 1. The first example is adapted from Maubach [6]. Let $T_0$ 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

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

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

![Fig. 8. The mesh $T_{16}$ 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 $y = 1/2$.](image)

The algorithm LocalRefine has been incorporated into a code for solving second order elliptic boundary value problems (cf., [8]). Error estimators are used to determine the sets $S_k$, in the code, leading to refinement in regions where the gradient of the solution changes rapidly. The next two examples are test problems computed using this code. The exact solution to the problem is known in each case. For these examples, $T_0$ is a subdivision of $[0, 1]^3$ having 96 tetrahedra.

Example 2. Fig. 9 shows the mesh $T_0$ obtained while solving a problem with exact solution

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

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 Fig. 9.

Example 3. Fig. 10 shows the mesh $T_0$ obtained while solving a problem with exact solution $u_{ex} = e^{3x+3y+z}$. It is easy to verify that on any plane parallel to the $xy$ plane both $|u_{ex}|$ and $|\text{grad} u_{ex}|$ behave like $e^{3x+3y}$, while for planes parallel to the $xz$ or $yz$ planes their behavior is determined by $e^{3x+z}$ and $e^{3y+z}$ respectively. This qualitative difference in behavior is evident in the adapted mesh $T_8$ (cf., Fig. 10).
Fig. 9. Cut along the plane $x = 1/4$ of $\mathcal{T}_0$ of example 2 (there are 11,418 tetrahedra in the mesh).

Fig. 10. The planes $x = 1$, $y = 1$, and $z = 1$ for $\mathcal{T}_8$ of example 3 (there are 19,448 tetrahedra in the mesh).

References

University, 1996.

**Department of Mathematics, Penn State University, University Park, PA 16802**
*E-mail address: dna@math.psu.edu*

**Department of Mathematics, Rutgers University, New Brunswick, NJ 08903**
*E-mail address: arup@math.rutgers.edu*

**Department of Mathematics, Swiss Federal Institute of Technology, CH-1015 Lausanne**
*Current address: ELCA Informatique SA, CH-1000 Lausanne 13*
*E-mail address: Luc.Pouly@elca.ch*