Local bisection refinement for n-simplicial grids generated by reflection

SIAM J. SCI. COMPUT.
Vol. 16, No. 1, pp. 210–227, January 1995

014

LOCAL BISECTION REFINEMENT FOR N-SIMPLICIAL GRIDS GENERATED BY REFLECTION*

JOSEPH M. MAUBACH†

Abstract. A simple local bisection refinement algorithm for the adaptive refinement of n-simplicial grids is presented. The algorithm requires that the vertices of each simplex be ordered in a special way relative to those in neighboring simplices. It is proven that certain regular simplicial grids on $[0, 1]^n$ have this property, and the more general grids to which this method is applicable are discussed. The edges to be bisected are determined by an ordering of the simplex vertices, without local or global computation or communication. Further, the number of congruency classes in a locally refined grid turns out to be bounded above by n, independent of the level of refinement. Simplicial grids of higher dimension are frequently used to approximate solution manifolds of parametrized equations, for instance, as in [W. C. Rheinboldt, *Numer. Math.*, 53 (1988), pp. 165–180] and [E. Allgower and K. Georg, *Utilitas Math.*, 16 (1979), pp. 123–129]. They are also used for the determination of fixed points of functions from $\mathbf{R}^n$ to $\mathbf{R}^n$, as described in [M. J. Todd, *Lecture Notes in Economic and Mathematical Systems*, 124, Springer-Verlag, Berlin, 1976]. In two and three dimensions, such grids of triangles, respectively, tetrahedrons, are used for the computation of finite element solutions of partial differential equations, for example, as in [O. Axelsson and V. A. Barker, *Finite Element Solution of Boundary Value Problems*, Academic Press, Orlando, 1984], [R. E. Bank and B. D. Welfert, *SIAM J. Numer. Anal.*, 28 (1991), pp. 591–623], [W. F. Mitchell, *SIAM J. Sci. Statist. Comput.*, 13 (1992), pp. 146–147], and [M. C. Rivara, *J. Comput. Appl. Math.*, 36 (1991), pp. 79–89]. The new method is applicable to any triangular grid and may possibly be applied to many tetrahedral grids using additional closure refinement to avoid incompatibilities.

Key words. grid generation, grid refinement

AMS subject classification. 65M50

1. Introduction.

Local grid refinement of computational grids has succesfully been applied in two dimensions in order to locally improve the approximate solution of systems of partial differential equations. However, many interesting problems of a physical nature are formulated in three or more dimensions, increasing the need for a local refinement method in more dimensions. This paper presents such a local refinement method of the bisection type, which is called n-dimensional since it is applicable to grids of n-simplices, independent of the dimension n. The examination of the new method was initiated with a three-dimensional bisection algorithm of the author published in [21], and stimulated by the work by Bey in [11].

Two- and three-dimensional triangulations are a subject of thorough investigation. Many different grid generation, refinement, and improvement techniques in two and three dimensions are now available. An extensive overview of such techniques can be found in the bibliographies of Bern and Eppstein [8], or Baker [5]. Among the available generation methods in two, three, or more dimensions, there are methods based on Voronoi's tessellations, as in George and Hecht [15]; methods based on Delaunay triangulations, as in Bern, Eppstein and Yao [9], Riedinger et al. [25], and Schröder and Shephard [27]; and methods based on fractal concepts, as in Bova and Carey [12]. Mesh improvement techniques include techniques changing the grid topology, as in Frey and Field [14], and vertex moving techniques keeping the topology, as in Arney and Flaherty [2]. Some techniques guarantee the resulting grid to have certain properties, see, for instance Shephard and Georges [28] or Bern and Eppstein [10]. Mesh refinement techniques will be discussed below.

The method to be introduced is a grid refinement method, applicable to simplicial grids of any dimension. It will be compared with both triangular and tetrahedron refinement methods. First, consider the two-dimensional case. Most frequently, a triangle is refined by dividing it into four congruent descendants, as in [6], [4], or [11] applied to two dimensions. This

*Received by the editors April 6, 1992; accepted for publication (in revised form) December 22, 1993.

†Mathematics Department, University of Pittsburgh, Thackary Hall 301, Pittsburgh, Pennsylvania, 15260 (joseph@garfield.math.pitt.edu).

method has the advantage that all created descendants are congruent to the parent, but in order to prevent the coarse grid from being refined globally (which is necessary to keep the grid's triangles compatible), the method has to be used in combination with additional bisection refinement, called closure refinement.

Another approach, using bisection refinement, is that of Rivara [26], where for each triangle it is always the longest edge that is bisected. In this case, the resulting number of congruency classes may be unlimited, but one can prove that the angles of the triangles in a resulting locally refined grid, are uniformly bounded away from 0 and π, which is needed for many finite element applications. Also here, additional closure refinement is necessary to ensure that refined grids are compatible.

The newest vertex bisection method is introduced by Mitchell in [22]. The main advantage of his method is that it can be applied to any coarse triangular grid, and such that the number of congruency classes of all triangles in the locally refined grid will be at most four times the number of coarse grid triangles. Mitchell's bisection method is relatively simple—the triangle's edge to be bisected is determined without any computation.

Like the bisection method of Mitchell, the application of the local bisection refinement method presented in this paper leads to a refined grid of simplices, such that the number of congruency classes is finite, independent of the level of refinement. Further, the edge to be bisected is determined without computation or global communication, i.e., without using information concerning its own or other simplices geometry. This makes the method simple to implement, suited for finite element methods, and highly suitable for parallel processing.

Refinement methods for grids of tetrahedrons can roughly be divided into the group simultaneously creating eight descendants, as in Bey [11] and Ong [23]; the group creating two descendants by bisection, as in Bänsch [7], Kossaczky [19], Maubach [21], and Rivara [26]; and the method to be presented in this paper. Both the new method and that of Bey generalize to n-dimensional grids of simplices. The division of a tetrahedron into eight descendants is such that only four will be congruent to the parent, and in order to prevent grids from getting uniformly refined, one has to use additional closure refinements, as in [6] and [11]. However, this refinement, in combination with the proper closure refinement, can be applied to any coarse tetrahedral grid to yield compatible locally refined grids. Allowing for additional closure refinement, it is likely that the new method can also be used for many coarse tetrahedral grids. The use of proper closures will be a topic for future research.

The remainder of this introduction will list more specific properties of the new method and compare them to properties of some of the mentioned existing techniques.

In contrast with the quadrant method for the refinement of grids of rectangles, or the octant method for the refinement of a n-cube grid (see Lohner [20]), the presented local bisection method does not create incompatibilities. Every face of an n-simplex will be fully shared by at most one other n-simpex. Because every simplex is compatible without exception, algorithms can be implemented in a simple manner. Further, because the new method is a bisection method one can use binary tree data structures independent on the dimension. Binary tree data structures have been well investigated throughout the last decades, many optimal or near-optimal tree traversion and tree balancing algorithms exist, as is shown in Knuth [18].

Grids of n-simplices for large n have long been used in combinatorial fixed point theory (see Sperner [29], Freudentahl [13], and Todd [30]), giving an overview of existing combinatorial fixed point methods. The method presented in this paper can theoretically generate all of the grids to be found in [30]. The number of unknowns n in fixed point theory can be compared with the degrees of freedom resulting from a finite element discretization of a partial differential equation, and can be easily in the order of 10,000. As $n!$ simplices cover an n-cube, practical generation is out of scope and in order to locate fixed points, fortunately

not necessary. In fixed point theory, n-cube refinement is rarely used because it creates many more new vertices to reach a certain level of refinement than bisection refinement does ($3^n - 2^n$ versus n vertices), and because the total amount of work to locate a fixed point is proportionally related to the number of new vertices created.

The computational effort to determine the location of a grid element containing a certain point is the same for the octant technique and the new bisection algorithm (independent on the dimension). This can be seen as follows. Consider the three-dimensional case and assume that the point is situated inside one of the grid's elements. Using an octant method, the refinement of this element (a cube) creates eight descendants via intersection of the element with three axis-parallel planes. Therefore it takes three searches to determine in which of the eight descendants the point is located. In contrast, using the bisection method, it can be shown that after each n refinements the resulting descendants are exactly half the size of the initially refined element (a tetrahedron). Each bisection creates two descendants inside the parent, seperated by a plane whence it takes three searches to determine in which of the eight descendants the point is located. For this case, the bisection method presented in this paper is as efficient in the storage and location of information as the octtree approach is in [20], especially since every grid of n-cubes can be covered with n-simplices. Potential applications of the new method to neural networks are a topic of future investigation.

Now, consider the applicability of the bisection refinement method. Because the method is applicable only to simplicial grids, quadrant and octant techniques will not be part of the following discussion. This method generalizes the two-bisection method by Mitchell in [22] to n dimensions in the sense that that both methods make use of a special ordering of the vertices of the elements in the grid (triangles in two dimensions). The n-dimensional method turns out to be different from but equivalent to Mitchell's method in two dimensions. Because Mitchell proves that his method can be applied to every coarse grid in two dimensions, the new method automatically inherits this property. Unfortunately, it is not yet clear whether the new method can be applied to any coarse simplicial grid in n dimensions. Mitchell's proof for two dimensions uses graph theoretical results of which no multi-dimensional extensions exist, to the knowledge of the author. It is unlikely that every simplicial grid in n dimensions can have the vertices of each of its simplices ordered properly for the new bisection method. However, for the case where a simplicial grid G is generated by reflections as shown in [30] in an $k_1 \times \cdots \times k_n$ grid of n-cubes covering $[a_1, b_1] \times \cdots \times [a_n, b_n]$, this paper proves that the vertices are properly ordered and that the refinement method can be applied. The proofs provided admit moving of the grids vertices (see [2]) because this does not change the vertex ordering within a simplex. The new method is also applicable to grids which are part of the locally refined grid G. Further, any simplicial grid that is the image of a nonsingular mapping applied to (part of a possibly locally refined) grid G is suited for refinement with the new method because the mapping will not change the proper order of the vertices relative to each other. This means that the method can also be used for grids which have to fit curved boundaries, especially because the new method allows one to move the vertices of the grid. Finally, one can also cover regions with component grids of type G, as is done for n-cube type grids by Henshaw and Chesshire in [17].

The remainder of this paper is organized as follows. First, in §2, the construction of the coarse grid of n-simplices covering $[0, 1]^n$ is shortly introduced, mainly relying on results obtained in [1]. Thereafter, §3 introduces the bisection step. Section 4 exploits the bisection step in order to demonstrate that the number of congruency classes is bounded above by n, independent on the level of refinement, and §5 employs the bisection step in the recursive local bisection grid refinement algorithm. It is shown that the depth of the recursion is finite, and that grid incompatibilities can easily be avoided. In §6, the grid refinement algorithm is applied

for the generation of some locally refined grids in two and three dimensions, indicating that the bisection type refinement is highly local in nature and not spreading. Finally, §7 provides some conclusions, and is followed by a list of references.

2. The coarse grid. This section presents the coarse simplicial grid on $[0, 1]^n$, which, with the use of reflections, can be used to create a coarse simplicial grid covering an $k_1 \times \cdots \times k_n$ grid of cubes covering $[a_1, b_1] \times \cdots \times [a_n, b_n]$. The permutations used to this end are introduced and some elementary definitions and examples are given. Thereafter, a definition of congruency is provided, and at the end of this section it is shown that the simplices of the coarse grid are congruent to each other. In the upcoming sections the prefix "n-" will often be omitted, in order to simplify the notation.

The simplicial grid on $[0, 1]^n$ will be generated with the use of n-permutations. Each n-permutation denoted by $\pi = (i_1\, i_2\, \ldots\, i_n)$ can be uniquely associated with a linear functional

$$A_\pi(x_1, \ldots, x_n) = (x_{i_1}, \ldots, x_{i_n}),$$

which represents a coordinate permutation. As an example, consider $\pi = (2\ 1\ 3)$, which can be represented by the matrix

$$A_\pi = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

In the upcoming sections, $\pi(\boldsymbol{x})$ is meant to be an abbreviation of $A_\pi \boldsymbol{x}$.

In order to enable to examine congruency classes, two simplices are said to be congruent if the first one is the image of the second one under a congruency mapping, which is of the type

$$\boldsymbol{y} = \alpha(A\boldsymbol{x} + \boldsymbol{b}),$$

where α is a positive real number, A an orthonormal matrix, and b an n-vector. Note that mappings of the type A_π are congruency mappings. Actually, mappings of the type A_π are a concatenation of simple coordinate reflections and rotations. As an example, (2 1 3) has a corresponding linear functional with matrix representation

$$A_{(2\ 1\ 3)} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \circ \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

a combination of a reflection in the x_2-axis and a rotation in the x_1x_2-plane.

For the covering of $[0, 1]^n$ the following so-called reference simplex (different from that usually found in finite element methods) is needed. This reference simplex is identified with an ordered row of $n + 1$ vertices $\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \ldots, \hat{\boldsymbol{x}}_n$, where $\hat{\boldsymbol{x}}_0 = (0, \ldots, 0)$ and

$$\hat{\boldsymbol{x}}_i = \sum_{j=1}^{i} \boldsymbol{e}_i,$$

with $\boldsymbol{e}_i$ being the ith unit vector. Using the reference simplex, for a permutation π, the simplex T_π is identified with the ordered row of vertices

$$\pi(\hat{\boldsymbol{x}}_0), \pi(\hat{\boldsymbol{x}}_1), \pi(\hat{\boldsymbol{x}}_2), \ldots, \pi(\hat{\boldsymbol{x}}_n).$$

In this paper, a simplex is always identified with its ordered row of vertices. This is important, since the bisection step proposed in §3 exploits the ordering of the vertices.

In [1] it is shown that the collection of simplices $\{T_\pi : \pi \text{ an } n\text{-permutation}\}$ covers the unit cube such that all simplices are compatible, i.e., every face of every simplex is shared with at most one other simplex. Now consider the algorithm below, for the generation of the coarse simplicial grid covering $[0, 1]^n$.

```
Construct coarse grid:
BEGIN
 FOR every n-permutation π
 DO
  Create T_π: π(x̂_0), π(x̂_1), π(x̂_2), ..., π(x̂_n);
 END;
END.
```

Note that the vertices $\underline{0} = (0, \ldots, 0)$ and $\underline{1} = (1, \ldots, 1)$ are invariant under every permutation π, whence they are shared by every coarse grid simplex. As an example of a coarse grid, consider the six tetrahedrons covering $[0, 1]^3$,

$$
\begin{aligned}
T_{(1\,2\,3)} &: (0,0,0), (1,0,0), (1,1,0), (1,1,1)\\
T_{(1\,3\,2)} &: (0,0,0), (1,0,0), (1,0,1), (1,1,1)\\
T_{(2\,1\,3)} &: (0,0,0), (0,1,0), (1,1,0), (1,1,1)\\
T_{(3\,1\,2)} &: (0,0,0), (0,1,0), (0,1,1), (1,1,1)\\
T_{(3\,2\,1)} &: (0,0,0), (0,0,1), (0,1,1), (1,1,1)\\
T_{(2\,3\,1)} &: (0,0,0), (0,0,1), (1,0,1), (1,1,1),
\end{aligned}
$$

where each tetrahedron is identified with the row of its vertices. Here, the first tetrahedron is obtained by applying the identity permutation $\pi = (1\ 2\ 3)$ to the reference tetrahedron, and the second one is the image of the reference tetrahedron under the application of the coordinate permutation $\pi = (1\ 3\ 2)$.

Now, the number of congruency classes of simplices of the coarse grid is easy to determine, as the following lemma shows.

LEMMA 2.1. *The number of the coarse n-grid congruency classes is* 1.

Proof. All coordinate permutations are congruency mappings. □

Finally, using the above coarse grid of simplices on a single cube, a coarse grid of $k_1 \times \cdots \times k_n$ cubes (filled with simplices) covering $[a_1, b_1] \times \cdots \times [a_n, b_n]$ is constructed by reflecting the above coarse grid cube across its faces in all coordinate directions, as in [1].

3. The bisection step. This section introduces the n-dimensional bisection step for n-simplices, and examines the descendants obtained with repeated application of this bisection step. The number of congruency classes obtained by the proposed bisection step is determined in §4.

The bisection of an n-simplex presented below only involves the ordering of the vertices of this simplex. Initially, all coarse grid n-simplices T are said to be of level $l(T) = 0$. If a simplex T is bisected, the two created simplices are called its descendants, and the ordering of their vertices is defined by the following bisection step:

```
Bisect (simplex):
BEGIN
 Let k := n - l(simplex) mod n;
 Get simplex vertices: x_0, x_1, ..., x_{n-1}, x_n;
 Create the new vertex: z := ½{x_0 + x_k};
 Create descendant_0: x_0, x_1, ..., x_{k-1}, z, x_{k+1}, ..., x_n;
```

Create descendant$_1$: $x_1, x_2, \ldots, x_k, z, x_{k+1}, \ldots, x_n$;
Let l(descendant$_0$) := l(simplex) + 1;
Let l(descendant$_1$) := l(simplex) + 1;
END.

The first thing to be observed is that this algorithm actually represents a bisection refinement step. This is caused by the fact that edge x_0x_k is bisected, and by the fact that the descendants each get one of the two vertices x_0 and x_k. Since the parent is the union of its descendants and the coarse grid covers $[0, 1]^n$, it can be concluded that any grid resulting from the repeated application of this bisection step will properly cover $[0, 1]^n$.

Second, the above bisection step may lead to grid incompatibilities, because it only focusses on one simplex, neglecting its neighboring simplices. In order to avoid these incompatibilities, the bisection step is incorporated in the local bisection refinement algorithm presented in §5.

Further, in order to reduce computer storage, the parent simplex can be deleted after its descendants are created, but for some applications it is advisable to keep the whole binary tree, resulting from the bisection refinement, in the computer memory. One of these applications would be the determination of a simplex containing a prescribed point x.

Applied to the case of two dimensions, this bisection step is slightly different from that proposed in Mitchell [22]. Mitchell's newest vertex method for the bisection of a triangle can be defined as follows. If a triangle has vertices ordered x_0, x_1, x_2, then x_1x_2 will be bisected, and the new vertex $z = \frac{1}{2}\{x_1 + x_2\}$ is created. The first vertex of both descendants will be the new vertex z, the second one will be x_0, and the last one will be x_1, respectively x_2. The difference between the newest vertex method and the above bisection step is demonstrated by Figs. 1 and 2, which show the different order of the vertices of a triangle and its two descendants. Note, however, that the methods are equivalent in two dimensions, and that the new method therefore can be applied to every two-dimensional triangular grid. Using the above bisection method, the "newest vertex" of a simplex of level $l > 0$ is given by $n - (l - 1) \mod n$; with Mitchell's method the "newest vertex" is always at position 0.

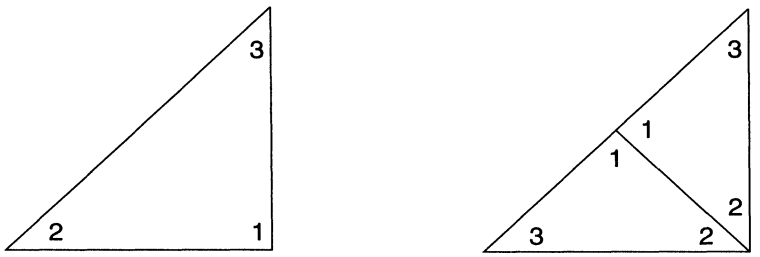

FIG. 1. *Newest vertex bisection refinement in two dimensions.*

As an example, consider the bisection step applied to one of the initial coarse grid tetrahedrons introduced earlier:

$$T_{(1\ 3\ 2)}\colon\ (0, 0, 0), (1, 0, 0), (1, 0, 1), (1, 1, 1)\,.$$

The bisection of this tetrahedron creates two descendants,

$$(0, 0, 0), (1, 0, 0), (1, 0, 1), (\tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}) \text{ and } (1, 0, 0), (1, 0, 1), (1, 1, 1), (\tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2})\,.$$

Again applying the bisection step, the second descendant T_1 is bisected into

$$(1, 0, 0), (1, 0, 1), (1, \tfrac{1}{2}, \tfrac{1}{2}), (\tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}) \text{ and } (1, 0, 1), (1, 1, 1), (1, \tfrac{1}{2}, \tfrac{1}{2}), (\tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2})\,,$$

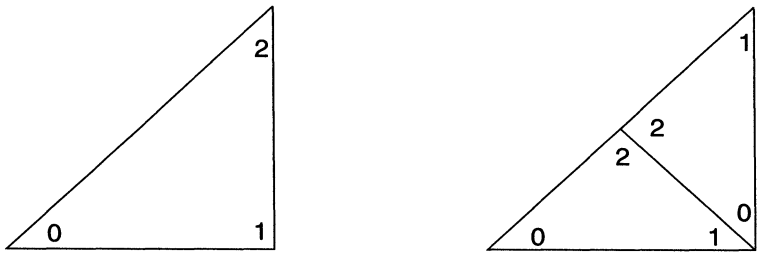

FIG. 2. *n-dimensional bisection refinement in two dimensions.*

and its first descendant T_2 will be bisected into the simplices

$$(1,0,0),(1,0,\tfrac{1}{2}),(1,\tfrac{1}{2},\tfrac{1}{2}),(\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2}) \text{ and } (1,0,1),(1,0,\tfrac{1}{2}),(1,\tfrac{1}{2},\tfrac{1}{2}),(\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2})$$

of level 3. For the sake of simplicity, denoting

$$\begin{aligned} T_1 &: 1,0,0),(1,0,1),(1,1,1),(\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2}) \\ T_2 &: (1,0,0),(1,0,1),(1,\tfrac{1}{2},\tfrac{1}{2}),(\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2}) \\ T_3 &: (1,0,1),(1,0,\tfrac{1}{2}),(1,\tfrac{1}{2},\tfrac{1}{2}),(\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2}), \end{aligned}$$

one finds that $T_{(1\ 3\ 2)}$ is the grandparent of T_3, T_2, and T_1, and that the level of T_i is equal to i for all $i = 1, 2, 3$. Also, observe that the congruency mapping

$$y = 2(x - [1,0,1]^t)$$

maps T_3 onto $(0,0,0),(0,0,-1),(0,1,-1),(-1,1,-1)$. Hence, $y = 2R(x - [1,0,1]^t)$ with $R = \mathrm{Diag}(-1,1,-1)$ maps T_3 onto the coarse grid tetrahedron $T_{(3\ 2\ 1)}$, which at its turn is congruent with $T_{(1\ 3\ 2)}$. This indicates that in three dimensions it takes precisely three bisection steps to create a tetrahedron which is a factor 2 smaller in all directions than its grandparent, and that tetrahedrons of level 3 are congruent to their grandparent of level 0, modulus reflections.

It can be observed that the created descendants of level 1 and 2 are not congruent to one of the coarse grid simplices. However, in §4 it is shown that descendants of the same level (modulus n) are mutually congruent. This will limit the possible number of congruency classes considerably, since there clearly are 2^n descendants of level n to each of the $n!$ coarse grid simplices.

4. The number of congruency classes. This section examines the number of congruency classes created by the bisection step defined in §3. With the use of Theorem 4.1, it is shown that the amount of created congruency classes is precisely equal to the dimension n, independent of the level of refinement k. Theorem 4.1 provides more information than that needed to count the number of congruency classes; it describes the geometry of a simplex created by repeatedly applied bisection.

THEOREM 4.1. *Let T be an n-simplex created by the repeated application of the bisection step in §3 to one of the coarse grid simplices defined in §2. Assume that T is of level $0 \le l \le n$ with ordered vertices*

$$x_0, x_1, x_2, \ldots, x_n .$$

Further, define $y_i = x_i - x_0$ *for all* $i \in \{1, \ldots, n\}$. *Then there exists an* n*-permutation* π *and a reflection matrix* $R = \mathrm{Diag}(\pm 1, \ldots, \pm 1)$ *such that*

$$y_i = \alpha_i R \sum_{j=1}^{i} \pi(e_j),$$

for all $i \in \{1, \ldots, n\}$, *where*

$$\begin{cases} \alpha_i = 2^{-[l/n]} & \text{if } i \in \{1, \ldots, n - (l \mod n)\}, \\ \alpha_i = 2^{-[l/n]-1} & \text{if } i \in \{n - (l \mod n) + 1, \ldots, n\} \end{cases}$$

only depends on the level of the simplex.

Proof. The induction hypothesis is obviously satisfied for coarse grid simplices, taking $R = I_n$ the identity matrix. Now, consider a level $0 \le l < n$ simplex with vertices $x_0, x_1, x_2, \ldots, x_n$, and assume that a permutation π and diagonal matrix R exist such that the induction hypothesis holds. For each of its two level $l+1$ descendants it will be shown that there exists a permutation π' and a diagonal matrix R' such that the induction hypothesis holds. Note that $2^{-[l/n]} = 1$, and that $l \mod n = l$.

First consider the first descendant, with vertices $x'_0, x'_1, x'_2, \ldots, x'_n$. This satisfies the induction hypothesis, taking the associated permutation $\pi' = \pi$ and $R' = R$ (the scalars α'_i are determined by the level of the descendant). In order to see this, note that there are three cases to be distinguished:

$$\begin{cases} x'_i = x_i, & i \in \{0, \ldots, n-l-1\}, \\ x'_i = \frac{1}{2}\{x_0 + x_{n-l}\}, & i = n-l, \\ x'_i = x_i, & i \in \{n-l+1, \ldots, n\}. \end{cases}$$

First, for all $i \in \{1, \ldots, n-l-1\}$, $\alpha_i = 1$, yielding

$$y'_i = x'_i - x'_0 = x_i - x_0 = y_i = \alpha_i R \sum_{j=1}^{i} \pi(e_j) = R \sum_{j=1}^{i} \pi(e_j) = \alpha'_i R' \sum_{j=1}^{i} \pi'(e_j).$$

Second, for $i = n-l$, $\alpha_{n-l} = 1$, leading to

$$y'_i = \frac{1}{2}\{x_0 + x_{n-l}\} - x_0 = \frac{1}{2} y_{n-l} = \frac{1}{2}\alpha_{n-l} R \sum_{j=1}^{i} \pi(e_j) = \frac{1}{2} R \sum_{j=1}^{i} \pi(e_j) = \alpha'_{n-l} R' \sum_{j=1}^{i} \pi'(e_j),$$

and finally, for $i \in \{n-l+1, \ldots, n\}$, $\alpha_i = \frac{1}{2}$, showing that

$$y'_i = y_i = \alpha_i R \sum_{j=1}^{i} \pi(e_j) = \frac{1}{2} R \sum_{j=1}^{i} \pi(e_j) = \alpha'_i R' \sum_{j=1}^{i} \pi'(e_j).$$

Now consider the second descendant, with vertices $x'_0, x'_1, x'_2, \ldots, x'_n$. This satisfies the induction hypothesis, taking the associated permutation π' to be such that

$$\begin{cases} \pi'(e_i) = \pi(e_{i+1}), & i \in \{0, \ldots, n-l-1\}, \\ \pi'(e_i) = \pi(e_1), & i = n-l, \\ \pi'(e_i) = \pi(e_i), & i \in \{n-l+1, \ldots, n\}, \end{cases}$$

and taking the reflections matrix $R' = R \circ R_{\pi(e_1)}$, where $R_{\pi(e_1)}$ stands for a reflection in the $\pi(e_1)$ axis. In order to see this, there are again three cases to be distinguished:

$$\begin{cases} x'_i = x_{i+1}, & i \in \{0, \ldots, n-l-1\}, \\ x'_i = \frac{1}{2}\{x_0 + x_{n-l}\}, & i = n-l, \\ x'_i = x_i, & i \in \{n-l+1, \ldots, n\}. \end{cases}$$

First, for $i \in \{1, \ldots, n-l-1\}$ one finds that

$$\begin{aligned}
\boldsymbol{y}_i' &= \boldsymbol{x}_{i+1} - \boldsymbol{x}_1 = \boldsymbol{y}_{i+1} - \boldsymbol{y}_1 \\
&= \alpha_{i+1} R \sum_{j=1}^{i+1} \pi(\boldsymbol{e}_j) - \alpha_1 R\pi(\boldsymbol{e}_1) \\
&= R \sum_{j=1}^{i+1} \pi(\boldsymbol{e}_j) - R\pi(\boldsymbol{e}_1) \\
&= R \sum_{j=2}^{i+1} \pi(\boldsymbol{e}_j) \\
&= R \sum_{j=1}^{i} \pi(\boldsymbol{e}_{j+1}) \\
&= \alpha_i' R \sum_{j=1}^{i} \pi'(\boldsymbol{e}_j) \\
&= \alpha_i' R' \sum_{j=1}^{i} \pi'(\boldsymbol{e}_j),
\end{aligned}$$

using the facts that $\alpha_1 = \alpha_{i+1} = 1$ since $1 \leq i \leq i+1 \leq n-l$ and R can be replaced by R', since $\pi(\boldsymbol{e}_1) = \pi'(\boldsymbol{e}_{n-l})$ is not part of the summation. Second, for $i = n-l$

$$\begin{aligned}
\boldsymbol{y}_i' &= \frac{1}{2}\{\boldsymbol{x}_0 + \boldsymbol{x}_{n-l}\} - \boldsymbol{x}_1 = \frac{1}{2}\boldsymbol{y}_{n-l} - \boldsymbol{y}_1 \\
&= \frac{1}{2}\alpha_{n-l} R \sum_{j=1}^{n-l} \pi(\boldsymbol{e}_j) - \alpha_1 R\pi(\boldsymbol{e}_1) \\
&= \frac{1}{2} R \sum_{j=1}^{n-l} \pi(\boldsymbol{e}_j) - R\pi(\boldsymbol{e}_1) \\
&= \frac{1}{2} R \sum_{j=2}^{n-l} \pi(\boldsymbol{e}_j) - \frac{1}{2} R\pi(\boldsymbol{e}_1) \\
&= \frac{1}{2} R \sum_{j=1}^{n-l-1} \pi'(\boldsymbol{e}_j) - \frac{1}{2} R\pi'(\boldsymbol{e}_{n-l}) \\
&= \frac{1}{2} R' \sum_{j=1}^{n-l-1} \pi'(\boldsymbol{e}_j) + \frac{1}{2} R'\pi'(\boldsymbol{e}_{n-l}) \\
&= \frac{1}{2} R' \sum_{j=1}^{n-l} \pi'(\boldsymbol{e}_j) \\
&= \alpha_{n-l}' R' \sum_{j=1}^{n-l} \pi'(\boldsymbol{e}_j)
\end{aligned}$$

since $\alpha_1 = \alpha_{n-l} = 1$ and because R' only differs from R by an additional reflection in the $\pi(\boldsymbol{e}_1)$-axis. Finally, for $i \in \{n-l+1, \ldots, n\}$, one finds that

$$\begin{aligned}
\boldsymbol{y}'_i &= \boldsymbol{x}_i - \boldsymbol{x}_1 = \boldsymbol{y}_i - \boldsymbol{y}_1 \\
&= \alpha_i R \sum_{j=1}^{i} \pi(\boldsymbol{e}_j) - \alpha_1 R\pi(\boldsymbol{e}_1) \\
&= \frac{1}{2} R \sum_{j=1}^{i} \pi(\boldsymbol{e}_j) - R\pi(\boldsymbol{e}_1) \\
&= \frac{1}{2} R \sum_{j=1}^{n-l} \pi(\boldsymbol{e}_j) - R\pi(\boldsymbol{e}_1) + \frac{1}{2} R \sum_{j=n-l+1}^{i} \pi(\boldsymbol{e}_j) \\
&= \frac{1}{2} R' \sum_{j=1}^{n-l} \pi'(\boldsymbol{e}_j) + \frac{1}{2} R \sum_{j=n-l+1}^{i} \pi'(\boldsymbol{e}_j) \\
&= \alpha'_i R' \sum_{j=1}^{i} \pi'(\boldsymbol{e}_j) .
\end{aligned}$$

Note that $\pi' = \pi$ in the case of the first descendant, and that

$$\pi' = \pi \circ (n-l, 1, 2, \ldots, n-l-1, n-l+1, \ldots, n)$$

in the case of the second one. Note that for $0 \leq l < n$, $\boldsymbol{x}_1 - \boldsymbol{x}_0 = \boldsymbol{y}_1 = \alpha_1 \pi(\boldsymbol{e}_1) = \pi(\boldsymbol{e}_1)$. Therefore, using induction, one can easily show that for each simplex of level $0 \leq l \leq n$, R is given by $R = \mathrm{Diag}(\underline{1} - 2\boldsymbol{x}_0)$.

This completes the induction proof for simplices of level $0 \leq l \leq n$. Note that for $l = n$ all coordinates of all $\boldsymbol{y}_i$ are equal to 0, or to $\frac{1}{2}$, showing that the induction hypothesis also holds in the case where $n \leq l < 2n$. As this can be repeated for every $(k-1)n \leq l < kn$, the induction holds for all levels $0 \leq l$. □

The unique representation of a simplex formulated by Theorem 4.1 straightforwardly leads to some elementary results, stated in the following lemmata.

LEMMA 4.1. *Every created simplex of level n is congruent to a coarse grid simplex.*

LEMMA 4.2. *The number of congruency classes of simplices of each level* $0 \leq k < n$ *is equal to* 1.

Proof. Assume that T represented by $\underline{\alpha}$, R, and π is of the same level as T' represented by $\underline{\alpha}'$, R', and π'. Because they are of the same level, $\underline{\alpha} = \underline{\alpha}'$. The vertices of T are given by $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_n$, and those of T' by $\boldsymbol{x}'_0, \ldots, \boldsymbol{x}'_n$. Set $C(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{x}_0$, $C'(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{x}'_0$, and note that these are linear congruency mappings. Because $\boldsymbol{x} \in T$ implies that there exists nonnegative coefficients λ_j such that

$$\boldsymbol{x} = \boldsymbol{x}_0 + \sum_{j=1}^{n} \lambda_j [\boldsymbol{x}_j - \boldsymbol{x}_0] \quad \text{and} \quad \sum_{j=1}^{n} \lambda_j \leq 1 ,$$

one finds that

$$\begin{aligned}
C(\boldsymbol{x}) &= \sum_{j=1}^{n} \lambda_j [\boldsymbol{x}_j - \boldsymbol{x}_0] \\
&= \sum_{j=1}^{n} \lambda_j \boldsymbol{y}_j
\end{aligned}$$

$$= \sum_{j=1}^{n} \lambda_j \alpha_j R\pi(\hat{x}_j)$$

$$= R\pi\pi'^{-1}R'^{-1} \sum_{j=1}^{n} \lambda_j \alpha'_j R'\pi'(\hat{x}_j)$$

$$= R\pi\pi'^{-1}R'^{-1}C'(x'),$$

where

$$x' = x'_0 + \sum_{j=1}^{n} \lambda_j [x'_j - x'_0] .$$

As the mapping $C^{-1}R\pi\pi'^{-1}R'^{-1}C'$ is a congruency mapping from T' onto T, these simplices are congruent. □

LEMMA 4.3. *The number of congruency classes of simplices of all levels* $0 \leq k \leq n$ *is equal to* n.

Clearly, this shows that the number of created congruency classes is bounded by n, independent on the level of refinement. The following lemma will be used in §5.

LEMMA 4.4. *Assume that* $i \neq j$ *are such that* $0 < i, j \leq n$. *Let* y_k *be defined as in Theorem* 4.1 *for all* $k = 1, \ldots, n$. *Then the absolute value of all nonzero coordinates of* $y_i - y_j$ *is identical and equal to* $c \in \{\frac{1}{2}, 1\}$ *if and only if* $\alpha_i = \alpha_j = c$. *Otherwise one of* α_i, α_j *is equal to* $\frac{1}{2}$, *the other is equal to* 1.

Finally, note that in the bisection step in §3 it is not necessary to take the new vertex equal to

$$z := \frac{1}{2}\{x_0 + x_k\} .$$

In compliance with the local bisection algorithm provided in §5, one can also choose $\lambda \in (0, 1)$ and take

$$z := \lambda x_0 + (1 - \lambda)x_k ,$$

i.e., move the new vertex to a more suitable position along the edge spanned by x_0 and x_k. However, if this is done the number of congruency classes will change and Theorem 4.1 will no longer be straightforwardly applicable. However, a grid for which the vertices have been moved like this can still be easily de-refined. Moving vertices will be a topic of future research.

Theorem 4.1 applies to any unstructured coarse grid consisting of N n-simplices which is refined with the use of the bisection step introduced in §3, showing that the resulting number of congruency classes will be bounded above by $2^n \cdot N$ independent on the level of refinement. As mentioned in §3, the application of the bisection step is likely to lead to grid incompatibilities. It should be pointed out that in many cases one can eliminate the incompatibilities by using additional closure refinement, as for instance in [11] and [26]. However, one has to prevent the situation where a shared face has neighbors bisecting different edges, see for instance [7].

5. The local bisection refinement algorithm. This section introduces the n-dimensional local bisection refinement algorithm, using the local bisection step defined in §3. The presented algorithm is an extension to the refinement algorithm for two dimensions, given by Mitchell in [22]. It will ensure that the resulting locally refined grid of simplices remains compatible at all times.

In order to be able to introduce to refinement algorithm, we first need some elementary definitions. A simplex T' is called a neighbor of T if they have a face in common. Second, a simplex T' is called compatibly divisible with T if it is a neighbor and if T and T' will bisect the same edge. Now, avoiding incompatibilities by the assumption that a simplex and its neighbors can be bisected simultaneously, the recursive local bisection refinement algorithm for the refinement of an n-simplex is given by

```
Refine (simplex):
BEGIN
 WHILE  A neighbor is not compatibly divisible
 DO
   Refine (neighbor);
 END;
 Bisect (simplex);
 FOR each neighbor:
 DO
  Bisect (neighbor);
  END;
 END.
```

In this case, contrary to the proofs to be found in [22], it is not graph theory but Theorem 4.1 that provides the bases for the following theorem, which guarantees that the refinement algorithm terminates.

THEOREM 5.1. *Let T and a neighor T' be as in Theorem* 4.1. *Then there exists a unique $n+1$ permutation σ of $\{0, \ldots, n\}$, denoted by $\sigma = (\sigma(0), \ldots, \sigma(n))$,such that for all but one vertex $\boldsymbol{x}_i$ of T, $\boldsymbol{x}_{\sigma(i)}$ is a vertex of T'. Assume that T' shares the edge of T to be bisected. Then, under the assumption that $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are shared*

(1) *If $i \neq j$ such that $0 \leq i, j \leq k := n - l(T)$, then $0 \leq \sigma(i), \sigma(j) \leq k' := n - l(T')$. In addition one finds $\sigma(j) = \sigma(0) + j$ for all j, or $\sigma(j) = \sigma(0) - j$ for all j.*

(2) *If j is such that $i \leq k < j \leq n$, then $k' < \sigma(j) \leq n$. In addition $\sigma(j) = j$.*

(3) *Either $l(T') = l(T)$, or $l(T') = l(T) - 1$. In addition, in the first case T and T' are compatibly divisible, and in the second case T will be compatibly divisible with one of the descendants of T'.*

Proof. The existence of a unique permutation σ follows from the fact that T and T' share n vertices out of their $n+1$. First, assume that $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are different shared vertices such that $0 \leq i, j \leq k$ (for instance, vertex $\boldsymbol{x}_0$ and $\boldsymbol{x}_k$ are shared). Then

$$\boldsymbol{y}_i - \boldsymbol{y}_j = \boldsymbol{x}_i - \boldsymbol{x}_j = \boldsymbol{x}'_{\sigma(i)} - \boldsymbol{x}'_{\sigma(j)} = \boldsymbol{y}'_{\sigma(i)} - \boldsymbol{y}'_{\sigma(j)}.$$

Due to the choice of i and j, $\alpha_i = \alpha_j = 1$ whence Lemma 4.4 implies that $\alpha'_{\sigma(i)} = \alpha'_{\sigma(j)} = 1$. At its turn this leads to $0 \leq \sigma(i), \sigma(j) \leq k'$. Second, choosing $i \leq k < j \leq n$, one finds

$$\boldsymbol{y}_j - \boldsymbol{y}_i = \boldsymbol{x}_j - \boldsymbol{x}_i = \boldsymbol{x}'_{\sigma(j)} - \boldsymbol{x}'_{\sigma(i)} = \boldsymbol{y}'_{\sigma(j)} - \boldsymbol{y}'_{\sigma(i)},$$

where $\boldsymbol{y}_j - \boldsymbol{y}_k$ has nonzero coordinates of absolute value $\frac{1}{2}$ and 1. Because $\alpha'_{\sigma(i)} = 1$, it follows that $\alpha'_{\sigma(j)} = \frac{1}{2}$, implying that $k' < \sigma(j) \leq n$. The last claim follows from a pigeonhole principle, since all but one vertex of T is shared by T'. All additional properties follow from a pigeonhole principle, or by counting the number of nonzero entries in $\boldsymbol{y}_j - \boldsymbol{y}_i$. □

With the use of induction, Theorem 5.1 guarantees that the refinement algorithm terminates because neighbors are either compatibly divisible, or of lower level, and because the $n!$ coarse grid simplices in every one of the $k_1 \times \cdots \times k_n$ n-cubes form a compatibly divisible group.

Theorem 5.1 points out that even if T and T' are compatibly divisible one can encounter the situation where the row of vertices $x_0, \ldots, x_k$ is pointwise equal to $x'_k, \ldots, x'_0$ but for at most one point. Theorem 4.1 admits this case. Whether this case actually occurs depends on numbering of the vertices of the coarse initial grid.

For the initial grid presented in §2, compatibly divisible simplices T and T' are observed to always share vertices $x_0 = x'_0$. In order to see that this is very special, consider the following. Assuming that $x = x_i = x'_j$ is shared by T and T', one finds that $y_i = x_i - x_0 = x'_j - x'_0 = y'_j$, implying that $i = j$. Applying Theorem 4.1, this straightforwardly implies that for R, π, and $\underline{\alpha}$ describing T, and for R', π', and $\underline{\alpha'}$ describing T', one finds $R' = R$, $\underline{\alpha'} = \underline{\alpha}$, and $\pi' = \tau\pi$, where τ is an exchange permutation. Hence, in this special case, one can determine the compatibly divisible neighbors of T by determining their associated permutation (note that T has $n - 1$ neighbors sharing the edge to be bisected).

Finding neighbors to be bisected seems to depend on the datastructures used to represent the grid. One of the possibilities is to store the pointer to the neighbor across every face of every simplex. However, Theorem 4.1 shows that every simplex is uniquely determined by a combination of reflections, a permutation, and a scaling determined by its level. Further, Theorem 5.1 implies that it is possible to compute the reflections, the permutation, and scaling of a compatibly divisible neighbor, given those descriptors of the current simplex. Therefore, instead of storing pointers to all neighbors for every simplex, the neighbor pointers can be computed on demand, as in the three-dimensional case shown in Hebert [16]. This reduces the computer storage per simplex and increases the parallelism already present in the local refinement method. The author's nonoptimized Fortran 77 implementation of the presented bisection step and refinement algorithm takes less than three pages. It stores the pointers to neighbors and is fast, mainly due to the absence of local and global communication and computations.

6. Applications. This section provides some examples of grids obtained with the use of the local bisection refinement method introduced in §5. Thereafter, a short discussion on the upperbounds derived in §§4 and 5, is provided.

In order to show that the proposed bisection refinement method is applicable for the finite element solution of partial differential equations, consider Fig. 3. This figure shows a triangular grid, generated with the local bisection refinement algorithm of §5, along the position of a layer of the solution of a partial differential equation. This layer has the form of a curve, closely resembling the straight line $y = 0.45x$. The grid is generated as follows. First, a coarse grid of small squares is generated on an 8 by 5 division of the domain $[0, 1] \times [0, 0.625]$. Second, each rectangle is covered with a coarse grid of two triangles, analogously to the coarse grid covering $[0, 1]^2$. Then, for every level k, a finite element solution is computed on the grid of level k, and those triangles where the gradient of the computed finite element solution is larger than a certain threshold are bisected. In Fig. 3, grid 12 stands for the grid obtained after 12 levels of refinements along the curve. Because the level of refinement is already too high to distinguish details, Fig. 4 shows a magnification of a part of the previous grid. Note that the grid is compatible, and that all angles are either $\frac{1}{4}\pi$, or $\frac{1}{2}\pi$.

As a second example, consider some applications of the bisection refinement applied to the three-dimensional unit-cube $[0, 1]^3$, shown in Figs. 5–10. In all cases, refinement is applied repeatedly to those tetrahedrons and their descendants intersecting a plane (Fig. 5), intersecting a line (Fig. 6), or intersecting the hemi-sphere

$$\left(x - \frac{1}{2}\right)^2 + \left(y - \frac{1}{2}\right)^2 + \left(z - \frac{1}{2}\right)^2 = \frac{1}{16} \quad \text{with } x \geq \frac{1}{2},$$

as shown in Figs. 7–10. In all examples, the number of congruency classes is equal to 3, and

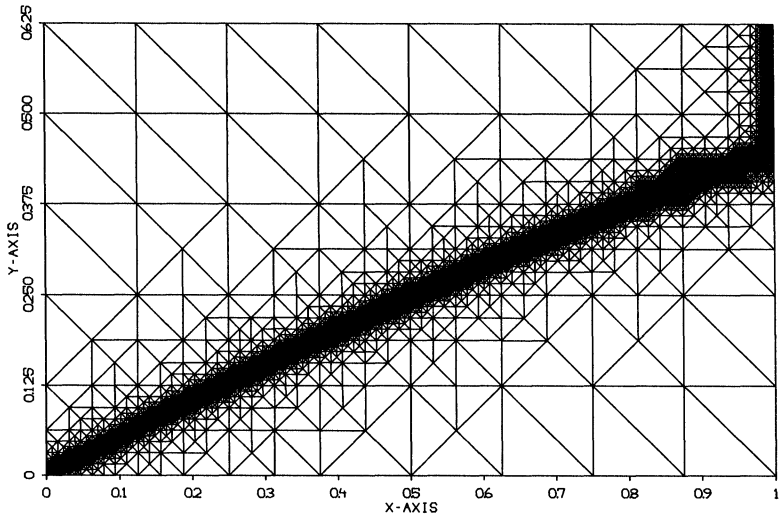

FIG. 3. *Triangular grid* 12 *on* $[0, 1] \times [0, 0.625]$.

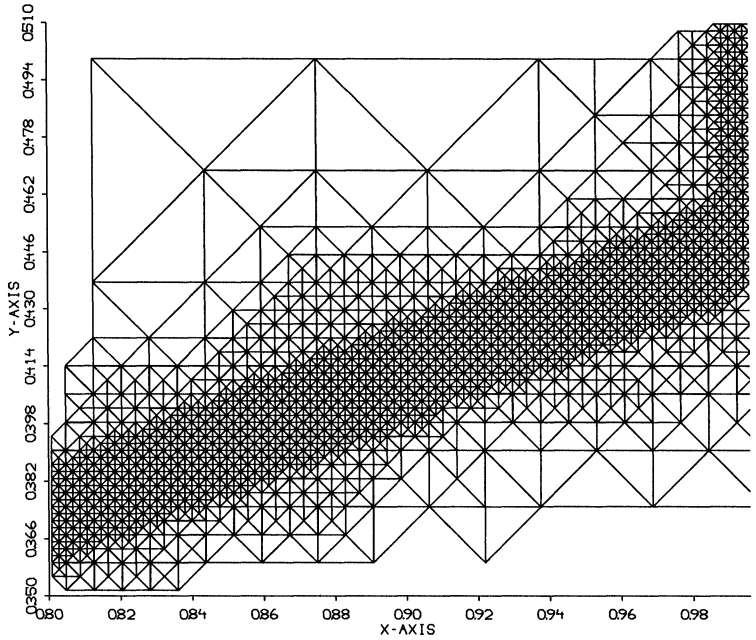

FIG. 4. *Magnification of grid* 12 *shown in Fig.* 3.

the figures clearly show the local nature of the repeatedly applied refinement. As the occuring angles are well bounded away from 0 and π, the grids are well suited for the computation of finite element solutions to partial differential equations, as was done in [21].

Finally, a short note concerning the recursion depth of the refinement algorithm. In two dimensions, the recursion depth can actually be equal to the maximum k, for any value of k. This maximum recursion depth will be attained in the following example, where for the sake of simplicity the initial coarse grid only consists of the triangle $\{(0, 0), (1, 0), (1, 1)\}$. Bisecting this triangle and one of its descendants repeatedly, one obtaines a row of triangles

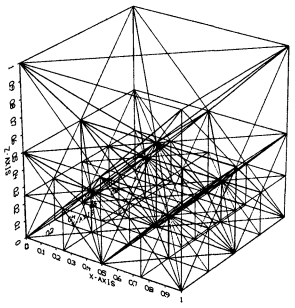

FIG. 5. *A cube which is six levels refined in the plane* $\{(x, y, z) : z = 0\}$.

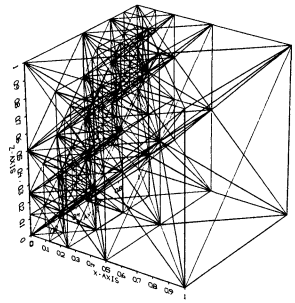

FIG. 6. *A cube which is nine levels refined along the line* $\{(x, y, z) : x = 0 \wedge y = z\}$.

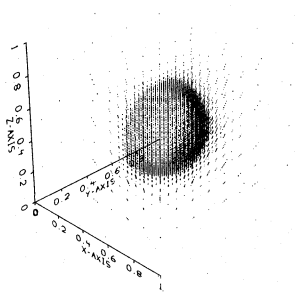

FIG. 7. *Vertices of grid* 18 *around the hemisphere* $(x - \frac{1}{2})^2 + (y - \frac{1}{2})^2 + (z - \frac{1}{2})^2 = \frac{1}{16}$ *with* $x \geq \frac{1}{2}$.

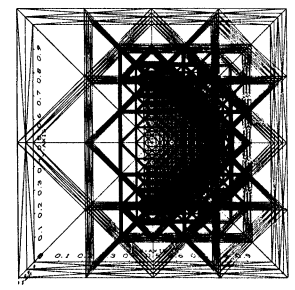

FIG. 8. *View on grid* 18 *from above.*

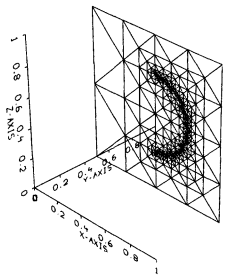

FIG. 9. *Cross-intersection with* $y = \frac{3}{8}$.

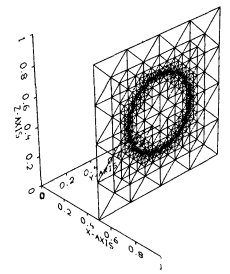

FIG. 10. *Cross-intersection with* $x = \frac{1}{2}$.

$$\{(0,0),(1,0),(1,1)\},\{(1,0),(1,1),(\tfrac{1}{2},\tfrac{1}{2})\},\{(1,1),(1,\tfrac{1}{2}),(\tfrac{1}{2},\tfrac{1}{2})\},$$
$$\{(1,1),(1,\tfrac{1}{2}),(\tfrac{3}{4},\tfrac{3}{4})\},\{(1,1),(1,\tfrac{3}{4}),(\tfrac{3}{4},\tfrac{3}{4})\}.$$

Together with the latter simplex, the simplex $\{(1,\frac{1}{2}),(1,\frac{3}{4}),(\frac{3}{4},\frac{3}{4})\}$ has been created. It is easy to verify that the recursion depth involved with the local bisection refinement of the latter simplex of level 4 is equal to 4. Since this refinement is actually the refinement around the coarse grid vertex $(1, 1)$ it obvious that the recursion depth upper bound k is obtained for every dimension n, for instance by refining the reference simplex as above around its vertex $\underline{1}$.

7. Conclusions. The n-dimensional local bisection refinement of grids of n-simplices is possible, without causing grid incompatibilities. The presented bisection algorithm is simple to implement, and the edges to be bisected are determined without using any communication or computation. This makes this bisection method eminently applicable for grid refinement in parallel. The refinement algorithm is applicable to a variety of grids, as long as the vertices of every simplex in the grid are properly ordered. This paper proves that special regular grids of simplices on $[0, 1]^n$ admit the desired ordering of the vertices. In addition, it is argued that the refinement method is also applicable to grids which are the image of (part of) the (possibly locally refined) special grid on $[0, 1]^n$ under a nonsingular mapping. The bisection method is applicable to every triangular grid because it is equivalent to Mitchell's method in [22], and can possibly be applied to many tetrahedral grids with the use of additional closure refinement. Moving the grid by shifting the vertices is allowed since this does not alter the vertices' ordering. Further, it is highly likely that many more simplicial grids admit a proper ordering of the vertices. This is a topic for future research.

The new bisection refinement method can locate an element containing a certain point as fast as the quadrant and octant methods can. Independent of the dimension, it allows for the usage of binary data structures, and therefore may make use of many well-known binary tree

traversal and balancing techniques. In addition, it may be well suited for the determination of fixed points, where the dimension is usually large.

Further, the number of congruency classes created is n, independent of the level of refinement, which means that the bisection method is naturally suited for finite element applications, as the smallest and largest angle are a priori determined and bounded by the initial coarse grid, independent of the level of refinement. For arbitrary unstructured coarse simplicial grids, allowing for incompatibilities, the number of congruency classes turns out to be $2^n \cdot N$, where N stands for the number of congruency classes of coarse grid simplices.

The recursion depth of the refinement algorithm and the number of simplices to be processed during the refinement of one simplex are a function of the level k, and are a priori determined, simplifying the implementation of this algorithm.

Finally, note that the local bisection refinement algorithm can largely reduce the number of simplices to be used for the approximation of solution manifolds of parametrized equations.

Acknowledgments. I would like to thank J. Bey for the stimulating discussions concerning the use of permutations for the construction of a coarse grid of tetrahedrons. In addition I would like to thank J. Aarden for the indication that n-dimensional bisection refinement should be possible.

REFERENCES

[1] E. Allgower and K. Georg, *Generation of triangulations by reflections*, Utilitas Math., 16 (1979), pp. 123–129.
[2] D. C. Arney and J. E. Flaherty, *An adaptive mesh-moving and local refinement method for time-dependent partial differential equations*, A.C.M. Trans. Math. Software, 16 (1990), pp. 48–71.
[3] O. Axelsson and V. A. Barker, *Finite Element Solution of Boundary Value Problems*, Academic Press, Orlando, FL, 1984.
[4] E. F. D'Azevedo, *Optimal triangular mesh generation by coordinate transformation*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 755–786.
[5] T. J. Baker, *Developments and trends in three-dimensional mesh generation*, Appl. Numer. Math., 5 (1989), pp. 275–304.
[6] R. E. Bank and B. D. Welfert, *A posteriori error estimates for the Stokes problem*, SIAM J. Numer. Anal., 28 (1991), pp. 591–623.
[7] E. Bansch, *Local mesh refinement in 2 and 3 dimensions*, Impact of Comput. Sci. Engrg., 3 (1991), pp. 181–191.
[8] M. Bern and D. Eppstein, *Mesh generation and optimal triangulation*, Technical Report CSL-92-1, Xerox PARC and University of California, Irvine, 1992.
[9] M. Bern, D. Eppstein, and F. Yao, *The expected extremes in a Delaunay triangulation*, Internat. J. Comput. Geom. Appl., 1 (1991), pp. 79–91.
[10] M. Bern, D. Eppstein, and J. R. Gilbert, *Provably good mesh generation*, in 31st Annual Symposium on Foundations of Computer Science, Vol. I, II (St. Louis, MO., 1990), IEEE Comput. Soc. Press, Los Alamitos, CA, 1990, pp. 231–241.
[11] J. Bey, *Simplicial grid refinement in three and more dimensions*, Technical Report of the Mathematisches Institut, Universität Tübingen, Auf der Morgenstelle 10, D-72076 Tübingen, Germany, in preparation.
[12] S. W. Bova and G. F. Carey, *Mesh generation/refinement using fractal concepts and iterated function systems*, Internat. J. Numer. Methods Engrg., 33 (1992), pp. 287–305.
[13] H. Freudenthal, *Simplizialzerlegungen von Beschränkter Flachheit*, Ann. Math. 43 (1942), pp. 590–582.
[14] W. H. Frey and D. A. Field, *Mesh relaxation: A new technique for improving triangulations*, Internat. J. Numer. Methods Engrg., 31 (1991), pp. 1121–1133.
[15] P. L. George and F. Hecht, *Automatic mesh generator with specified boundary*, Comput. Methods Appl. Mech. Engrg., 92 (1991), pp. 269–288.
[16] D. J. Hebert, *Symbolic local refinement of tetrahedral grids*, Technical Report ICMA-93-181, University of Pittsburgh, J. Symbolic Comput., 17 (1994), pp. 457–472.
[17] W. D. Henshaw and G. Chesshire, *Multigrid on composite meshes*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 914–923.
[18] D. E. Knuth, *The Art of Computer Programming, Vol. 3, Sorting and Searching*, Addison-Wesley, Reading, MA, 1973.
[19] I. Kossaczky, *A recursive approach to local mesh refinement in two and three dimensions*, J. Comput. Appl. Math., to appear.

[20] R. LOHNER, *Some useful data structures for the generation of unstructured grids*, Comm. Appl. Numer. Methods, 4 (1988), pp. 123–135.

[21] J. M. MAUBACH, *Iterative Methods for Non-Linear Partial Differential Equations*, C.W.I., Amsterdam, 1991.

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

[23] M. ONG, *Hierarchical basis preconditioners for second order elliptic problems in three dimensions*, Ph.D. Thesis, CAM report 89-31, Department of Mathematics, University of California at Los Angeles, Los Angeles, CA, 1993.

[24] W. C. RHEINBOLDT, *On the computation of multidimensional solution manifolds of parameterized equations*, Numer. Math., 53 (1988), pp. 165–180.

[25] R. RIEDINGER, M. HABAR, P. OELHAFEN, AND H. J. GUNTHERODT, *About the Delaunay-Voronoi tesselation*, J. Comput. Phys., 74 (1988), pp. 61–72.

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

[27] W. J. SCHRÖDER AND M. S. SHEPHARD, *Geometry-based fully automatic mesh generation and the Delaunay triangulation*, Internat. J. Numer. Methods Engrg., 26 (1988), pp. 2503–2515.

[28] M. S. SHEPHARD AND M. K. GEORGES, *Reliability of automatic 3-D mesh generation*, Comput. Methods Appl. Mech. Engrg., 101 (1992), pp. 443–462.

[29] E. SPERNER, *Neuer Beweis für die Invarianz der Dimensionszahl und des Gebietes*, Abh. Math. Sem. Univ. Hamburg 6.

[30] M. J. TODD, *The Computation of Fixed Points and Applications*, Lecture Notes in Economics and Mathematical Systems 124, Springer Verlag, Berlin, 1976.