SOME REFINEMENT ALGORITHMS AND DATA STRUCTURES FOR REGULAR LOCAL MESH REFINEMENT *

RANDOLPH E. BANK†, ANDREW H. SHERMAN‡, AND ALAN WEISER§

Abstract. In this paper, we consider local mesh refinement algorithms and data structures for finite element methods for linear elliptic partial differential equations in the plane. Quadrilateral and triangular are treated in a unified fashion. Because we restrict the local refinement to be regular, the resulting finite element systems are always sparse, and the refinement algorithms can be implemented efficiently, in time proportional to the number of elements.

1. Introduction. The finite element method for computing approximate solutions to linear elliptic partial differential equations in the plane [12] is usually considered to have three parts: subspace selection, matrix assembly, and matrix solution. If a particular approximate solution is deemed too inaccurate, improvement may be obtained by selecting a larger subspace, using either mesh refinement or higher-order basis functions. In this paper, we consider various aspects of mesh refinement, particularly in the context of accuracy improvement in the presence of singularities in the solution to the partial differential equation.

A *finite element mesh* is a subdivision of the domain Ω of the differential equation into a number of polygons, called *elements*. In this paper, we deal only with meshes formed of shape-regular quadrilateral or triangular elements, and we assume that Ω is, in fact, specified as the union of such elements. Extensions to allow for curved edges in the boundary of Ω are straightforward.

Throughout this paper, we assume that the finite element basis functions are piecewise linear for triangular meshes or bilinear (isoparametric) for quadrilateral meshes. In part, the reason for this is simplicity, since many of our remarks apply as well to higher order bases. Moreover, since we are concerned with mesh refinement in the presence of singularities, it may often be the case that little benefit can be gained through the use of higher order, more costly finite elements. For discussion of accuracy improvement through the use of higher order elements for problems with smooth solutions or point singularities, see [1, 14].

Mesh refinement strategies are of three types. In *global mesh refinement*, every element in the mesh is refined to obtain a finer mesh. In most cases, this is the simplest strategy to implement, but it can be very wasteful in the sense that many elements are generated away from the areas of interest. A modification of the global strategy is *semi-local mesh refinement*, in which elements in one or more selected cross-sections of the mesh are refined. In certain cases this strategy may be implemented as easily as global refinement and may be less wasteful. However, complications occur when the desired refinement does not correspond to a natural cross-section of the mesh.

The last refinement strategy is (adaptive) *local mesh refinement*, in which only selected elements are refined. This is an attractive strategy for problems involving either sharp fronts or point singularities, because the refinement can be restricted to those portions of Ω where it is required. However, for adaptive local mesh refinement to be efficient, it is necessary that:

(i) the best (or nearly best) elements to refine can be selected cheaply;

* *Scientific Computing*, R. Stepleman *et al* (eds.), IMACS/North-Holland, 1983.

† Department of Mathematics, University of California at San Diego, La Jolla, California 92093.

‡ Exxon Production Research Company, Houston Texas 77001.

§ Exxon Production Research Company, Houston Texas 77001.

(ii) the resulting systems of linear equations retain their sparseness as the mesh is refined; and

(iii) the local refinement process itself can be implemented efficiently.

For our purposes, we will assume that (i) holds; there is developing an extensive literature on this selection problem, and the reader is referred to such work as [14, 4, 8] for details. In this work we focus on (ii) and (iii). For example, if the system of equations becomes increasingly dense as the mesh is refined, then the advantage gained by refining only a small number of elements is partially lost. (In global refinement, all matrices retain the original sparseness.) And if the local refinement process cannot be implemented efficiently, then the actual cost of computing an approximate solution to the differential equation may actually be higher with local refinement than with the other two strategies, even in cases where the other strategies are extremely wasteful.

An outline of this paper is as follows. Section 2 introduces notation and other basic information common to all the meshes we consider. Sections 3 and 4 develop formal criteria for regular meshes and refinement strategies that guarantee that (ii) holds for quadrilateral and triangular meshes, respectively. Section 5 describes a sample refinement algorithm that, we will show, generates refined meshes satisfying the criteria developed in Sections 3 and 4. Section 6 discusses the implementation of the refinement algorithm, concentrating mainly on the required data structures, and demonstrating that the algorithm, in some sense, optimally satisfies (iii). An appendix outlines the proof of several statements made in Sections 3 and 4. We note that some of the information in this paper has appeared previously, in less unified form, in [14, 7, 6, 5].

2. Definitions and Notation. Let Ω be a bounded, polygonal region of $\mathbb{R}^2$. The domain Ω is assumed to be decomposed into a small number of *macro elements* t_i, $1 \le i \le nt_0$ which form a tessellation $\mathcal{T}_0$ of Ω. The macro elements are either triangles or quadrilaterals, and we require that is $t_i, t_j \in \mathcal{T}_0$, $t_i \ne t_j$, then $\bar{t}_i \cap \bar{t}_j$ is either empty, a common vertex, or a shared edge.[1]

We assume all macro elements are shape regular; that is, each element's side lengths are bounded above and below by fixed multiples of that element's diameter, and all interior angles are uniformly bounded above 0 and below π. In particular, quadrilateral elements must be strictly convex. Shape regularity does not require global quasi-uniformity of the mesh.

Globally, we denote the vertices of $\mathcal{T}_0$ by v_k, $1 \le k \le nv_0$ and the edges by e_k, $1 \le k \le ne_0$, As with the usual types of finite element computations (e.g., matrix assembly), it is important in our algorithms to have also have local notation to describe the edges and vertices of a given element. Thus we say that element t_i contains k_e vertices ν_i^j, $1 \le j \le k_e$ and k_e edges ε_i^j, $1 \le j \le k_e$, where $k_e = 3$ for triangles and $k_e = 4$ for quadrilaterals (cf. Figure 2.1). Thus, for $1 \le i \le nt_0$, and $1 \le j \le k_e$, $\nu_i^j = v_k$ for some k, $1 \le k \le nv_0$ and $\varepsilon_i^j = e_\ell$ for some ℓ, $1 \le \ell \le ne_0$. Throughout this paper, we will view local designations (e.g., ν_i^j) and global designations (e.g., v_k) as interchangeable names for a unique entity, and use whichever designation makes more sense in context.

To ensure the shape regularity of the elements in $\mathcal{T}_0$ is inherited by elements created during the refinement process, we restrict attention to *bisection-type mesh refinement*. A triangular element t_i is subdivided into four geometrically similar triangles, called the *sons* of t_i, by pairwise connecting the midpoints of the three edges

[1] $\bar{t}_i$ denotes the closure of t_i.

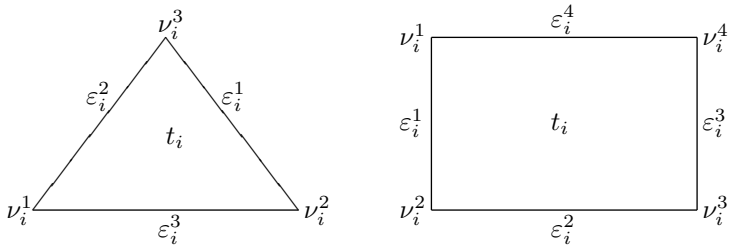


FIG. 2.1. *Local Notation.*

of t_i. Element t_i is referred to as the *father* of its sons. In the case of quadrilaterals, the sons are created by joining midpoints of opposing sides of t_i (see Figure 2.2). In this case, the sons are not geometrically similar to their father (except for the important special case in which t_i is a parallelogram), but they remain shape regular. Indeed, they become "nicer" in the sense that the bilinear mapping which maps the unit square (or reference element) to a son of t_i is more nearly affine than the corresponding mapping for t_i itself. In Figure 2.2, the numbers in the corners of the quartet of sons are to be interpreted as the superscript j in the local designations $\nu_{s_i+k}^j$ of the vertices of t_{s_i+k}, $0 \le k \le 3$. They define the geometric orientation of each member of the quartet relative to the father. In particular, note that vertices of each son in Figure 2.2 are locally ordered in a fashion consistent with Figure 2.1. It is this consistency, and the fact that the sons of an element bear known, fixed relationships to their father, and not the specific conventions we adapt, that are important in the algorithms we discuss.

In the case of triangles, we will use a second type of refinement called *"green" refinement*, in which a vertex of t_i is connected to the midpoint of the opposite edge (Figure 2.3). While green refinement creates elements which can be less shape regular than their father, we will restrict its use so that no element created by green refinement is ever refined itself.

An *admissible mesh* [4, 3] is a collection of refined and unrefined elements, which is recursively defined as follows:

(i) $\mathcal{T}_0$ is an admissible mesh; and

(ii) if $\mathcal{T}$ is admissible and $t \in \mathcal{T}$ is refined, then $\mathcal{T} \cup$ sons of t is admissible.[2]

Given an admissible mesh $\mathcal{T}$, and vertex v which is a corner of each unrefined element it touches is called *regular*, and all other vertices are called *irregular* (Figure 2.4). Because admissible meshes allow irregular vertices, they are not necessarily tessellations of Ω.

The *irregularity index* [4] of a mesh $\mathcal{T}$ is the maximum number of irregular vertices on a side of any unrefined element. A *k-irregular mesh* has irregularity index no greater than k. For example, the mesh in Figure 2.4 is a 3-irregular mesh. Vertices lying on $\partial\Omega$ (which are necessarily regular) are called *boundary vertices*; all other vertices are called *interior vertices*.

Each edge of an element t_i is either a boundary edge of Ω or part of the perimeter of one or more other elements in the mesh. We define the *neighbor* of t_i across edge ε_i^j to be the smallest element with one edge which completely overlaps ε_i^j. For convenience, we denote both this neighbor and its global element number by τ_i^j. It is

[2]Here sons of t are created only by regular refinement.

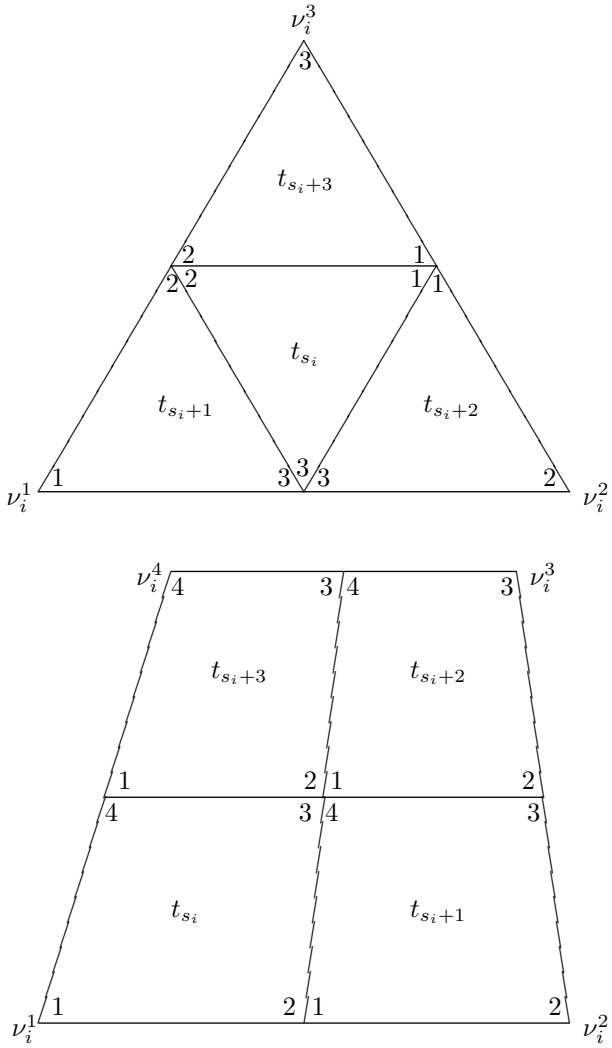


FIG. 2.2. *Bisection-Type Mesh Refinement.*

important to note that the neighbor relation is nonsymmetric and time-dependent.

The *level* ℓ_i of element t_i is defined inductively as follows

$$\ell_i = \begin{cases} 1 & \text{if } t_i \in \mathcal{T}_0 \\ \ell_{f_i} + 1 & \text{if } t_i \notin \mathcal{T}_0 \end{cases},$$

where t_{f_i} is the father of t_i.

If we regard Ω as a pseudo-element whose "sons" are the macro-elements of $\mathcal{T}_0$, the process of creating an admissible mesh can be viewed as the creation of an *element tree*, in which the root is Ω, other nodes correspond to macro-elements or elements created during the refinement process, and the branches lead downward from an element to its sons. All internal nodes in the tree have exactly four sons, except for Ω, which has nt_0 sons. Leaves of the tree correspond to unrefined elements. The level ℓ_i of t_i then

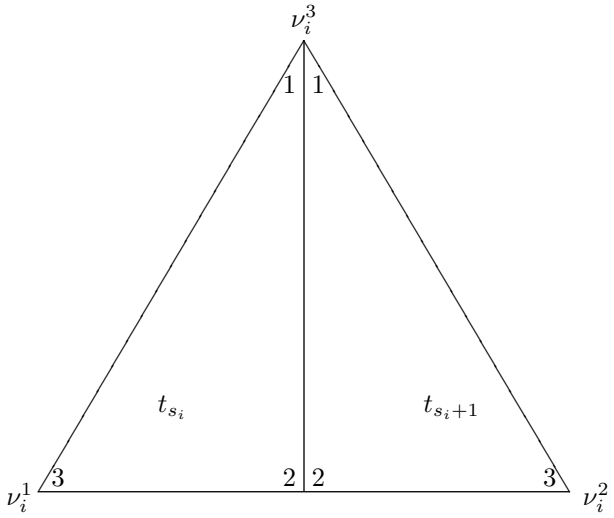


FIG. 2.3. *Green Refinement (One of Three Possibilities).*

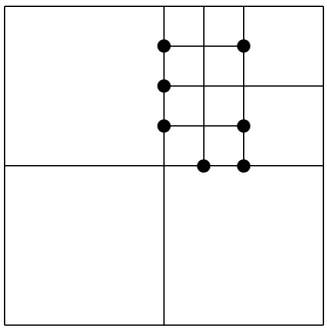


FIG. 2.4. *Irregular Vertices.*

corresponds to the distance from t_i to Ω in the tree; that is, the length of the shortest part from t_i to the root.

In general, it is advantageous to "regularize" an admissible mesh $\mathcal{T}$ by restricting the number of irregular vertices on each edge. There are several reasons for this: simplifying computations such as matrix assembly and mesh refinement, increasing approximation power by insuring that neighboring elements are not of vastly differing sizes, and guaranteeing that each element is in the support of a bounded number of basis functions. (cf. Sections 3 and 4). There are a number of ways to accomplish this regularization, but we shall mainly consider the following *1-irregular rule* [14, 6, 5] and several of it variants.

> **1-Irregular Rule:** Refine any unrefined element for which any of the sides contains more than one irregular vertex.

We shall see that applying such a rule as often as possible to an admissible mesh $\mathcal{T}$ leads to a regularized mesh having several important properties described in the next two sections. The algorithm described in Section 5 incorporates both this rule

and the $k_e - 1$ *neighbor rule*:

> **$k_e - 1$ Neighbor Rule:** Refine any element with $k_e - 1$ neighbors that have been regularly refined.

3. Quadrilateral Meshes. In this section, we consider bisection-type local mesh refinement for quadrilateral meshes (e.g, [14, 4, 2]). Let $\mathcal{T}_0$ be the given initial mesh and $\mathcal{T}$ be an admissible mesh generated from $\mathcal{T}_0$. For convenience, in this section we assume $\mathcal{T}_0$ consists of a single square macro element. Thus we can use bilinear, rather than isoparametric basis functions in our discussions, and the initial adjacency information about elements in $\mathcal{T}_0$ is not required. These simplifications do not have a significant impact on the issues addressed in this section.

Let $\mathcal{S} = \mathcal{S}(\mathcal{T})$ denote the space of $\mathcal{C}^0$ piecewise bilinear functions associated with $\mathcal{T}$.

$$\mathcal{S} = \{\phi \,|\, \phi \text{ is continuous and } \phi|_{t\in\mathcal{T}} \text{ is bilinear}\}$$

We define a basis $\mathcal{B} = \{b_i\}$ for $\mathcal{S}$ as the unique set of elements of $\mathcal{S}$ satisfying the Lagrange conditions

$$b_i(v_j) = \delta_{ij}$$

for all regular vertices v_j in $\mathcal{T}$. Here δ_{ij} is the Kronecker delta. The basis $\mathcal{B}$ is well defined [14, 4]. This choice of basis gives a natural correspondence between basis functions and the regular vertices in the mesh. Figure 3.1 shows a typical case in which the elements forming the support of b_i are denoted by dots.

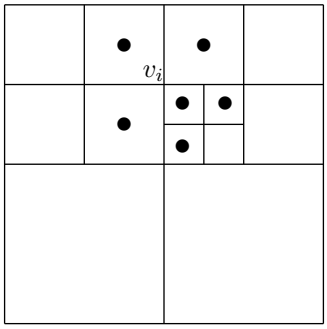


FIG. 3.1. *Support of b_i.*

In a mesh with no irregular vertices, the support of a given basis function b_i is limited to those unrefined elements which have v_i as a vertex. There are thus at most four nonzero, linearly independent basis functions in any given element. These two properties imply that the finite element stiffness matrix will be sparse and the elementwise assembly procedure will be similar for all elements. The following example [14] shows that these desirable properties cannot be guaranteed for general admissible meshes.

Let $\mathcal{T}_0$ be $(0,1) \times (0,1)$, and let $\mathcal{T}_\ell$ be the $(\ell+1)$-level mesh resulting from the regular refinement of exactly those elements containing the point $p = (2/3, 2/3)$. $\mathcal{T}_4$ is shown in Figure 3.2. Since $2/3 = 1 - 1/2 + 1/4 - 1/8 + 1/16 - \ldots$, $\mathcal{T}_\ell$ is formed by alternately refining the upper right or lower left son of the of the smallest currently refined element containing p. Each mesh $\mathcal{T}_\ell$, $\ell \geq 3$ is 2-irregular.

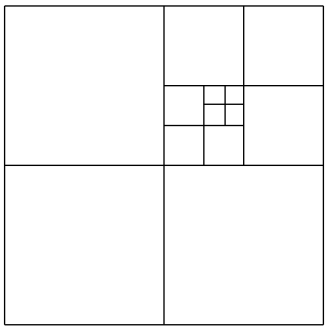


FIG. 3.2. $\mathcal{T}_4$.

The regular vertices in $\mathcal{T}_\ell$ are the ten boundary vertices and the ℓ interior vertices

$$v_i = (p_i, p_i), \qquad p_i = \sum_{j=0}^{i} \left(\frac{1}{2}\right)^j, \quad 1 \le i \le \ell.$$

Then the smallest unrefined element in $\mathcal{T}_\ell$ containing p is in the support of the basis functions corresponding to v_i, $1 \le i \le \ell$, and the resulting stiffness matrix will be essentially dense for large values of ℓ. The element assembly process will also be complicated due to the large numbers of nonzero basis functions in many elements.

This provides the motivation for considering subsets of the class of admissible meshes for which certain desirable properties can be guaranteed. Our basic strategy is to take an arbitrary admissible mesh $\mathcal{T}$ and to generate a more refined mesh $\mathcal{T}'$ having the desired properties by applying one or more refinement rules. Let $\mathcal{T}'$ have unrefined elements $\{t_i'\}$, regular vertices $\{v_i'\}$, and associated finite element space $\mathcal{S}'$ with basis $\mathcal{B}' = \{b_i'\}$.

For example, a mesh $\mathcal{T}'$ generated using the 1-irregular rule will satisfy the following properties (see [14]):

- Q1. $\mathcal{T}'$ is 1-irregular;
- Q2. $\mathcal{T}'$ uniquely contains the fewest elements of any 1-irregular mesh containing $\mathcal{T}$;
- Q3. There are at most four basis functions b_j' having support in any element t_i';
- Q4. The restrictions to any element t_i' of those basis functions functions nonzero in t_i' are linearly independent;
- Q5. The support of any basis function b_i' intersects the support of at most twelve other basis functions b_j';
- Q6. $|\{t_i'\}| \le 9|\{t_i\}|$;
- Q7. $\{v_i'\}$ can be partitioned into mutually disjoint sets $V_1, V_2, \ldots, V_{12}$ such that for any distinct v_i' and v_j' in the same set V_k, $\mathrm{supp}(b_i') \cap \mathrm{supp}(b_j')$ contains no elements.

The effect of properties Q1-Q7 is to ensure the efficiency of the overall solution process. Q1, Q3, and Q4 combine to allow the assembly procedure for the stiffness matrix that is much simpler than what might be required for $\mathcal{T}$. Q1 implies that the levels of neighboring elements differ by at most one, and allows the use of mesh data structures that conserve storage without a severe penalty in computational effort (see Sections 5 and 6). Q2 shows that no mesh with these two properties can have fewer

elements than $\mathcal{T}'$. Q6 ensures that the stiffness matrix for $\mathcal{T}'$ is not too much larger than the one for $\mathcal{T}$. and Q5 shows that it is not significantly denser. (in fact Q5 and Q6 are usually very pessimistic; normally $\mathcal{T}'$ has only a few extra elements, and rows of the stiffness matrix have nine nonzeroes rather than the thirteen suggested by Q5.) In the terminology of [3], Q6 shows that the *overlap index* of $\mathcal{S}'$ is bounded by twelve. Q7 is of interest in applying iterative methods to the solution of the system for $\mathcal{T}'$. For instance, Q7 implies that an SOR iteration for the system can be broken up into separate vectorizable iterations for each V_k, since the new coefficient for a basis function in V_k does not depend on any other new coefficient for a basis function in V_k.

In the Appendix, we outline a proof that for an arbitrary admissible mesh $\mathcal{T}$, Q4 holds if and only if

Q8. There exist no unrefined elements t_1, t_2 and t_3 with $\ell_1 < \ell_2 < \ell_3$, all containing a common irregular vertex on their boundaries.

For quadrilaterals, the 1-irregular rule is equivalent to the following *big element rule*:

Big-Element Rule: Whenever Q8 is violated, refine t_1.

We do not know of other rules that always require no more mesh refinement than the 1-irregular rule but are still sufficient to guarantee properties Q3-Q6. For example, we might use the following *middle element rule:*

Middle-Element Rule: Whenever Q8 is violated, refine t_2.

The resulting mesh would satisfy Q3 and Q4 and might contain fewer elements than $\mathcal{T}'$, but no property analogous to Q5 could be insured.

A 2-irregular rule analogous to the 1-irregular rule cannot guarantee Q3, since the mesh illustrated in Figure 3.2 would satisfy such a rule. In the case of quadrilateral elements, the 1-irregular rule by itself is sufficient to guarantee Q1-Q7. Nonetheless, it may be desirable to also impose the 3-neighbor rule. By refining an element with irregular vertices on three sides, one adds only two additional vertices but obtains four additional basis functions in $\mathcal{S}'$. Meshes generated using a combination of the two rules satisfy properties analogous to Q1-Q7, with, in some cases, larger constants.

4. Triangular Meshes. In this section, we consider bisection-type local mesh refinement for triangular meshes (e.g, [6, 5]). As in the case of quadrilaterals, our goal is to develop a procedure which is efficient for mesh generation and use in matrix assembly and solution.

Let $\mathcal{T}_0$ be a given initial mesh and $\mathcal{T}$ be an admissible mesh generated from $\mathcal{T}_0$. For convenience, in this section we assume $\mathcal{T}_0$ consists of a single triangle. Let $\mathcal{S} = \mathcal{S}(\mathcal{T})$ denote the space of $\mathcal{C}^0$ piecewise linear functions associated with $\mathcal{T}$.

$$\mathcal{S} = \{\phi \,|\, \phi \text{ is continuous and } \phi|_{t\in\mathcal{T}} \text{ is linear}\}$$

We define the Lagrange basis $\mathcal{B} = \{b_i\}$ for $\mathcal{S}$ as in the case of quadrilateral elements.

Given an admissible mesh $\mathcal{T}$, we can obtain a more refined mesh $\mathcal{T}'$, with unrefined elements $\{t_i'\}$, regular vertices $\{v_i'\}$, subspace $\mathcal{S}'$ and basis $\mathcal{B}' = \{b_i'\}$, by application of the 1-irregular rule.

The properties of the 1-irregular rule when applied to triangular meshes differ somewhat from the case of quadrilateral elements. First, unlike quadrilaterals, refinement of triangular elements does not always generate new regular vertices. Let the $\ell+1$ level mesh $\mathcal{T}_\ell$ be generated by successive refinement of the center element of the smallest quartet of currently unrefined elements. $\mathcal{T}_3$ is illustrated in Figure 4.1. All the $\mathcal{T}_\ell$ are 1-irregular. However, the only regular nodes for $\ell > 2$ are the six boundary nodes.

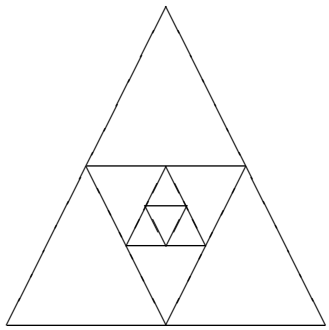


FIG. 4.1. $\mathcal{T}_3$.

Second, the 1-irregular rule for triangular meshes does not guarantee exactly three basis functions will be nonzero in each element. Consider the situation in Figure 4.2. The mesh satisfies the 1-irregular rule, but the four basis functions corresponding to the vertices marked by dots are nonzero in t.

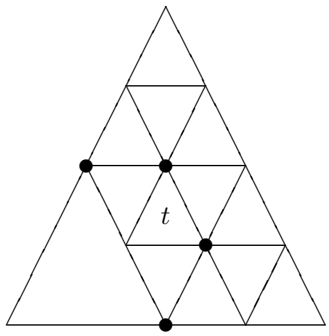


FIG. 4.2. *Four Nonzero Basis Functions in t.*

Given a 1-irregular mesh $\mathcal{T}'$, we can generate a more refined mesh $\mathcal{T}''$ by applying, wherever possible, the following *green rule:*

> **Green Rule:** With as few elements as possible, triangulate any unrefined element with an irregular vertex on one or more of it sides.

The three situations in which the green rule can occur are shown in Figure 4.3.

The following analogues of Q1-Q7 hold:

T1. $\mathcal{T}'$ is 1-irregular;
T2. $\mathcal{T}'$ uniquely contains the fewest elements of any 1-irregular mesh containing $\mathcal{T}$;

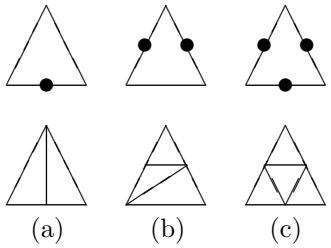


FIG. 4.3. *The Green Rule.*

T3. There are at most three basis functions b_j'' having support in any element $t_i'' \in \mathcal{T}''$.
T4. The restrictions to any element t_i'' of those basis functions functions nonzero in t_i'' are linearly independent;
T5. The support of any basis function b_i'' intersects the support of at most twelve other basis functions b_j'';
T6. $|\{t_i'\}| \leq 13|\{t_i\}|$ and $|\{t_i''\}| \leq 2|\{t_i'\}|$;
T7. $\{v_i'\}$ can be partitioned into mutually disjoint sets $V_1, V_2, \ldots, V_9$ such that for any distinct v_i' and v_j' in the same set V_k, $\text{supp}(b_i') \cap \text{supp}(b_j')$ contains no elements.

T5 and T6 are usually pessimistic. The most common number of nonzeroes in a row of the stiffness matrix is seven, and for most meshes encountered in practice $\mathcal{T}''$ contains fewer than twice as many elements as $\mathcal{T}$.

The big element rule and the 1-irregular rule are not equivalent for triangular meshes; the big element rule may force more refinement than the 1-irregular rule. While the green rules solves the problem of irregular vertices by regularizing them at the conclusion of the refinement process, other strategies are possible. For example, a rule like

> Refine the neighbors of any element whose refinement is dictated by accuracy considerations.

used in conjunction with, say, the big element rule, forces the irregular vertices to lie in regions where refinement is not required for reasons of accuracy. In that case, regularizing may not be necessary.

The 2-neighbor rule, used in conjunction with the 1-irregular rule, leads to 1-irregular meshes in which each remaining irregular vertex is located at the midpoint of an edge of a unique unrefined element. When the green rule is applied to such a mesh, the only situation which occurs is that of Figure 4.3a. Analogues of T1-T6 hold for this process, but the constants are somewhat larger.

Note that the green rule could also be used with quadrilateral elements, refining quadrilaterals with irregular vertices on their sides into quadrilaterals and triangles. The three special cases for quadrilateral meshes satisfying the 1-irregular and 3-neighbor rules are shown in Figure 4.4.

One can also use the green rule in conjunction with the k-irregular rule for any fixed $k > 1$. Analogues of T1-T6 would hold for such meshes, with differing constants. The number of special cases requiring irregular refinement as in Figure 4.3 increases with increasing k. The amount by which the shape regularity of the result elements

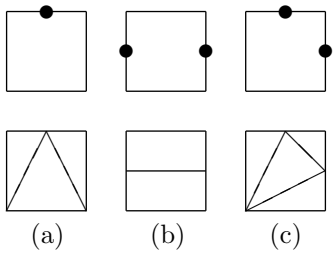


FIG. 4.4. *The Green Rule for Quadrilaterals.*

is degraded also increases.

5. A Local Mesh Refinement Procedure. In this section we present an algorithm for computing locally refined meshes of the types described in Sections 3 and 4 [6]. For quadrilateral elements, our algorithm implements the 1-irregular and 3-neighbor rules, and for triangles, it implements the 1-irregular and 2-neighbor rules.

Our scheme assumes that a logical-valued function $DVTEST$ is available which indicates, for a given element in the mesh, whether the element is to be refined. $DVTEST$ can either be a user specification of a fixed refinement pattern (say, by using element level numbers to control refinement) or be the output of a self-adaptive mechanism within the code which uses local error indicators. An element in the mesh may be refined either because $DVTEST$ indicates it should be refined, or because it violates the $k_e - 1$-neighbor rule or the 1-irregular rule. Note that a given element may satisfy both rules when it it is processed, but violate a rule later in the refinement process due to the refinement of one or more of its neighbors. Thus it is clear that we must examine certain elements more than once.

Procedure REFINE
{ **If** this is the first call to $REFINE$, **THEN** $[nt \leftarrow nt_0]$;
$i \leftarrow 1$;
While $(i \leq nt)$ **Do**
[**For** $j \leftarrow 1$ **to** k_e **Do**
[**If** τ_i^j is unrefined **Then**
[**If** τ_i^j has $k_e - 1$ or k_e refined neighbors **Then** $DIVIDE(\tau_i^j)$;
Else If $\ell_i > \ell_{\tau_i^j} + 1$ **Then** $DIVIDE(\tau_i^j)$]];
If $DVTEST(t_i)$ **Then** $DIVIDE(t_i)$;
$i \leftarrow i + 1$]};

Procedure DIVIDE
{ $s_i \leftarrow nt + 1$;
$nt \leftarrow nt + 4$;
create t_{s_i+j}, $0 \leq j \leq 3$, along with associated vertices};

Logical Function DVTEST
$\{\dots\}$

FIG. 5.1. *Procedure REFINE.*

The refinement procedure $REFINE$ is presented in Figure 5.1. Notice that

REFINE is a one-pass algorithm in which elements are processed in the order they are created; suppose the refinement of t_k is forced by the k_e-1-neighbor or 1-irregular rule because of the refinement of t_i (for whatever reason). *REFINE* learns this fact (and acts on it) by testing the *neighbors* of an element against the k_e-1-neighbor and 1-irregular criteria before the element itself is tested for refinement in *DVTEST*.

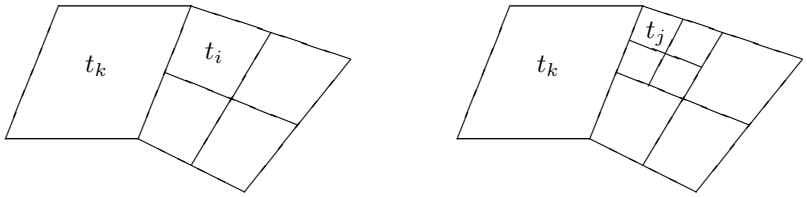


FIG. 5.2. *Refinement of t_i.*

For example, in the situation depicted in Figure 5.2, when t_i is refined, its four sons are added to the end of the element list. When son t_j is processed, its neighbor t_k violates the 1-irregular rule, so t_k is refined and *it's* sons are added to the end of the element list. If the refinement of t_k causes new violations of the 1-irregular or k_e-1-neighbor rules, this will be discovered and remedied when the sons of t_k are processed.

Since a given element has at most k_e neighbors, we test (and possibly refine) at most k_e+1 elements for any reason at any step of the process. We will show that generating the data to carry out any given step of *REFINE* requires constant time. Thus, the complexity of *REFINE* is linear in the number of elements.

Since the 2-neighbor and 1-irregular rules are satisfied by triangular meshes generated in *REFINE*, each remaining irregular vertex is the sole edge midpoint in some unrefined element. Each such element is refined by *GREEN* into two triangles.

Note that procedure *REFINE* can be reentered using the mesh generated by a previous call to *REFINE*. (This requires setting $nt \leftarrow nt_0$ outside *REFINE*.) It is thus easy to incorporate *REFINE* into adaptive refinement schemes in which mesh generation is alternated with equation solution. Information regarding which elements are to be refined at a given stage is passed to *REFINE* via *DVTEST*. In the case of triangular elements, green triangles are logically removed from the mesh before invoking *REFINE*, in order to avoid the possibility of creating new triangles with excessively small angles.

6. Data Structures and Implementation. In order to carry out procedure *REFINE* and the other procedures normally associated with finite element calculations, we must be able to generate and retrieve certain data about elements and vertices in the mesh. We describe this data below in the form of functions.

For a given element t_i, we must be able to compute:

F1. **knots:**

$$knots(j,i) = k \quad \text{where } \nu_i^j = v_k,\ 1 \le j \le k_e$$

F2. **neighbors:**

$$n(j,i) \left\{ \begin{array}{rl} = \tau_i^j & \text{if } \varepsilon_i^j \text{ is an interior edge} \\ < 0 & \text{if } \varepsilon_i^j \text{ is a boundary edge} \end{array} \right\}, \quad 1 \le j \le k_e$$

F3. **father:**

$$f(i) = k \quad \text{where } t_k \text{ is the father of } t_i$$

F4. **sons:**

$$s(i) = \begin{cases} s_i & \text{if } t_i \text{ is refined} \\ 0 & \text{if } t_i \text{ is not refined} \end{cases}$$

F5. **level:**

$$\ell(i) = \ell_i$$

For a given vertex v_k we must be able to compute:

F6. **vertex type:**

$$vty(k) = \begin{cases} 1 & \text{if } v_k \text{ is a user boundary vertex} \\ 2 & \text{if } v_k \text{ is a user interior vertex} \\ 3 & \text{if } v_k \text{ is a regular boundary vertex} \\ 4 & \text{if } v_k \text{ is a regular interior edge vertex} \\ 5 & \text{if } v_k \text{ is a regular interior center vertex} \\ 6 & \text{if } v_k \text{ is irregular} \end{cases}$$

User vertices are those associated with $\mathcal{T}_0$ ($1 \leq k \leq nv_0$). Regular interior edge vertices are those regular interior vertices which were created as the midpoint of an element edge during the refinement process. Regular interior center vertices were created as the center vertex in the refinement of a quadrilateral element.

F7. **vertex fathers:**

$$vf(j,k) = m_j \quad \begin{array}{l} \text{if } v_k \text{ was obtained as the midpoint of the edge} \\ \text{with endpoints } v_{m_j},\ 1 \leq j \leq 2 \end{array}$$

Vertex fathers are well defined for vertices of types 3, 4, and 6. If v_k is a center vertex, it was created during the refinement of a particular quadrilateral element, and we can require its global vertex number k to be the largest of the nine vertices involved in that refinement step. The vertex fathers of v_k are then defined as an arbitrary pair of opposing edge midpoints in the refined element.

F8. **coordinates:**

$$\begin{aligned} vx(k) &= x\text{-coordinate of } v_k \\ vy(k) &= y\text{-coordinate of } v_k \end{aligned}$$

Since algorithm $REFINE$ relies on integer triangle adjacency information, it does not require the x and y coordinates of the vertices; they are required, however, for the finite element matrix assembly process.

The data structure associated with and created by $REFINE$ corresponds to a *refined element tree*,,and consists of four integer arrays: two short arrays $IUSR$ and

$IBOUND$, and two larger arrays IRE and $IVERT$. $IBOUND$ is an array of length $2 \times nb_0$, where nb_0 is the number of boundary edges in $\mathcal{T}_0$, and contains information about the boundary conditions. $IUSR$ is a $2\,k_e \times nt_0$ array whose i-th column contains $knots(j,i)$ and $n(j,i)$, $1 \le j \le k_e$, for macro element t_i.

Let $maxv$ (respectively $maxt$) be the total number of vertices (respectively elements) contained in the finest mesh generated. A refined triangular mesh with $maxv$ vertices will have $maxt \sim 8/3\,maxv$ refined and unrefined elements, while a quadrilateral mesh will have $maxt \sim 4/3\,maxv$ total elements. Then IRE will be of size $k_e \times maxt$ and $IVERT$ will be of size $2 \times maxv$. $2 \times maxv$ real words of storage will be required for the x and y coordinates in the matrix assembly process.

FIG. 6.1. *Macro Element Nodes in IRE; Quadrilaterals (left) and Triangles (right).*

Suppose $nt_0 = 3 \bmod 4$. (If not, renumber the macro elements starting at 2, 3, or 4 rather than 1 to accomplish this.) Columns $1 - nt_0$ of IRE contain information about macro elements, as indicated in Figure 6.1. The remaining storage in IRE is devoted to refined elements. If t_{i+j}, $0 \le j \le 3$ form a quartet of sons of a regularly refined element, then columns $i-(i+3)$ of IRE contain information about the quartet as a whole. Because $nt_0 = 3 \bmod 4$, given t_k with $k > nt_0$, the decomposition

$$\begin{aligned} i &= k - k \bmod 4 \\ j &= k \bmod 4 \end{aligned} \tag{6.1}$$

both gives the starting column index i of the block of storage allocated to the quartet, and establishes that t_k is the j-th son of $t_{f(k)}$.

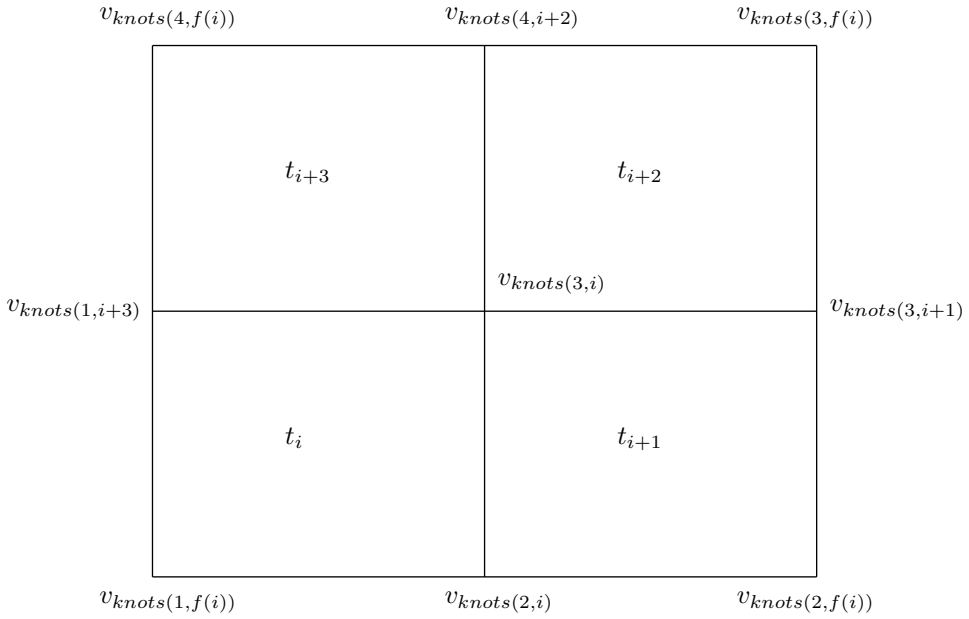
If t_{i+j}, $0 \le j \le 3$, form a quartet of quadrilaterals, columns $i - (i+3)$ of IRE contain the information indicated in Figure 6.2. If the four elements form a quartet of triangles, the the corresponding columns of IRE contain the information indicated in Figure 6.3. The information contained in both IRE blocks is the same, except that nine global vertex numbers are stored for quadrilaterals, while only six are needed for triangles.

In the case of triangular elements, green triangles can be appended to the end of the IRE array. The son field of the father should be set to point at the storage block allocated to the pair, and given a negative value to distinguish green triangle sons from regular triangle sons. For each pair of green triangles, we must store four global vertex numbers and the father of the pair, so two columns of IRE are sufficient to contain all the required data for the pair.

Information about vertices is stored in $IVERT$, with column k corresponding to vertex v_k. The type of vertex information varies according to vertex type, as illustrated in Figure 6.4.

Here IBC is simply a pointer into $IBOUND$ that allows determination of the appropriate boundary conditions. The $IVF1$, $IVF2$ and F fields require some explanation, since they contain the key adjacency information about the mesh.

When a new vertex v_k is created in the interior of Ω, the situation is typically one of those shown in Figure 6.5 for triangular elements. In both cases, the creation

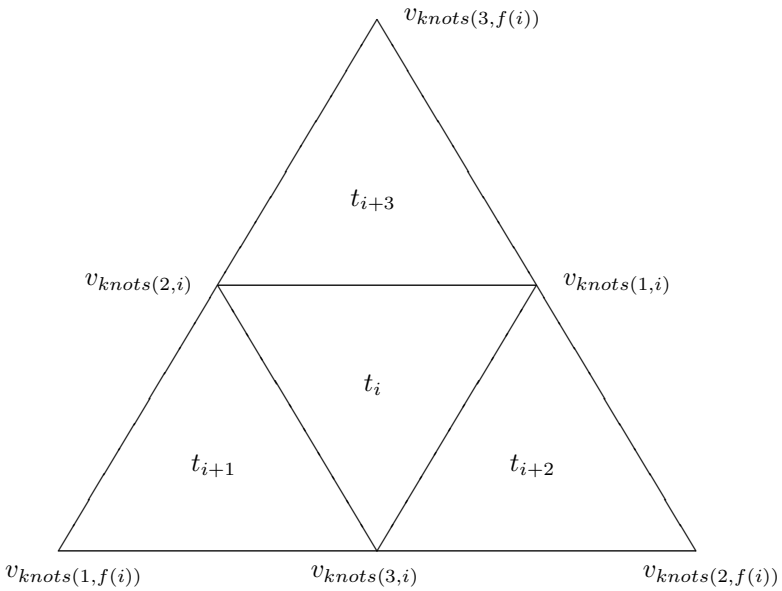


	i	$i+1$	$i+2$	$i+3$
1	$f(i)$	$\ell(i)$	$-$	$knots(3,i)$
2	$knots(2,i)$	$knots(3,i+1)$	$knots(4,i+2)$	$knots(1,i+3)$
3	$knots(1,f(i))$	$knots(2,f(i))$	$knots(3,f(i))$	$knots(4,f(i))$
4	$s(i)$	$s(i+1)$	$s(i+2)$	$s(i+3)$

FIG. 6.2. *Refined Quadrilateral Node in IRE.*

of v_k is caused by the creation of the quartet of regular elements t_{i+j}, $0 \le j \le 3$, $i = 0 \bmod 4$. Since v_k is irregular F is set to m, and $IVF1$ is set to $i+j$, where v_k is the midpoint of edge $\varepsilon^j_{f(i)}$ ($\varepsilon^{j+1}_{f(i)}$ for quadrilateral elements) of element $t_{f(i)}$. A decomposition of $IVF1$ as in (6.1) give the address of the storage block for the quartet in IRE, and establishes the geometric relationship of v_k to the quartet. If the situation is as in Figure 6.5a, then F is updated when t_m is refined, but v_k remains irregular. If the situation is as in Figure 6.5b, v_k will become regular if element t_m is refined. In this case the $IVF2$ field in the same way as $IVF1$, but for the second quartet. If v_k is a center node in the refinement of a quadrilateral into elements t_{i+j}, $0 \le j \le 3$, then the $IVF1$ field is set to i, since the geometric relationship of v_k to t_{i+j} is established by its vertex type.

The actual refinement process consists of filling in entries in IRE and $IVERT$. When an element is refined, all the information required for IRE is trivially known, except for the global vertex numbers for the edge midpoints. (The center node in the quadrilateral case must be created, so its global vertex number is known.) For each edge, we must either create new irregular or (regular) boundary vertices, or change current irregular vertices into regular ones. To do this, we must have the ability to compute $knots(j,i)$, $n(j,i)$, $1 \le j \le k_e$, $f(i)$, $s(i)$, and $\ell(i)$ for any element currently in the mesh. When a new vertex is created or changes status, all the necessary information is present, having been generated in the process of determining whether or not an irregular vertex already existed on the edge.



	i	$i+1$	$i+2$	$i+3$
1	$f(i)$	$knots(1,i)$	$knots(2,i)$	$knots(3,i)$
2	$\ell(i)$	$knots(1,f(i))$	$knots(2,f(i))$	$knots(3,f(i))$
3	$s(i)$	$s(i+1)$	$s(i+2)$	$s(i+3)$

FIG. 6.3. *Refined Triangle Node in IRE.*

$vty=1$	$vty=2$	$vty=3$	$vty=4$	$vty=5$	$vty=6$
0	0	$IVF1$	$IVF1$	$IVF1$	$-IVF1$
$-IBC$	0	$-IBC$	$IVF2$	0	F

FIG. 6.4. *Nodes in IVERT.*

We now discuss briefly the computation of functions F1-F8. If $1 \le i \le nt_0$, $knots(j,i)$, $n(j,i)$ and $s(i)$ are found by table look-up in $IUSR$ and IRE. $f(i)$ and $\ell(i)$ are trivially known. For $i > nt_0$, one makes the decomposition (6.1); then $knots(j,i)$, $s(i)$, $f(i)$ and $\ell(i)$ can be found by table look-up. Finding neighbors is more complicated.

For triangles, if $i = 0 \bmod 4$ (the center element) then $n(j,i)$ are trivially known. If $i \neq 0 \bmod 4$, the one neighbor (the center element) is known. The other two are found by checking the status of the two edge midpoint vertices of $t_{f(i)}$ which are corners of t_i. If such a vertex is a regular boundary vertex (type 3) or an irregular vertex (type 6), then $n(j,i)$ is found in the IBC or F fields, respectively, of $IVERT$. If the vertex is a regular interior vertex (type 4), then both $IVF1$ and $IVF2$ are analyzed as in (6.1). One of $IVF1$ or $IVF2$ points to the quartet containing t_i, and the other points to the neighboring quartet. Since we know how this vertex is geometrically related to each quartet, we can easily determine how the two quartets are related. For quadrilateral elements, two neighbors are known, and the other two can be determined by a procedure analogous to that for triangular elements.

For a given vertex v_k, its vertex type can be determined by checking the signs of

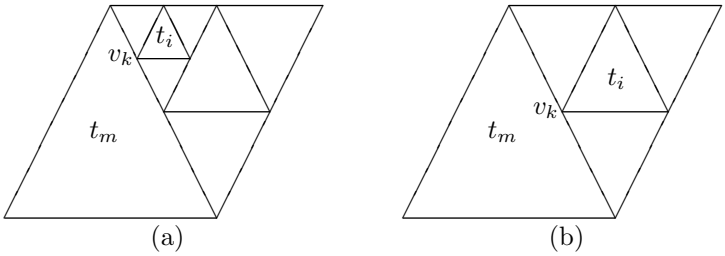


FIG. 6.5. *Creation of Vertex v_k.*

its two entries in $IVERT$. If $k > nv_0$, the vertex fathers are found by decomposing $IVF1$ as in (6.1). Once the geometric relationship of the vertex to the quartet pointed at by $IVF1$ is known, the vertex fathers can be looked up in IRE.

Thus, each piece of data required by algorithm $REFINE$ can be obtained in constant time. Another data structure which can be applied to arbitrary admissible meshes, but which does not obtain data in constant time, is presented in [10].

The finite element assembly procedure in addition requires the actual x and y coordinates of the vertices. These are easily generated from the vertex fathers when $k > nv_0$ using

$$vx(k) = \frac{vx(vf(1,k)) + vx(vf(2,k))}{2}$$
$$vy(k) = \frac{vy(vf(1,k)) + vy(vf(2,k))}{2}$$

Since $vf(j,k) < k$ for $k > nv_0$, $j = 1, 2$, and the x and y coordinates for the vertices in $\mathcal{T}_0$ are given, the remaining coordinates can be generated in one pass through the vertices in the order in which they were created.

We conclude with several remarks on storage requirements. For a given mesh with $maxv$ vertices, a standard data structure for representing the finite element mesh is a list of pointers from unrefined elements to adjacent vertices and a list of pointers from vertices to adjacent unrefined elements [11]. For triangular elements, the number of unrefined elements is about $2\,maxv$, so the standard data structure requires about $12\,maxv$ storage. IRE and $IVERT$ require about $10\,maxv$ storage for the same mesh. For quadrilateral meshes the number of unrefined elements is about $maxv$, so the standard data structure requires about $8\,maxv$ storage, as opposed to $7\,1/3\,maxv$ storage for IRE and $IVERT$. Hence our data structures, which allow for local mesh refinement, actually require less storage than standard data structures for static meshes.

7. Appendix. Proofs of properties T6, T7 and the equivalence of Q4 and Q8.

We now prove property T6:

> *There are fewer than 13 times as many unrefined elements in the triangular mesh $\mathcal{T}'$ as there are in $\mathcal{T}$.*

Proof. Since $\mathcal{T}_0$ consists of one triangle, the number of unrefined elements in either $\mathcal{T}_0$ or $\mathcal{T}'$ is three times the number of refined elements plus one. Since $\mathcal{T}$ is admissible, each element t shares a corner with at most 13 different elements of the same level as t, so it suffices to show that

A1. Each refined element $t' \in \mathcal{T}'$ shares a corner with a refined element $t \in \mathcal{T}$. with the level ℓ of t equal to the the level ℓ' of t'.

A1 follows by induction on ℓ'. Let $\ell_{\max}$ be the largest level of an element in $\mathcal{T}'$. If $\ell' = \ell_{\max} - 1$ then t' must be in $\mathcal{T}$, so let $t = t'$. Otherwise if $t' \notin \mathcal{T}$, t' was forced to refine by the 1-irregular rule, so there must be an element s' sharing part of a side with t', with the level of s' equal to $\ell' + 1$. By induction, there is a refined element $s \in \mathcal{T}$ sharing a corner with s' with level $\ell' + 1$. Then t, the father of s, must satisfy A1 (Figure 7.1). The proof is very similar to the proof of property Q6 given in [14]. □

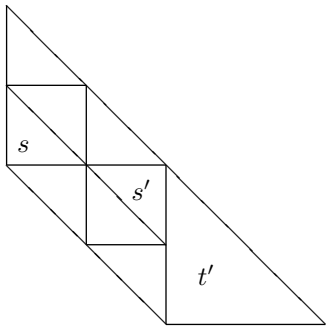


FIG. 7.1. *Proof of Property T6.*

We now prove property T7:

> *The regular triangular mesh vertices $\{v_i'\}$ for the mesh $\mathcal{T}''$ can be partitioned into mutually disjoint sets $V_1, V_2, \ldots, V_9$ such that for any distinct v_i' and v_j' in the same set V_k, $supp(b_i') \cap supp(b_j')$ contains no elements.*

Proof. For simplicity, we consider the case in which $\mathcal{T}_0$ consists of the triangle with vertices $(0,0)$, $(0,1)$ and $(1,0)$. Let $\ell_{\min}(i)$ denote the minimum level of an unrefined element in $\mathcal{T}'$ for which vertex v_i is a corner. Let

$$VL^\ell = \{v_i \,|\, \ell_{\min}(i) = \ell\}.$$

Since $\mathcal{T}_0$ consists of one triangle, at most six edges in $\mathcal{T}'$ meet at v_i so b_i is not nonzero in any element in $\mathcal{T}'$ at level $\ell_{\min}(i)+3$ or greater. (Figure 7.2 depicts the worst case: $\ell_{\min}(i) = \ell_k - 2$.)

Recalling the coordinates of v_i are (x_i, y_i), let

$$\begin{aligned} V^0 &= \{v_i \,|\, (x_i - y_i)2^{\ell_i - 1} = 0 \bmod 3\} \\ V^1 &= \{v_i \,|\, (x_i - y_i)2^{\ell_i - 1} = 1 \bmod 3\} \\ V^2 &= \{v_i \,|\, (x_i - y_i)2^{\ell_i - 1} = 2 \bmod 3\}. \end{aligned}$$

The vertices in sets V^0, V^1, and V^2 when $\ell_i = 3$ are labeled 0, 1, and 2, respectively, in Figure 7.3. By construction, $\text{supp}(b_i) \cap \text{supp}(b_j)$ contains no elements for distinct $v_i, v_j \in V^k \cap VL^\ell$. Thus we can set

$$V_{\ell+3k} = (VL^\ell \cup VL^{\ell+3} \cup VL^{\ell+6} \ldots) \cap V^k$$

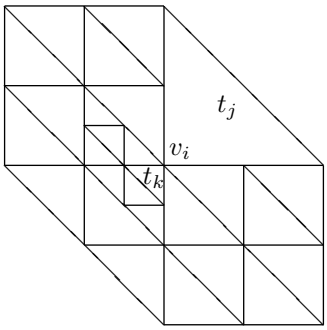


FIG. 7.2. $\ell_{\min}(i) = \ell(t_j)$.

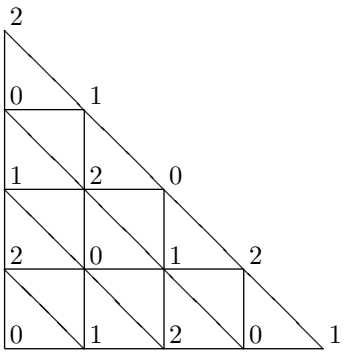


FIG. 7.3. V^0, V^1, *and* V^2.

for $\ell = 1, 2, 3$, $k = 0, 1, 2$. A similar construction shows property Q7. Note that $V_1, \ldots, V_9$ can be constructed in a simple one pass algorithm processing vertices. Finding fewer sets which satisfy property T7 is possible (since $\mathcal{T}''$ is planar, four sets suffice, but algorithms for constructing such sets may be more expensive. □

We now outline a proof of equivalence of Q4 and Q8, i.e.,

A2. In each element t in an admissible quadrilateral mesh $\mathcal{T}$, the restrictions to t of the b_i which are nonzero in t are linearly independent

holds if and only if

A3. No irregular vertex v in $\mathcal{T}$ touches unrefined elements t_1, t_2, and t_3 with levels $\ell_1 < \ell_2 < \ell_3$.

Proof. First, suppose that A3 does not hold: there exists a vertex v touching t_1, t_2, and t_3 of different levels. Suppose t_1 is the largest element in $\mathcal{T}$ touching such a vertex. Each straight line segment (consisting of one or more edges) in $\mathcal{T}$ touches a regular vertex, so we can assume that v_3, v_4, and v_5 are chosen to be the first regular vertices away from t on their straight line segments. Figure 7.4 depicts this situation, where v_3, v_4, and v_5 may, in fact, be corners of t, t_3, and t_2 respectively. Since $\ell_2 > \ell_1$ and $\ell_3 > \ell_1$, corners v_1 and v_2 of t_1 must be regular as well. It follows that b_1, b_2, b_3, b_4, and b_5 must all be nonzero in element t. Since their restrictions to t are bilinear functions, A2 cannot hold.

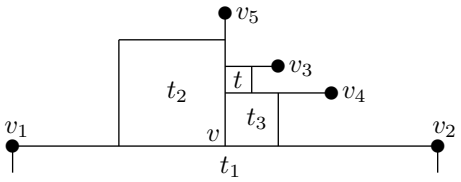


FIG. 7.4. *Condition A3 Violated.*

Now suppose A3 holds. Since $\{b_i\}$ is a basis for $\mathcal{S}$, to show A2, it suffices to show that there are at most four b_i which are nonzero in any element t. Suppose that t has four regular corners v_1, v_2, v_3, and v_4. Then b_1, b_2, b_3, and b_4 are nonzero in t, and all other b_i must be zero by the Lagrange conditions and continuity. Suppose t has exactly one irregular corner v_4. Then v_4 must be on the side of a neighbor t_2 of t, connecting corners v_3 and v_5 of t_2, with ℓ_2 less than the level of t. It follows that v_3 and v_5 are regular vertices, as are the other corners v_1 and v_2 of t. Then b_1, b_2, b_3, and b_5 are nonzero in t, and all other b_i must be zero in t (Figure 7.5). The case when t has exactly two irregular corners is handled similarly. Note that t cannot have three irregular corners. Thus A2 holds in every case. □

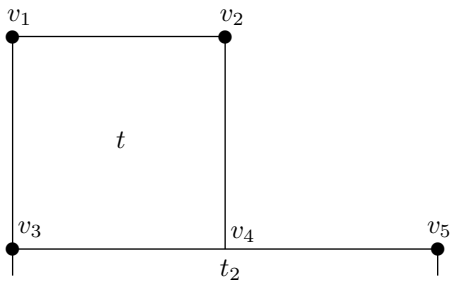


FIG. 7.5. *t Has One Irregular Corner.*

Acknowledgments: We are grateful to the management of Exxon Production Research Company for permission to publish this paper, and to Marie Mason of EPR for efficiently typing it. This research was supported in part by the Office of Naval Research through contracts N00014-80-C-0645 (University of Texas at Austin), N00014-82-K-0197 (University of California at San Diego), and N00014-76-C-0277 (Yale University).

REFERENCES

[1] I. BABUŠKA AND M. R. DORR, *Error esitmates for the combined h and p versions of the finite element method*, Tech. Rep. BN-951, Institute for Physical Science and Technology, University of Maryland, 1980.

[2] I. BABUŠKA AND A. MILLER, *A posteriori error esitmates and adaptive techniques for the finite element method*, Tech. Rep. BN-968, Institute for Physical Science and Technology, University of Maryland, 1981.

[3] I. BABUŠKA AND W. C. RHEINBOLDT, *Error estimates for aadptive finite element computations*, SIAM J. Numer. Anal., 15 (1978), pp. 736–754.

[4] ———, *Reliable error estimation and mesh adaptation for the finite element method*, in Computational Methods in Nonlinear Mechanics, North-Holland, New York, 1980, pp. 67–108.

[5] R. E. BANK AND A. H. SHERMAN, *A multilevel iterative method for solving finite element*

equations, in Proc. Fifth Sympos. on Reservoir Engineering, Society of Petroleum Engineers of AIME, Dallas, 1979, pp. 117–126.

[6] ———, *A refinement algorithm and dynamic data structure for finite element meshes*, Tech. Rep. CNA-166, Center for Numerical Analysis, University of Texas at Austin, 1980.

[7] R. E. Bank, A. H. Sherman, and A. Weiser, *On the regularity of local mesh refinement*, in Proceedings of the IMACS Tenth World Conference, 1982, pp. 61–64.

[8] R. E. Bank and A. Weiser, *Some a posteriori error estimators for elliptic partial differential equations*, Mathematics of Computation, 44 (1985), pp. 283–301.

[9] D. B. Gannon, *Self Adaptive Methods for Parabolic Differential Equations*, PhD thesis, University of Illinois, 1980.

[10] W. C. Rheinboldt and C. K. Mesztenyi, *On a data structure for adaptive finite element mesh refinements*, ACM Trans. Math. Software, 6 (1980), pp. 166–187.

[11] R. B. Simpson, *A survery of two dimensional finite element mesh generation*, in Ninth Manitoba Conference on Numerical Mathematics and Computing, Manitoba, 1979, pp. 49–124.

[12] G. Strang and G. Fix, *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, New Jersey, 1973.

[13] J. R. Van Rosendale, *Rapid Solution of Finite Eelement Equations on Locally Refined Grids by Multi-Level Methods*, PhD thesis, University of Illinois, 1980.

[14] A. Weiser, *Local-mesh, local-order, adaptive finite element methods with a posteriori error estimators*, Tech. Rep. 213, Computer Science Department, Yale University, 1981.