AN ALGORITHM FOR COARSENING UNSTRUCTURED MESHES

RANDOLPH E. BANK* AND JINCHAO XU†

Abstract. We develop and analyze a procedure for creating a hierarchical basis of continuous piecewise linear polynomials on an arbitrary, unstructured, nonuniform triangular mesh. Using these hierarchical basis functions, we are able to define and analyze corresponding iterative methods for solving the linear systems arising from finite element discretizations of elliptic partial differential equations. We show that such iterative methods perform as well as those developed for the usual case of structured, locally refined meshes. In particular, we show that the generalized condition numbers for such iterative methods are of order J^2, where J is the number of hierarchical basis levels.

Key words. Finite element, hierarchical basis, multigrid, unstructured mesh.

AMS subject classifications. 65F10, 65N20

1. Introduction. Iterative methods using the hierarchical basis decomposition have proved to be among the most robust for solving broad classes of elliptic partial differential equations, especially the large systems arising in conjunction with adaptive local mesh refinement techniques [5][2]; they have been shown to be strongly connected to space decomposition methods and to classical multigrid methods [27][28][4][17]. Classical hierarchical basis and multigrid methods are defined in terms of an underlying refinement structure of a sequence of nested meshes. In many cases this is no disadvantage, but it limits the applicability of the methods to truly unstructured meshes, which may be highly nonuniform but *not* derived from some grid refinement process. In this work, we develop and analyze an algorithm for generating a hierarchical basis for the continuous piecewise linear finite element space corresponding to an arbitrary, nonuniform, unstructured triangulation of a region Ω. This allows us to extend the hierarchical basis and other related iterative methods in a natural way to such meshes. Some work on multigrid methods on non-nested meshes is reported in Bramble, Pasciak and Xu [9], Xu [27], Zhang [31], Chan and Smith [11], Mavriplis [21], Kornhuber [20], Hoppe and Kornhuber [19], and Bank and Xu [6] [7].

The practical algorithm for unrefining an arbitrary mesh is developed in Sections 2 and 3. Our basic idea is to force an arbitrary unstructured mesh into the mold of a possibly nonuniform, locally refined mesh. In doing this, we impose some logical structure on the mesh, and this logical structure will admit the creation of a hierarchical basis. The logical structure itself is recovered from the fine mesh by a coarsening algorithm.

In Section 2, we develop a general refinement paradigm allowing the refinement of a given element into $2-4$ child elements in a number of different patterns. The critical point of this refinement process is that new vertices created during the refinement of an element need not lie on the boundary of that element, but can be moved some distance from the boundary, either into the strict interior of the element, or into the interior of a neighboring element. This allows the sequence of refined meshes to become physically nonnested, but still retain an underlying logical refinement structure. Using this paradigm, in Section 3 we develop a practical algorithm for creating a logical

*Department of Mathematics, University of California at San Diego, La Jolla, CA 92093. The work of this author was supported by the Office of Naval Research under contract N00014-89J-1440.

†Department of Mathematics, Penn State University, University Park, PA 16802. The work of this author was supported by National Science Foundation.

refinement structure given only the final fine mesh. This is done by a process of coarsening, in which one vertex is removed at each step, and the mesh updated. We develop several heuristic rules to determine the ordering of vertices for elimination, and for updating the mesh during an elimination step.

The coarsening algorithm occasionally needs to be restarted, when it arrives at a mesh in which no further vertices can be eliminated within the constraints of the algorithm. The restart procedure is quite simple, and consists of triangulating certain quadrilateral elements which arise during intermediate steps of the elimination process, creating a mesh in which there are only triangular elements. The restart procedure implicitly corresponds to incorporating an *edge swapping* algorithm as part of the refinement procedure. Such procedures are quite common in adaptive refinement and mesh generation algorithms. In practical terms, this has little effect on our procedure. The fillin generated by the elimination process is modestly increased, and the graphs of the filled in matrices no longer coincide exactly with the triangular finite element meshes. There is no effect on the complexity of the iterative methods based on the hierarchical basis decomposition, and so far as we have observed, no significant effect on the observed rates of convergence of these methods. Unfortunately, in terms of our theory, we can allow edge swapping only on a fixed number of levels without severely weakening our estimates. We view this more as a shortcoming of our current theory than as a flaw in the basic approach.

In Section 4, we turn to the formal construction of the hierarchical basis functions, and the mathematical analysis of some of their important properties. The basis functions are developed using simple interpolation operators arising naturally from the refinement paradigm developed in Section 2. On an intuitive level, the success of multigrid and other hierarchical basis iterations derives from the fact that the supports of the hierarchical basis functions grow as one proceeds to coarser grid levels. This allows effective damping of smoother components of the error on the coarser levels. In the case of nested refinement, the hierarchical basis functions are in fact nodal basis functions for the coarser meshes. As one might expect, this greatly simplifies the mathematical analysis. In the current case, the hierarchical basis functions exhibit a similar growth in support. They have the appearance of perturbed (or "lumpy") nodal basis functions from the coarser meshes (see Figure 9), but mathematically remain fairly complicated linear combinations of fine grid nodal basis functions. Thus, while we should expect iterative methods based on such functions to behave similarly to those for nested meshes, the mathematical analysis is more complicated.

In Section 5 we present some iterative methods for solving elliptic partial differential equations using our hierarchical basis decompositions. Our main result is that the generalized condition number for these schemes grows as J^2, where J is the number of hierarchical basis levels. This is essentially the same result as for the classical hierarchical basis schemes. [1] [4] [28] [29] [30] [23] [24] [8]. We can also define regular multigrid and BPX like methods based on the hierarchical basis decompositions. These methods are based on the observation of Griebel [15] [14] and Griebel and Oswald [16], who show that such methods can be interpreted as standard block iterations applied to a larger, singular system of equations.

Finally, in Section 6, we give some numerical illustrations and provide some details of the implementation. Our methods have essentially the same implementation as their counterparts for the case of nested meshes. This is because, once the refinement structure, in particular the *vertex parents* function, is known, and the interpolation coefficients have been determined, the remainder of the implementation is completely

algebraic in nature [6].

2. A Paradigm for Logically Structured Grid Refinement. In this section we will develop a general model for the logically structured refinement of a triangular mesh. This model will serve as the basis for the coarsening algorithm for unstructured meshes to be developed in the next section.

Let us assume that we are given a coarse triangulation $\mathcal{T}_1$. We will let $\mathcal{X}_1$ and $\mathcal{E}_1$ denote the sets of vertices and element edges in $\mathcal{T}_1$. The edges and vertices of $\mathcal{T}_1$ are all given the unique level number of 1. Triangles can also be assigned levels, but these are not necessary for this model.

We assume that a sequence of meshes $\mathcal{T}_k$, with vertex sets $\mathcal{X}_k$ and edge sets $\mathcal{E}_k$ are generated inductively for $k > 1$ as follows:

Algorithm Refine

1. A certain subset of edges $\mathcal{F}_{k-1} \subseteq \mathcal{E}_{k-1}$ is chosen for refinement. Without loss of generality, we require that all edges in $\mathcal{F}_{k-1}$ be of level $k-1$.
2. The initial form of the new triangulation triangulation $\mathcal{T}_k'$ is created by bisecting each edge in the set $\mathcal{F}_{k-1}$ and then creating new elements as required. The form of the descendent triangles for a given triangle $t \in \mathcal{T}_{k-1}$ depends on the number of edges of t that were refined (see Figure 1).
3. The vertex set $\mathcal{X}_k'$ is composed as the union of $\mathcal{X}_{k-1}$ and the set $\tilde{\mathcal{X}}_k'$ consisting of the midpoints of the edges in $\mathcal{F}_{k-1}$. The newly created vertices (those in $\tilde{\mathcal{X}}_k'$) are assigned level number k.
4. The edge set $\mathcal{E}_k'$ consists of the union of three sets: those edges in $\mathcal{E}_{k-1}$ which are not in $\mathcal{F}_{k-1}$, the descendent edges arising from the refinement of the edges in $\mathcal{F}_{k-1}$, and finally the new edges added in creating the descendent triangles in $\mathcal{T}_k'$. The edges in the two latter sets, which we denote by $\tilde{\mathcal{E}}_k'$, are assigned level k.
5. The final triangulation $\mathcal{T}_k$ is created by allowing each vertex in the set $\tilde{\mathcal{X}}_k'$ to move a small distance from the midpoint of the edge which it bisects. This movement is controlled by the parameters $\bar{\theta}$ and $\bar{\epsilon}$ as described below. The connectivity of the mesh remains the same as in $\mathcal{T}_k'$. This results in final vertex set $\mathcal{X}_k = \mathcal{X}_{k-1} \cup \tilde{\mathcal{X}}_k$, and edge set $\mathcal{E}_k = \{\mathcal{E}_{k-1} \setminus \mathcal{F}_{k-1}\} \cup \tilde{\mathcal{E}}_k$, where $\tilde{\mathcal{X}}_k$ ($\tilde{\mathcal{E}}_k$) is the set of perturbed vertices (edges) corresponding to $\tilde{\mathcal{X}}_k'$ ($\tilde{\mathcal{E}}_k'$).

We note that Steps $1-4$ above (with the identifications $\mathcal{T}_k' = \mathcal{T}_k$, $\tilde{\mathcal{X}}_k' = \tilde{\mathcal{X}}_k$, and $\tilde{\mathcal{E}}_k' = \tilde{\mathcal{E}}_k$) describe a refinement procedure which results in a sequence of physically and logically nested meshes. It is only in Step 5 that the triangulations become physically nonnested. However, the triangulations remain *logically* nested, in the sense that one can describe an element $t \in \mathcal{T}_{k-1}$ which gives rise to several refined elements in $\mathcal{T}_k$, although the union of those elements may not be t.

We shall begin by considering several aspects of the less complicated algorithm based only on Steps 1-4. First, we note that case C (D) in Figure 1 can be easily decomposed into two (three) substeps of type B, in which only a single new vertex is added. At all intermediate stages, t remains the union of two or more triangles, although some of the intermediate triangles are themselves refined and hence are not elements in $\mathcal{T}_k$. This is *not* true of case E; to decompose case E into three substeps in which only a single vertex is added, we must introduce several types of quadrilateral elements, as illustrated in Figure 2. In Figure 2 F, we have a degenerate quadrilateral, consisting of a triangle with one edge refined. In Figure 2 G, with two edges refined, we have one triangular element, which remains a triangle in $\mathcal{T}_k$, and one trapezoid.

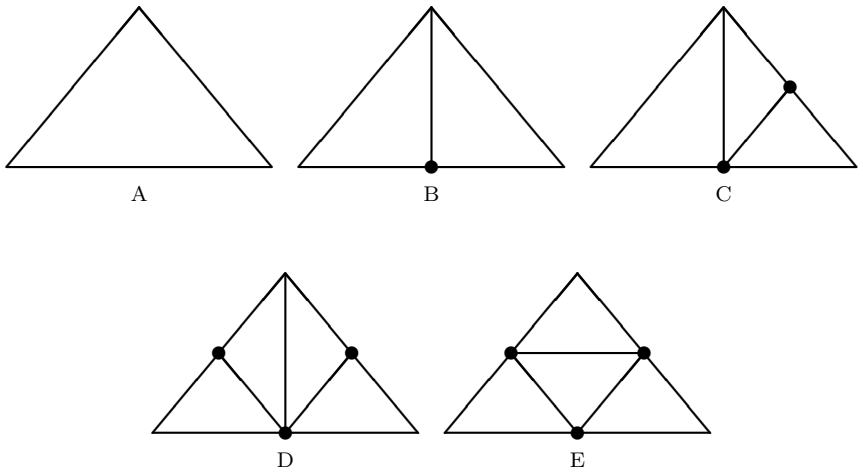

FIG. 1. *Possible refinements of an element $t \in \mathcal{T}_{k-1}$ depending on whether zero (A), one (B), two (C), or three (D-E) edges are bisected. All the newly created vertices are assigned level k.*

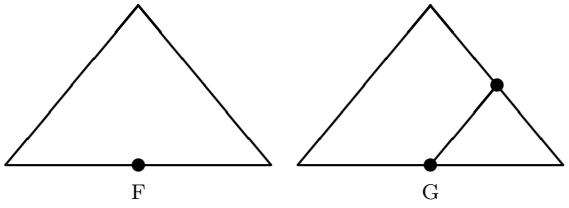

FIG. 2. *Quadrilateral elements may be generated during intermediate substeps of the refinement process.*

When the third edge is refined, the trapezoid becomes three triangles in $\mathcal{T}_k$, as in Figure 1 E.

We pause here in our discussion to make several remarks. First, we have assumed that only edges of level $k-1$ can be refined in creating $\mathcal{T}_k$ from $\mathcal{T}_{k-1}$. This condition is rarely imposed on practical refinement algorithms. However, for a given $\mathcal{T}_k$, typically one can construct a posteriori a sequence $\{\mathcal{T}_j\}_{j=1}^k$ which has this property.

Second, we implicitly assume that the refinement procedure has been devised in such a way that shape regularity of the triangulations $\mathcal{T}_k$ is guaranteed. Shape regularity concerns influence the selection of the subset $\mathcal{F}_{k-1}$ in Step 1. While there are several good strategies for controlling shape regularity during the refinement process [5] [2] [22] [25], their details do not concern us here.

Third, we are assuming an *edge* based model for refinement; it is possible to also have an *element* based model, where an element chosen for refinement is divided into three elements, a shown in Figure 3. While such refinement is possible, it is usually not desirable, as it tends to generate elements with poor shape regularity. As a practical matter, our coarsening algorithm must take this possibility into account (as a special case), but in this work we will ignore this possibility to simplify our discussion as

much as possible.

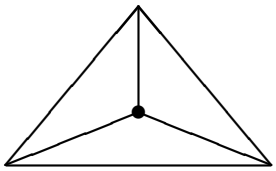

FIG. 3. *Element based refinement.*

Finally our refinement model does not allow the possibility of dynamic *edge swapping* (see Figure 4) a common technique for improving the shape regularity of a mesh. Such a procedure could result in a sequence of meshes which are not nested in either the physical or logical sense. While our goal is to coarsen unstructured meshes, creating a sequence of generally nonnested triangulations, our algorithms try to impose a *logical* nesting of the meshes while allowing them to be *physically* nonested. We note that our practical algorithm occasionally reaches a point where dynamic edge swapping (in a disguised form) is required in order for the coarsening to continue. While edge swapping in such circumstances does not significantly influence the observed rate of convergence of the resulting algebraic hierarchical basis multigrid method, the abstract theory presented in this manuscript can only handle edge swapping on a fixed number of levels. We also note that by allowing both cases D and E in Figure 1, which differ only by a swapped edge, we do permit limited a priori (static) edge swapping.

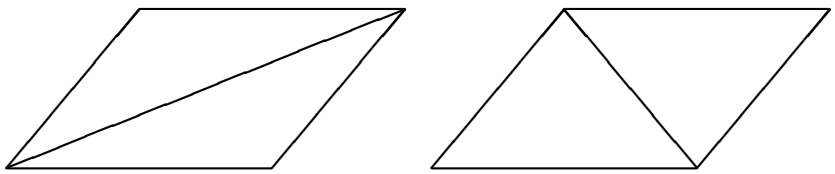

FIG. 4. *An example of edge swapping.*

We now turn to a discussion of Step 5 of Algorithm Refine. Let vertex $v' \in \tilde{\mathcal{X}}_k'$, $v' = (x', y')^t$, be the midpoint of an edge $e \in \mathcal{F}_{k-1}$. Let $v_r \in \mathcal{X}_{k-1}$, $v_r = (x_r, y_r)^t$, and $v_\ell \in \mathcal{X}_{k-1}$, $v_\ell = (x_\ell, y_\ell)^t$, denote the two endpoints of e. We call the vertices v_r and v_ℓ the *vertex parents* or just *parents* of vertex v' (see Figure 5). We note that one of the parents of v' must be level $k-1$, while the other could be any level between 1 and $k-1$. Since v' is the midpoint of e, we must also have $v' = (v_\ell + v_r)/2$. The vertex $v \in \tilde{\mathcal{X}}_k$, $v = (x, y)^t$, is defined in terms of the parents by

$$
\begin{aligned}
v &= \begin{bmatrix} x \\ y \end{bmatrix} \\
&= \theta \begin{bmatrix} x_r \\ y_r \end{bmatrix} + (1-\theta) \begin{bmatrix} x_\ell \\ y_\ell \end{bmatrix} + \epsilon \begin{bmatrix} y_\ell - y_r \\ x_r - x_\ell \end{bmatrix} \qquad (1)
\end{aligned}
$$

$$= \theta v_r + (1-\theta) v_\ell + \epsilon w$$

where $w^t(v_r - v_\ell) = 0$, $\| w \| \equiv \sqrt{w^t w} = \| v_r - v_\ell \|$. The parameter θ measures the displacement along the tangent direction to e and must satisfy

$$\bar{\theta} \leq \theta \leq 1 - \bar{\theta}. \tag{2}$$

The parameter ϵ measures the displacement in the normal direction to e and must satisfy

$$|\epsilon| \leq \bar{\epsilon}. \tag{3}$$

Here $\bar{\theta}$ and $\bar{\epsilon}$ are the two control parameters mentioned in Step 5 of the algorithm. The boundary case $\bar{\theta} = 1/2$, $\bar{\epsilon} = 0$ forces $\tilde{\mathcal{X}}'_k \equiv \tilde{\mathcal{X}}_k$ and the resulting meshes will be those generated by just Steps 1-4. The case $\bar{\epsilon} = 0$ forces the triangulations to remain physically as well as logically nested. An example illustrating one possible perturbation for a refinement of the form shown in Figure 1 B is shown in Figure 5.

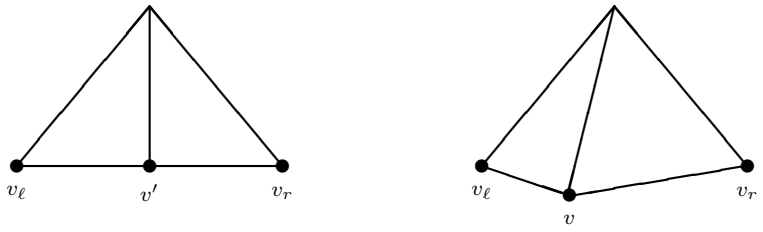

FIG. 5. *Some refined elements in $\mathcal{T}'_k$ on the left are perturbed by moving the newly created vertex v'. The resulting nonnested elements in $\mathcal{T}_k$ are shown on the right. Note that the connectivity is unchanged, so the triangulation remains* logically *nested.*

3. Details of the Coarsening Algorithm. In this section we describe a heuristic algorithm for coarsening a given fine, unstructured mesh. The basic idea is that we assume that the mesh came from the refinement procedure described in Section 2, and then attempt to recover the logical refinement structure from the fine mesh. The overall algorithm has a strong connection with a symbolic incomplete LU factorization of the nodal stiffness matrix [18], a point which is explored in detail in [6].

In our algorithm, we begin with the fine triangulation (graph) $\mathcal{T}_J$, and vertex set $\mathcal{X}_J$, and through a process of incomplete symbolic Gaussian Elimination (i.e., removing one vertex at a time), gradually transform $\mathcal{T}_J$ to the coarse grid triangulation $\mathcal{T}_1$, with vertex set $\mathcal{X}_1$. At intermediate steps, the graph will be a triangulation composed only of triangular and quadrilateral elements, since these are the only types of elements which arise in our edge based refinement paradigm. As usual, the intersection of two distinct elements in any graph will be either the empty set, a shared vertex or a shared edge. All intermediate graphs may not be appropriate finite element meshes, however, since we must allow nonconvex quadrilaterals (see Figure 2 F).

We begin with some notation, definitions and descriptions which we need to describe the coarsening algorithm. See Rose [26] and George and Liu [13] for a complete discussion of the connection of graph theory and Gaussian elimination. To keep the notation as simple as possible, we will drop subscripts unless absolutely necessary. Let $\mathcal{T}$ be a graph (triangulation), with vertex set $\mathcal{X}$ and edge set $\mathcal{E}$. Then for a vertex

$v \in \mathcal{X}$, we define the edge adjacency set $\text{adj}^e_{\mathcal{T}}(v)$ to be the set of vertices in $\mathcal{X}$ which is connected to v by an edge in $\mathcal{E}$ ($w \in \text{adj}^e_{\mathcal{T}}(v) \rightarrow e \equiv (v, w) \in \mathcal{E}$). The set $\text{adj}^q_{\mathcal{T}}(v)$ (the quadrilateral adjacency set) is nonempty only if v is a vertex of some quadrilateral elements in $\mathcal{T}$. In this case, for each quadrilateral having v as a vertex, two of the remaining vertices are elements of the set $\text{adj}^e_{\mathcal{T}}(v)$; the third (the vertex "opposite" to v and not connected by an edge in $\mathcal{E}$) will be in the set $\text{adj}^q_{\mathcal{T}}(v)$. Finally, we set

$$\text{adj}_{\mathcal{T}}(v) = \text{adj}^e_{\mathcal{T}}(v) \cup \text{adj}^q_{\mathcal{T}}(v)$$

and denote by $\deg^e_{\mathcal{T}}(v)$, $\deg^q_{\mathcal{T}}(v)$, and $\deg_{\mathcal{T}}(v) = \deg^e_{\mathcal{T}}(v) + \deg^q_{\mathcal{T}}(v)$ the sizes of $\text{adj}^e_{\mathcal{T}}(v)$, $\text{adj}^q_{\mathcal{T}}(v)$, and $\text{adj}_{\mathcal{T}}(v)$, respectively.

We next describe the classification of the vertices in the set $\mathcal{X}$. We assume that $\mathcal{X}$ can be decomposed at the union of four nonoverlapping sets

$$\mathcal{X} = \mathcal{X}^c \cup \mathcal{X}^b \cup \mathcal{X}^i \cup \mathcal{X}^0.$$

The purpose of this decomposition is to allow the coarsening algorithm to preserve (approximately) important properties of the fine mesh such as the shape of the physical domain, the locations of internal interfaces, points where boundary conditions change type, etc.

Vertices in the set $\mathcal{X}^c$ are called *corners*. Such vertices are known a priori to be members of the coarse grid, and include such vertices as actual geometric corners of the region, boundary points where boundary conditions change type, vertices marking the junction of internal interfaces or the junction of an internal interface and the boundary, and any other vertices one wants in the coarse grid for any reason. The set $\mathcal{X}^b$ are the *boundary* vertices, and consists of the remaining vertices on the boundary of the mesh exclusive of those classified as corners. A vertex $v \in \mathcal{X}^b$ implies that v is an endpoint of exactly two boundary edges, and that v is not an endpoint of any edge on an internal interface; vertices failing to meet this requirement must be classified as corners.

The set $\mathcal{X}^i$ are called *interface* vertices, and is made up of vertices on internal interfaces which are not classified as corners. An internal interface is any (one dimensional) path in the graph $\mathcal{T}$ made of a set of edges $\mathcal{C} \subset \mathcal{E}$, which we want to distinguish by making it an internal interface. Often such curves have significant physical meaning, locating a "front" or a discontinuity in a coefficient function, for example. We remark that a given domain may have no internal interfaces, in which case $\mathcal{X}^i = \emptyset$. A vertex $v \in \mathcal{X}^i$ implies that v is an endpoint of exactly two interior edges lying on the given interface, and v is not an endpoint of any edge lying on any other internal interface; vertices failing to meet this requirement must be classified as corners. Finally, all vertices which are not classified as corners, boundary vertices or interface vertices are called *interior* vertices and belong to the set $\mathcal{X}^0$. If $|\mathcal{X}| = N$, one would typically expect that $|\mathcal{X}^c| = O(1)$, $|\mathcal{X}^b| = O(\sqrt{N})$, $|\mathcal{X}^i| = O(\sqrt{N})$, and $|\mathcal{X}^0| = O(N)$, so that most vertices of the mesh will be classified as interior vertices.

We next consider the quality measures that we will use to control the shape regularity of the elements generated through the coarsening process.

We first consider a triangular element t shown in Figure 6, with vertices $v_i = (x_i, y_i)$, $1 \leq i \leq 3$, area a, and edge lengths ℓ_i, $1 \leq i \leq 3$.

The shape regularity quality of t, denoted by $G(t)$, is given by

$$G(t) = \frac{4\sqrt{3}a}{\ell_1^2 + \ell_2^2 + \ell_3^2}. \tag{4}$$

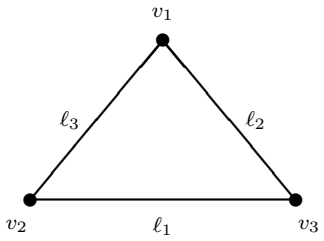

FIG. 6. *The edges and vertices of triangle t.*

The function $G(t)$ is normalized to equal one for an equilateral triangle and to approach zero for triangles with small angles. In particular, $G(t)$ is scaling invariant with respect to t. To understand the geometric meaning of $G(t)$ in somewhat more detail, without loss in generality, we assume for the moment that $v_1 = (0,0)$, $v_2 = (1,0)$, and $v_3 = (x,y)$ with $y \geq 0$, and consider the dependence of the $G(t)$ on the location of the vertex v_3. Noting that $0 \leq G(t) \leq 1$, we seek the set of points (x,y), for which $G(t) = \alpha$. From (4)

$$G(t) = \frac{2\sqrt{3}y}{1 + x^2 + (1-x)^2 + 2y^2} = \alpha.$$

This can be manipulated to the form

$$\left(x - \frac{1}{2}\right)^2 + \left(y - \frac{\sqrt{3}}{2\alpha}\right)^2 = \frac{3}{4}\left(\frac{1}{\alpha^2} - 1\right) = r^2$$

which is the equation of a circle.

For $\alpha = 1$, the center is $(1/2, \sqrt{3}/2)$ and the radius is $r = 0$; this means that the equilateral triangle is the only triangle for which $G(t) = 1$. For $\sqrt{3}/2 \leq \alpha \leq 1$, we have $0 \leq r \leq 1/2$; on this range, t cannot have any obtuse angles. As α is further decreased, the radius r becomes larger, and the triangle geometries with the quality α become more degenerate.

The question of assigning shape regularity to quadrilaterals is more complicated. Developing a measure in terms of standard finite element shape regularity requirements is clearly wrong, since we know that the edge based refinement process will necessarily generate some poorly shaped quadrilaterals when measured by such a criterion. On the other hand, quadrilaterals are only transitional elements which will ultimately become triangles. Thus we are lead to measure the quality of a quadrilateral in terms of the quality of (potential) triangles which can be generated from it. First, suppose that t is a nonconvex quadrilateral. In this case we decompose t uniquely into two triangles, denoted t_1 and t_2, and set

$$G(t) = \min\{G(t_1), G(t_2)\} \tag{5}$$

where $G(t_i)$ is defined by (4) for the triangles. If t is a convex quadrilateral, one can add either diagonal to form triangles; we denote the first pair as t_1 and t_2, and the second pair as t_1' and t_2'. For this case we set

$$G(t) = \max\left(\min\{G(t_1), G(t_2)\},\, \min\{G(t_1'), G(t_2')\}\right). \tag{6}$$

We next consider the parameters $\bar{\theta}$ and $\bar{\epsilon}$ of Section 2 used for controlling the refinement procedure. We now adopt here the notation used in (1). In our coarsening algorithm, we measure the movement of the vertex v relative to the midpoint v' of the straight line segment connecting its parents using the function $F(v)$ given by

$$F(v) = \frac{4\theta(1-\theta)}{1+100\epsilon^2}. \tag{7}$$

For the case $\theta = 1/2$, $\epsilon = 0$, $F(v) = 1$, while $F(v)$ becomes smaller as $|\theta - 1/2|$ or $|\epsilon|$ increases. In particular, within our coarsening algorithm, we impose the constraint

$$F(v) \geq F_{\min} \equiv 4\bar{\theta}(1-\bar{\theta}) > 0$$

which implies

$$\epsilon \leq \sqrt{\frac{(\bar{\theta}-1/2)^2 - (\theta-1/2)^2)}{\bar{\theta}(1-\bar{\theta})100}}.$$

This imposes a stronger constraint on ϵ as $|\theta - 1/2|$ increases. However, for a uniform bound, one has

$$\bar{\epsilon} = \sqrt{\frac{(\bar{\theta}-1/2)^2}{\bar{\theta}(1-\bar{\theta})100}}$$

in (3).

The next part of the discussion is concerned with the observation that quadrilateral elements are only transitional elements, and that part of our heuristic should directed towards keeping the number of such elements in the mesh at any give time as small as possible. To this end, we provide an attribute $m(v)$ for each vertex in $\mathcal{X}$. If $m(v) = 0$, then v is not the corner of any quadrilateral element. When a vertex is eliminated causing the creation of a new quadrilateral element (i.e., we move from a situation such as Figure 1 E to that of Figure 2 G),then $m(v)$ for the four vertices of the newly formed quadrilateral is updated as follows:

- For each of the two vertex parents, if currently marked with $m(v) = 0$, we reset $m(v) > 0$. This indicates that such vertices are of a *lower* level than the vertex which was eliminated.
- For the other two vertices in the quadrilateral, if currently marked with $m(v) = 0$, we reset $m(v) < 0$. This indicates that such vertices are likely to be of the *same* level as the vertex which was eliminated.

Our selection process will encourage the elimination of vertices with $m(v) < 0$, since we expect that elimination of such vertices will tend to reduce the number of quadrilateral elements. We note that the attribute $m(v)$ is time dependent; if at a later stage of the symbolic elimination $m(v)$ is no longer a corner of any quadrilateral elements, then we set $m(v) = 0$ again.

At any stage of the elimination process, all vertices in $\mathcal{X}$ either are assigned *tentative vertex parents* or they have no tentative parents and are called *orphans*. When a given vertex v, (which cannot be an orphan) is eliminated, $\mathcal{T}$ is updated to a new mesh $\hat{\mathcal{T}}$, with vertex set $\hat{\mathcal{X}}$ and edge set $\hat{\mathcal{E}}$ as follows:

Algorithm Delete (v)

1. The vertex v and all its incident edges are removed from $\mathcal{T}$. The updated vertex set $\hat{\mathcal{X}} = \mathcal{X} \setminus \{v\}$. The tentative parents of v become the *permanent* parents of v.
2. A new edge e (called a fillin edge) is added, connecting the two parents of vertex v. This completes the definition of the new mesh $\hat{\mathcal{T}}$. The updated edge set $\hat{\mathcal{E}}$ consists of the union of e and the edges remaining in $\mathcal{E}$ following Step 1.
3. For those vertices $w \in \mathrm{adj}_{\mathcal{T}}(v)$, we must update $\mathrm{adj}_{\hat{\mathcal{T}}}(w)$ and $m(w)$ as necessary.
4. For those vertices $w \in \cup_{z \in \mathrm{adj}_{\mathcal{T}}(v)} \mathrm{adj}_{\hat{\mathcal{T}}}(z)$, we must reevaluate their tentative parents or their status as orphans.

We now elaborate on the heuristic by which tentative vertex parents are chosen (or a vertex is determined to be an orphan). The following are *necessary* conditions to be satisfied in order for a vertex v to be assigned tentative parents, which we denote below as w_1 and w_2.

- v should not be a corner ($v \notin \mathcal{X}^c$), and $F(v) \geq F_{\min} > 0$.
- The tentative parents must satisfy $w_i \in \mathrm{adj}^e_{\mathcal{T}}(v)$, for $i = 1, 2$.
- Elimination of v as described above must create $\hat{\mathcal{T}}$ with only triangular and quadrilateral elements. This imposes the limits $3 \leq \deg_{\mathcal{T}}(v) \leq 6$ for $v \in \mathcal{X}^0 \cup \mathcal{X}^i$ and $3 \leq \deg_{\mathcal{T}}(v) \leq 4$ for $v \in \mathcal{X}^b$, and limits the number of possible pairs within the set $\mathrm{adj}^e_{\mathcal{T}}(v)$ which are eligible to be tentative parents.
- If $v \in \mathcal{X}^b$ then the tentative vertex parents must be vertices on the boundary, and connected to v by boundary edges. If $v \in \mathcal{X}^i$ then the tentative vertex parents must be vertices on the same internal interface as v, and connected to v by edges on the interface.
- Let $\mathcal{S}$ denote the set of new elements which would be generated in $\hat{\mathcal{T}}$ by the elimination of v, using w_1 and w_2 as tentative parents. Then we must have $\bar{G}(v) \equiv \min_{t \in \mathcal{S}} G(t) \geq G_{\min} > 0$, where (4)-(6) are used as appropriate to evaluate $G(t)$ for $t \in \mathcal{S}$.

We remark that the state of having tentative parents or of being an orphan is time dependent. As the elimination process proceeds, a given vertex might change states several times. If there are two or more pairs of vertices which satisfy all the necessary conditions to be tentative parents, then the function $Q(v)$ described below is evaluated for each pair, and a pair corresponding to the largest value of $Q(v)$ becomes the tentative parents.

For each vertex in the mesh we assign an overall quality $0 \leq Q(v) \leq 1$. If v is an orphan then $Q(v) = 0$. Otherwise, $Q(v) = .2\bar{G}(v) + .2F(v) + .6K(v)/18$, where $0 \leq K(v) \leq 18$ is an integer "point score" which reflects in a heuristic fashion factors which should favor the elimination of v.

- 4 points are assigned if $m(v) < 0$, and 2 points are assigned if $m(v) = 0$.
- If $v \in \mathcal{X}^i \cup \mathcal{X}^0$, 6, 4, 2, or 1 points are assigned if $\deg^e_{\mathcal{T}}(v)$ is 2, 3, 4, or 5, respectively. If $v \in \mathcal{X}^b$, 1 point is assigned if $\deg^e_{\mathcal{T}}(v)$ is 3.
- 2 points are assigned for each tentative parent w_i satisfying $m(w_i) > 0$, 1 point if $m(w_i) = 0$. 1 point is assigned for each $w_i \in \mathcal{X}^b$, and 2 points are assigned for each $w_i \in \mathcal{X}^c$.

One of our main objectives is to keep the number of quadrilaterals in the mesh at any given time as small as possible, so we award extra points in $K(v)$ on this basis. If $v \in \mathcal{X}^i \cup \mathcal{X}^0$, and $\deg^e_{\mathcal{T}}(v) = 2$, then typically eliminating v will reduce the number

of quadrilaterals in the mesh by two. If $\deg^e_{\mathcal{T}}(v) = 3$, then one quadrilateral will be eliminated, if $\deg^e_{\mathcal{T}}(v) = 4$, there will be no net change in the number of quadrilaterals, while if $\deg^e_{\mathcal{T}}(v) = 5$, there will be a net increase of one quadrilateral. (This remark is modified slightly if $\deg_{\mathcal{T}}(v) = 3$.) For $v \in \mathcal{X}^b$, and $\deg^e_{\mathcal{T}}(v) = 3$, we award fewer points for a similar situation, because it is easier to achieve.

Our overall symbolic elimination algorithm is now summarized below.

Algorithm Coarsen

1. All the vertices are placed in a heap according to the value of $Q(v)$, a vertex having the largest $Q(v)$ at the root. If the root vertex is not an orphan it is removed from the heap and eliminated using Algorithm Delete. As tentative vertex parents of surrounding vertices are updated, their positions in the heap generally change. We continue to eliminate the root vertex of the heap until we satisfy the termination criteria.
2. If the heap contains only orphans and the termination criteria is not satisfied, then we take the current mesh $\mathcal{T}$ and add diagonal edges to create two triangles from every quadrilateral in the mesh. For each vertex in the mesh we update adjacency lists, the function $m(*)$, evaluate tentative parents, and update its position in the heap. We then restart the elimination process in Step 1 above.
3. The elimination procedure terminates when one of the following is true.
   - all remaining vertices are corners.
   - the current mesh has only triangular elements and all the remaining vertices are orphans.
   - a prespecified target number of vertices for the coarse grid is achieved.

Typically, Algorithm Coarsen can eliminate $60 - 80\%$ of the vertices in a given mesh before a restart in Step 2 is required. Usually at such a time, the current mesh contains many quadrilaterals (despite our algorithmic attempts to keep the number small), and the extra constraints imposed by our algorithm on meshpoints which are vertices of quadrilaterals cause all meshpoints to become orphans. By eliminating the quadrilaterals, we eliminate the constraints, and also enlarge the sets $\text{adj}^e_{\mathcal{T}}(v)$, which usually allows many orphan vertices to then have tentative parents. However, adding diagonals to the quadrilaterals amounts to "dynamic edge swapping" when viewed from the perspective of Algorithm Refine, and can only be treated by our current theory if it is done a fixed number of times (independent of the number of levels).

We note that if $v \in \mathcal{X}^i \cup \mathcal{X}^0$ and $\deg_{\mathcal{T}}(v) = 3$ then v should be viewed as coming from an element refinement as in Figure 3. It would make some sense to assign such a vertex three parents instead of two. However, we expect such vertices to arise infrequently, and allowing the possibility of three parents adds what seems to us an unnecessary level of complication to an already complicated algorithm. On a positive note, such vertices are obviously easy to detect, create no quadrilaterals when eliminated, and their elimination improves the overall shape regularity of the new mesh. Thus we award some extra "points" in $K(v)$ for such vertices.

Although the basic structure controlling the order of elimination in Algorithm Coarsen is a heap, we think an effective algorithm could be designed based on a queue. Using a queue, one could potentially carry out the elimination process with less frequent updating of the data, and perhaps could eliminate groups of vertices simultaneously.

Finally, we remind the reader that our coarsening algorithm and the underlying

ideas are all heuristic. In the end, we can offer no formal justification for the algorithm other than it apparently works quite well, and that it is consistent with the refinement paradigm presented in Section 2. We expect that with more study and experimentation, these heuristics will be improved or better ones developed to replace them.

We now give a short numerical illustration of the use of Algorithm Coarsen. We take Ω to be a region in the shape of Lake Superior. The region has a fairly irregular outer boundary and six islands. The initial triangulation used 5927 vertices and is shown in Figure 7. We applied Algorithm Coarsen to this initial mesh, with a target coarse grid containing 300 vertices. Algorithm Coarsen was able to achieve this target value with two restarts, the first when the mesh had 1412 vertices, and the second when it had 398 vertices. All these meshes are shown in Figure 7. Note that the intermediate meshes with 1412 and 398 vertices do not correspond directly to the triangulations $\mathcal{T}_k$ of Section 2, but are simply the mesh as it existed prior to the restart, with many quadrilateral elements to be triangulated. Indeed, this calculation resulted in eight hierarchical levels. Also note that as the coarsening proceeds, the boundary $\partial\Omega$ is also approximated.

The vertex level $L(v)$ for vertices in the (original) fine mesh cannot be determined until Algorithm Coarsen is completed. Even then, $L(v)$ is not necessarily uniquely defined for all vertices in the mesh, since the coarsening process only yields a partial ordering of the vertices. Of course, it is known that $L(v) = 1$ for all vertices in the coarse grid. The levels of the remaining vertices must only satisfy

$$L(v) > \max\{L(w_1), L(w_2)\},$$

where w_1 and w_2 are the permanent parents of v. Our algorithm for assigning levels to vertices uses both a *lower level* $\underline{L}(v)$ and an *upper level* $\bar{L}(v)$. For coarse grid vertices, $\underline{L}(v) = \bar{L}(v) = L(v) = 1$. For other vertices,

$$\underline{L}(v) = \max\{\underline{L}(w_1), \underline{L}(w_2)\} + 1,$$

where w_1 and w_2 are the permanent parents of v. This is consistent with vertex levels assigned by Algorithm Refine, and can be easily computed by processing vertices in the *reverse* order in which they were eliminated. Let $\underline{L}_{\max} = \max_v \underline{L}(v)$, and let $S(v)$ denote the set of vertices which have v as one of their parents. Then if $S(v) = \emptyset$ $\bar{L}(v) = \underline{L}_{\max}$; otherwise,

$$\bar{L}(v) = \min_{w \in S(v)} \bar{L}(v) - 1,$$

which is easily computed by processing vertices in the *same* order in which they were eliminated. For some vertices (at least one at every level) $\underline{L}(v) = \bar{L}(v) \equiv L(v)$, while for many others, $\underline{L}(v) < \bar{L}(v)$, giving some flexibility in level assignment.

The assignment of levels has some practical implications, in that the nodal stiffness matrix (and implicitly, the hierarchical stiffness matrix) are blocked according to the partitioning of vertices into levels. Since the diagonal blocks of the stiffness matrix represented in terms of the hierarchical basis are actually computed as part of the preprocessing stages for hierarchical basis iterations, the block structure has some influence on the work and storage estimates. We have made some experiments using $L(v) \equiv \underline{L}(v)$ and $L(v) \equiv \bar{L}(v)$. Choosing $L(v) \equiv \bar{L}(v)$ results in the diagonal blocks having minimal order at every step, which should result in less work and storage. On typical problems, our empirically observed savings is about $5 - 10\%$ compared to

FIG. 7. *The initial triangulation with 5927 vertices, and intermediate grids with 1412, 398, and 300 vertices generated by Algorithm Coarsen.*

choosing $L(v) \equiv \underline{L}(v)$. Since the work and storage requirements are both linear in the number of vertices, this represents a modest, but still significant, reduction. We note that this choice need not be optimal in terms of work and storage, although at present, we do not see a way to improve upon it without investing significant additional computation. So far, we have noted little difference in observed rates of convergence for the two choices we have tried, and suspect all assignments consistent with the partial ordering will behave similarly.

In Table 1 we record the distribution of vertices among the eight levels for our numerical example of Lake Superior, using both the functions $\bar{L}(v)$ and $\underline{L}(v)$ to compute the levels. Here we notice a substantial difference between the two algorithms. However, this difference has a negligible impact on the observed rate of convergence.

We also note that in our example, the total number of nonzeroes in the upper triangular parts of all the matrices needed for the hierarchical basis multigrid method (see Section 6) is 33198 using levels determined by $\bar{L}(v)$ and 34971 using $\underline{L}(v)$. A problem with approximately 6000 vertices created using nested refinement would have

level	vertices computed using $\bar{L}(v)$	vertices computed using $\underline{L}(v)$
8	3732	21
7	1017	151
6	508	537
5	244	1140
4	89	1860
3	28	1118
2	9	500
1	300	300

TABLE 1
The distribution of vertices among the eight levels, computed using $\bar{L}(v)$ and $\underline{L}(v)$.

a total of approximately 30000 nonzeroes in the corresponding set of matrices. We attribute the increase in our example partly to the suboptimal choices of levels, and partly to edge swapping, which has the effect of increasing the density of nonzeroes in the hierarchical basis stiffness matrices.

4. Interpolation Operators for Nonnested Spaces. In this section, we define and analyze the interpolation operators used in constructing a hierarchical basis for a sequence of nonnested meshes. We will assume the notation developed in Section 2, and that the sequence of meshes $\{\mathcal{T}_k\}_{k=1}^{J}$ has been generated by Algorithm Refine. One messy detail of our analysis concerns the approximation of the boundary; all of the triangulations $\mathcal{T}_k$ are supposed to approximate the same underlying region Ω, but are able to do so only with different degrees of success. We denote by Ω_k the region corresponding to the triangulation $\mathcal{T}_k$, and without loss of generality, we assume that $\Omega_J \equiv \Omega$. For the remaining regions, we assume that all vertices which lie on $\partial\Omega_k$ also lie on $\partial\Omega$.

We define the bilinear form $a(u,w)_D$, $u, w \in \mathcal{H}^1(D)$, by

$$a(u,w)_D = \int_D \nabla u \cdot \nabla w \, dx$$

and the usual $\mathcal{L}^2$ inner product by

$$(u,w)_D = \int_D uw \, dx$$

for $u, w \in \mathcal{L}^2(D)$. For convenience we set $a(u,w) = a(u,w)_\Omega$ and $(u,w) = (u,w)_\Omega$. We define the (level dependent) energy seminorm by $|||u|||_D^2 = a(u,u)_D$, $|||u||| = |||u|||_\Omega$, and $\mathcal{L}^2$ norm by $\| u \|_D^2 = (u,u)_D$, $\| u \| = \| u \|_\Omega$. While it is convenient in this section to work with spaces without boundary conditions, we are careful to insure that the results also apply to the case where the spaces all satisfy homogeneous Dirichlet boundary conditions.

Associated with each triangulation $\mathcal{T}_k$, there is a space $\mathcal{M}_k \subset \mathcal{H}^1(\Omega_k)$ of continuous piecewise linear polynomials. We let N_k denote the dimension of $\mathcal{M}_k$. Since the triangulations $\mathcal{T}_k$ are not nested, the spaces $\mathcal{M}_k$ are not nested. However, by construction the vertex sets $\mathcal{X}_k$ *are* nested ($\mathcal{X}_k \subset \mathcal{X}_i$ for $i > k$).

We shall assume that all the elements in all the triangulations $\mathcal{T}_k$ are shape regular. In particular, we assume that there exists a positive constant $\bar{\delta}$, such that $G(t) \geq \bar{\delta}$ for

all $t \in \mathcal{T}_k$, $1 \le k \le J$, where the quality function $G(t)$ is defined in (4). We also need a measure of the relative sizes of elements in the various triangulations. Given an element t of size h_t, suppose that $\tilde{t}$ is the set of elements arising from the application of k steps of Algorithm Refine to t, with the additional assumption that at least one edge is refined at each step. Let h_{min} be the size of the smallest element in the patch $\tilde{t}$. Then there exists a constant $\bar{\gamma} = \bar{\gamma}(\bar{\theta}, \bar{\epsilon}, \bar{\delta})$, $\bar{\gamma} < 1$, such that $h_{min}/h_t \le \bar{\gamma}^k$.

We next define several primitive interpolation operators which play an important role in our hierarchical basis algorithms. First, let $\mathcal{I}_\infty^k : \mathcal{C}(\cup_j \Omega_j) \mapsto \mathcal{M}_k$ denote the usual nodal value interpolant. The interpolation operator $\mathcal{I}_k^{k+1} : \mathcal{M}_k \mapsto \mathcal{M}_{k+1}$ is a local interpolation operator based on the details of the refinement. Let $u \in \mathcal{M}_k$; then we can characterize $\mathcal{I}_k^{k+1} u \in \mathcal{M}_{k+1}$ in terms of its values at the set of vertices $\mathcal{X}_{k+1}$. First we set $\mathcal{I}_k^{k+1} u(v) = u(v)$ for all $v \in \mathcal{X}_k \subset \mathcal{X}_{k+1}$. For the remaining vertices, we refer to the notation of equation (1). We suppose $v_r \in \mathcal{X}_k$, $v_\ell \in \mathcal{X}_k$, and $v \in \mathcal{X}_{k+1} \setminus \mathcal{X}_k$. Then

$$\mathcal{I}_k^{k+1} u(v) = \theta u(v_r) + (1-\theta) u(v_\ell). \tag{8}$$

Notice that $\mathcal{I}_k^{k+1}$ is *not* pointwise interpolation except for the special case $\epsilon = 0$, that is, when all the meshes are physically as well as logically nested. In this case, $\mathcal{I}_k^{k+1}$ is just the restriction of $\mathcal{I}_\infty^{k+1}$ to $\mathcal{M}_k$. On the other hand, we note from (8) that the value of $\mathcal{I}_k^{k+1} u$ at a vertex $v \in \mathcal{X}_{k+1} \setminus \mathcal{X}_k$ is just a simple linear combination of the values of the function evaluated at v's vertex parents. This keeps the interpolation scheme very local, and extremely simple to implement. Finally, we note that (8) interpolates zero boundary conditions correctly.

We next define the composite interpolant $\mathcal{I}_k^\ell : \mathcal{M}_k \mapsto \mathcal{M}_\ell$ for $\ell > k$ by

$$\mathcal{I}_k^\ell = \mathcal{I}_{\ell-1}^\ell \mathcal{I}_{\ell-2}^{\ell-1} \cdots \mathcal{I}_k^{k+1} \tag{9}$$

with $\mathcal{I}_k^k = \mathcal{I}$, and the composite interpolant $\tilde{\mathcal{I}}_k^\ell : \mathcal{M}_\ell \mapsto \mathcal{M}_\ell$ by

$$\tilde{\mathcal{I}}_k^\ell = \mathcal{I}_k^\ell \mathcal{I}_\infty^k \tag{10}$$

for $k < \ell$, with $\tilde{\mathcal{I}}_\ell^\ell = \mathcal{I}$.

We now describe the hierarchical basis decomposition for $\mathcal{M}_J$. First note that by construction, each vertex $v \in \mathcal{X}_J$ has a *unique* level $1 \le L(v) \le J$. Using the notation of Algorithm Refine, and the identification $\tilde{\mathcal{X}}_1 \equiv \mathcal{X}_1$, we have the direct sum decomposition

$$\mathcal{X}_k = \tilde{\mathcal{X}}_1 \oplus \tilde{\mathcal{X}}_2 \oplus \cdots \oplus \tilde{\mathcal{X}}_k \tag{11}$$

for $1 \le k \le J$. For convenience, we will order the vertices of the mesh by level; v_i for $N_{k-1} + 1 \le i \le N_k$ will denote the set of vertices in $\tilde{\mathcal{X}}_k$.

For each space $\mathcal{M}_k$, let ϕ_i^k, $1 \le i \le N_k$, denote the usual nodal basis functions satisfying $\phi_i^k(v_j) = \delta_{ij}$ for $1 \le j \le N_k$. The hierarchical basis for $\mathcal{M}_k$ is defined inductively in the usual way. First, the hierarchical basis for $\mathcal{M}_1$ is just the nodal basis for $\mathcal{M}_1$. For $k > 1$, we construct a hierarchical basis by interpolating the hierarchical basis for $\mathcal{M}_{k-1}$ onto $\mathcal{M}_k$ using the operator $\mathcal{I}_{k-1}^k$, and taking the union of the resulting set of functions with the nodal basis functions for the level k nodes (ϕ_i^k, $N_{k-1} + 1 \le i \le N_k$). We let ψ_i^k denote the hierarchical basis functions for $\mathcal{M}_k$, ordered using the same scheme as the nodal basis functions. Then we have

$$\psi_i^k = \mathcal{I}_{L_i}^k \phi_i^{L_i} = \tilde{\mathcal{I}}_{L_i}^k \phi_i^k \tag{12}$$

where we have used the notation $L_i = L(v_i)$ to denote the level of vertex v_i.

We next consider a direct sum decomposition of the space $\mathcal{M}_k$. First, we define the spaces $\tilde{\mathcal{M}}_i^k = \mathcal{I}_i^k \mathcal{M}_i$ for $1 \le i \le k$. Clearly by (9) these spaces are nested $\tilde{\mathcal{M}}_i^k \subset \tilde{\mathcal{M}}_j^k \subset \tilde{\mathcal{M}}_k^k \equiv \mathcal{M}_k$, for $i \le j \le k$. By (10), we have $\tilde{\mathcal{M}}_i^k = \tilde{\mathcal{I}}_i^k \mathcal{M}_k$, for $1 \le i \le k$. Let $\mathcal{V}_i^k = (\tilde{\mathcal{I}}_i^k - \tilde{\mathcal{I}}_{i-1}^k)\mathcal{M}_k$ for $1 \le i \le k$, with the convention $\tilde{\mathcal{I}}_0^k = 0$. Then we have the direct sum decomposition

$$\tilde{\mathcal{M}}_i^k = \mathcal{V}_1^k \oplus \mathcal{V}_2^k \oplus \cdots \oplus \mathcal{V}_i^k \tag{13}$$

for $1 \le i \le k$. We are especially interested in (13) for the special case $i = k = J$,

$$\mathcal{M}_J \equiv \tilde{\mathcal{M}}_J^J = \mathcal{V}_1^J \oplus \mathcal{V}_2^J \oplus \cdots \oplus \mathcal{V}_J^J,$$

since it is the hierarchical decomposition of the space $\mathcal{M}_J$ which is of interest to us.

LEMMA 4.1. *Let $t \in \mathcal{T}_k$ be a triangle of size h_t, and let $\tilde{t}$ be the union of the triangles in $\mathcal{T}_\ell$, $\ell > k$, which arise from the refinement of t as in Algorithm Refine. Let $u \in \mathcal{M}_k$, and $\tilde{u}$ be the linear polynomial determined by the values of u at the vertices of t, and extended to $\tilde{t}$. Then*

$$(1 - c\bar{\epsilon})|||\tilde{u}|||_{\tilde{t}} \le |||\mathcal{I}_k^\ell u|||_{\tilde{t}} \le (1 + c\bar{\epsilon})|||\tilde{u}|||_{\tilde{t}} \tag{14}$$

where $c = c(\bar{\theta}, \bar{\epsilon}, \bar{\delta})$.

Proof. We note that (14) is a *local* result and does not require quasiuniformity of the mesh; however, this result does depend on the shape regularity of t and each triangle in $\tilde{t}$.

First, let us consider the linear function $\tilde{u}$, which we write as

$$\tilde{u} = \frac{\partial u}{\partial x} x + \frac{\partial u}{\partial y} y + u_0 = \nabla u \cdot \tilde{v} + u_0$$

where $\tilde{v} = (x, y)^t$, and u_0 is constant. Since the gradient of the constant function is zero, and $\mathcal{I}_k^\ell$ is exact for constants, without loss of generality, we can assume $u_0 = 0$.

Since $\mathcal{I}_k^\ell$ is a linear operator, and ∇u is constant, we have

$$\mathcal{I}_k^\ell u = \nabla u \cdot \mathcal{I}_k^\ell \tilde{v}.$$

Let us now examine $\mathcal{I}_k^\ell \tilde{v}$ using the decomposition (9). We consider first the simple case $\mathcal{I}_k^{k+1} v$, where v is a level $k+1$ vertex with vertex parents v_r and v_ℓ, in the notation of (1). By (8), we have

$$\mathcal{I}_k^{k+1} v = \theta v_r + (1 - \theta) v_\ell = v - \epsilon w$$

so at vertex v the interpolated function value of u is $\nabla u \cdot (v - \epsilon w)$. Note that the vector $-\epsilon w$ can be interpreted as moving v from its actual location in the nonnested mesh to the location it would have been had the refinement been nested ($\bar{\epsilon} = 0$). We also note that $\sqrt{w^t w} \le c h_t$. It is simple to see by induction that if v is a vertex of level j in $\tilde{t}$

$$\mathcal{I}_k^\ell u = \nabla u \cdot (v - e)$$

where $-e$ is a vector which moves v to its corresponding location in the case of nested refinement ($\bar{\epsilon} = 0$) of t, and

$$\begin{aligned} \sqrt{e^t e} &\le c\bar{\epsilon} h_t \{1 + \bar{\gamma} + \bar{\gamma}^2 + \cdots + \bar{\gamma}^{j-k-1}\} \\ &\le c\bar{\epsilon} h_t (1 - \bar{\gamma})^{-1}. \end{aligned}$$

Now let $\hat{t} \in \tilde{t}$ be a triangle in the level ℓ mesh with vertices v_i, nodal basis functions ϕ_i, and interpolated function values $\nabla u \cdot (v_i - e_i)$, $1 \le i \le 3$. Then on $\hat{t}$

$$\begin{aligned}\nabla \mathcal{I}_k^\ell u &= \nabla u - \sum_{i=1}^{3} \nabla u \cdot e_i \nabla \phi_i \\ &= \nabla u - \nabla u \cdot \{(e_2 - e_1)\nabla\phi_2 + (e_3 - e_1)\nabla\phi_3\}.\end{aligned}$$

Now we make a critical observation, that the vectors $e_2 - e_1$ and $e_3 - e_1$ denote the change in shape in element $\hat{t}$ relative to the case $\bar{\epsilon} = 0$, as opposed to a change in location; in effect, we have subtracted the translation common to all three vertices. Since $\hat{t}$ and the corresponding triangle in the case of nested refinement are both shape regular, we must have for $i = 2, 3$,

$$\sqrt{(e_i - e_1)^t (e_i - e_1)} \le c\bar{\epsilon} h_{\hat{t}}$$

Since $|\nabla \phi_i| \le c h_{\hat{t}}^{-1}$, we have immediately

$$(1 - c\bar{\epsilon})|||\tilde{u}|||_{\hat{t}} \le |||\mathcal{I}_k^\ell u|||_{\hat{t}} \le (1 + c\bar{\epsilon})|||\tilde{u}|||_{\hat{t}}.$$

The result now follows by summing over $\hat{t} \in \tilde{t}$. □

Since Lemma 4.1 is a critical result in our analysis, we pause here to give a numerical illustration. We consider a single element t, and refine 9 levels as in Figure 1 E, with $\theta = \bar{\theta} = 0.5$, and $\pm\epsilon = \bar{\epsilon} = 0.1$, $\pm\epsilon = \bar{\epsilon} = 0.05$, and $\pm\epsilon = \bar{\epsilon} = 0.01$. In Figure 8, we show the original element t and the regions $\tilde{t}$ for $k+1 \le \ell \le k+5$ and $\bar{\epsilon} = .1$. (the more refined regions have too many elements to be resolved in a small picture). In Figure 9, we show the corresponding contour maps for the function $u = \phi_i^k$ (one of the nodal basis functions for t) and the corresponding functions $\mathcal{I}_k^\ell u$. Finally, in Table 2, we show the function $|||\mathcal{I}_k^\ell u|||_{\tilde{t}}/|||\tilde{u}|||_{\tilde{t}} - 1$ for $k \le \ell \le k+9$, where one can see clearly the predicted behavior $c\bar{\epsilon}$.

ℓ	triangles	vertices	$\bar{\epsilon} = .1$	$\bar{\epsilon} = .05$	$\bar{\epsilon} = .01$
k	1	3	0.0	0.0	0.0
k+1	4	6	-2.77(-2)	-1.41(-2)	-2.87(-3)
k+2	16	15	-2.00(-2)	-1.21(-2)	-2.79(-3)
k+3	64	45	-1.54(-2)	-1.10(-2)	-2.75(-3)
k+4	256	153	-0.96(-2)	-9.64(-3)	-2.69(-3)
k+5	1024	561	-0.37(-2)	-8.26(-3)	-2.63(-3)
k+6	4096	2145	0.25(-2)	-6.89(-3)	-2.59(-3)
k+7	16384	8385	0.89(-2)	-5.52(-3)	-2.53(-3)
k+8	65536	33153	1.59(-2)	-4.01(-3)	-2.48(-3)
k+9	262144	131841	2.23(-2)	-2.71(-3)	-2.32(-3)

TABLE 2
The function $|||\mathcal{I}_k^\ell u|||_{\tilde{t}}/|||\tilde{u}|||_{\tilde{t}} - 1$ *for* $k \le \ell \le k+9$ *and* $\bar{\epsilon} = .1, .05, .01$.

LEMMA 4.2. *Let $t \in \mathcal{T}_k$ be a triangle of size h_t, and let $\tilde{t}$ be the union of the triangles in $\mathcal{T}_\ell$, $\ell > k$, which arise from the refinement of t as in Algorithm Refine. Let $u \in \mathcal{M}_\ell$ and $w = \mathcal{I}_\infty^k u$, and let $\tilde{w}$ be the linear polynomial determined by the values of w at the vertices of t, and extended to $\tilde{t}$. Then*

$$|||\tilde{w}|||_{\tilde{t}} \le C\sqrt{\ell - k}|||u|||_{\tilde{t}} \tag{15}$$

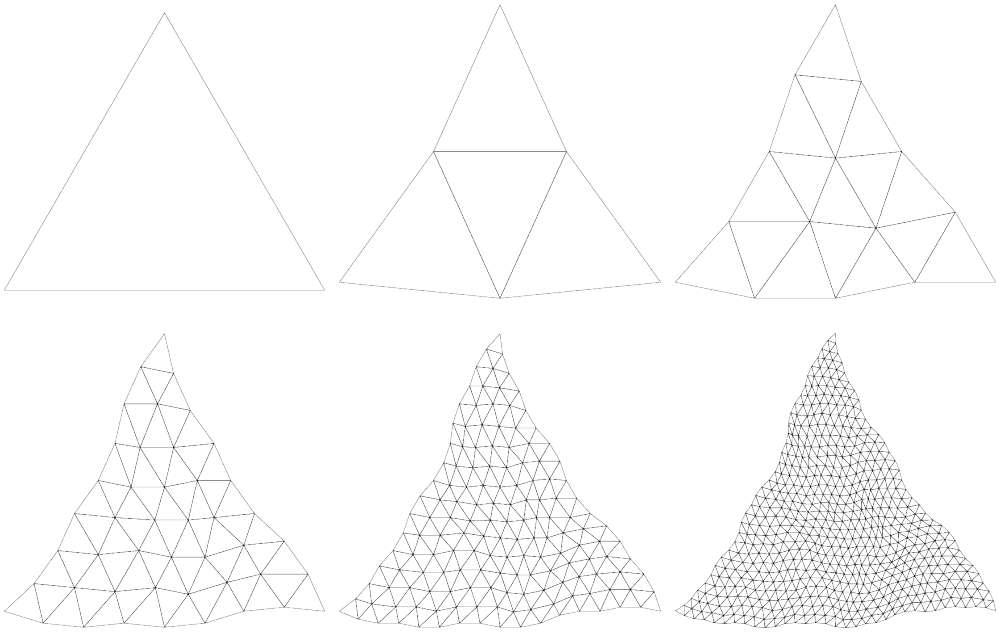

FIG. 8. *The element t and $\tilde{t}$ for $\ell = k+1, k+2, \cdots, k+5$, and $\bar{\epsilon} = .1$*

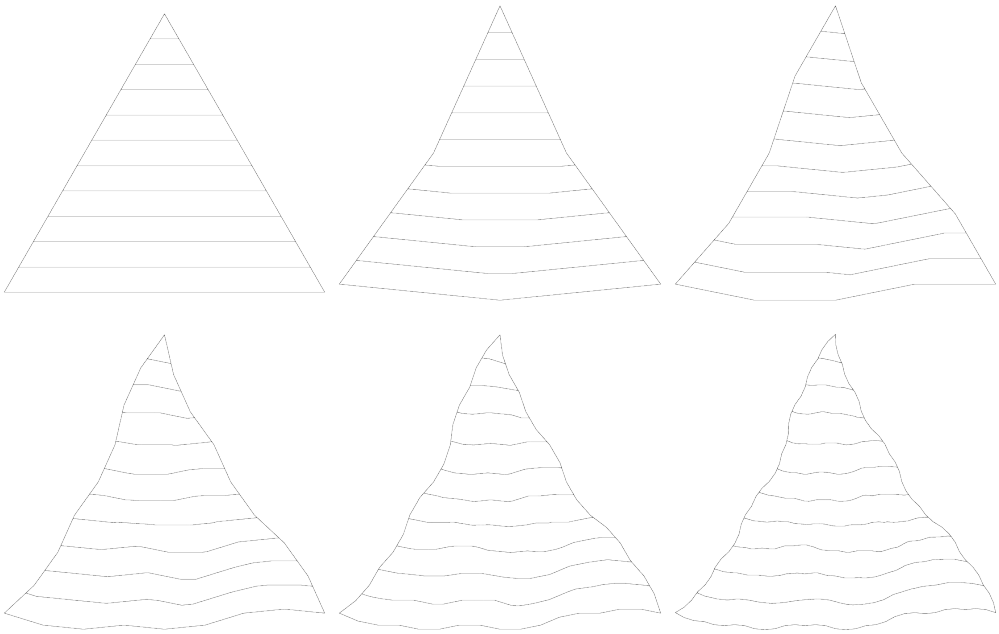

FIG. 9. *Contour maps for the function u and $\mathcal{I}_k^\ell u$ for $\ell = k+1, k+2, \cdots, k+5$, and $\bar{\epsilon} = .1$*

for $C = C(\bar{\theta}, \bar{\epsilon}, \bar{\delta})$.

Proof. Since $w \in \mathcal{M}_k$, w is just a linear polynomial on t. As in the proof of Lemma 4.1, let v_i, $1 \le i \le 3$, denote the vertices of t. Then, noting $w(v_i) = u(v_i)$, we have,

$$|||\tilde{w}|||_{\tilde{t}} \le C \max_i |w(v_i)| = C \max_i |u(v_i)| \le C \parallel u \parallel_{\infty,\tilde{t}} .$$

Let h_{min} denote the size of the smallest triangle in $\tilde{t}$. Then

$$\parallel u \parallel^2_{\infty,\tilde{t}} \le C \log\left(\frac{h_t}{h_{min}}\right) |||u|||^2_{\tilde{t}} \le C \log\left(\bar{\gamma}^{k-\ell}\right) |||u|||^2_{\tilde{t}} \le C(\ell - k)|||u|||^2_{\tilde{t}}.$$

This completes the proof. □

THEOREM 4.3. *Let $u \in \mathcal{M}_\ell$ and let $\ell > k$. Then*

$$|||\tilde{\mathcal{I}}^\ell_k u|||_{\Omega_\ell} \le C\sqrt{\ell - k}|||u|||_{\Omega_\ell} \tag{16}$$

for $C = C(\bar{\theta}, \bar{\epsilon}, \bar{\delta})$.

Proof. Let $t \in \mathcal{T}_k$, and let $\tilde{t}$ be the union of triangles in $\mathcal{T}_\ell$ which arise from the refinement of t as in Algorithm Refine. Let $w = \mathcal{I}^k_\infty u$ and let $\tilde{w}$ be the linear polynomial determined by the values of w at the vertices of t and extended to $\tilde{t}$. Then by Lemmas 4.1 and 4.2,

$$\begin{aligned} |||\tilde{\mathcal{I}}^\ell_k u|||_{\tilde{t}} &= |||\mathcal{I}^\ell_k w|||_{\tilde{t}} \\ &\le (1 + c\bar{\epsilon})|||\tilde{w}|||_{\tilde{t}} \\ &\le C(1 + c\bar{\epsilon})\sqrt{\ell - k}|||u|||_{\tilde{t}}. \end{aligned}$$

The result (16) follows by summing over $t \in \mathcal{T}_k$. □

The next lemma shows that a bound on the interpolant as in (16) in essentially equivalent to a strengthened Cauchy inequality [3] [12].

LEMMA 4.4. *Suppose $\mathcal{M} = \mathcal{V} \oplus \mathcal{W}$, and let $\mathcal{I}$ denote the interpolation operator defined as follows: if $u = v + w \in \mathcal{M}$, $v \in \mathcal{V}$, and $w \in \mathcal{W}$, then $\mathcal{I}(u) = v$. Then*

$$|||\mathcal{I}(u)|||_D \le C|||u|||_D \tag{17}$$

if and only if

$$|a(v, w)_D| \le \gamma|||v|||_D|||w|||_D \tag{18}$$

for $\gamma < 1$ and for all $v \in \mathcal{V}$ and $w \in \mathcal{W}$.

Proof. First, we assume (18) in order to prove (17). Let $u = v + w$, $v \in \mathcal{V}$, $w \in \mathcal{W}$. Then

$$\begin{aligned} |||u|||^2_D &= a(v + w, v + w)_D \\ &= |||v|||^2_D + |||w|||^2_D + 2a(v, w)_D \\ &\ge |||v|||^2_D + |||w|||^2_D - 2\gamma|||v|||_D|||w|||_D \\ &\ge (1 - \gamma^2)|||v|||^2_D. \end{aligned}$$

Therefore

$$|||\mathcal{I}(u)|||_D \le \frac{1}{\sqrt{1 - \gamma^2}}|||u|||_D.$$

Now we assume (17) to show (18). It suffices to take $|||v|||_D = |||w|||_D = 1$. Then, from (17)

$$|||v - w|||_D \geq \frac{1}{C}|||v|||_D = \frac{1}{C}.$$

Thus,

$$\begin{aligned} a(v,w)_D &= \frac{1}{2}|||v|||_D^2 + \frac{1}{2}|||w|||_D^2 - \frac{1}{2}|||v-w|||_D^2 \\ &\leq 1 - \frac{1}{2C^2}. \end{aligned}$$

□

THEOREM 4.5. *Let $u = v + w \in \mathcal{M}_k$, with*

$$v \in \mathcal{V}_1^k \oplus \mathcal{V}_2^k \oplus \cdots \oplus \mathcal{V}_i^k$$

and

$$w \in \mathcal{V}_{i+1}^k \oplus \mathcal{V}_{i+2}^k \oplus \cdots \oplus \mathcal{V}_k^k$$

for $1 \leq i < k$. Then

$$|a(v,w)_{\Omega_k}| \leq \left(1 - \frac{C}{k-i}\right)|||v|||_{\Omega_k}|||w|||_{\Omega_k}. \tag{19}$$

Proof. Apply Lemma 4.4 and Theorem 4.3. □

We next prove another strengthened Cauchy inequality.

THEOREM 4.6. *Let $u \in \mathcal{V}_i^\ell$ and $w \in \mathcal{V}_k^\ell$. Then there exists a constant $C = C(\bar{\theta}, \bar{\epsilon}, \bar{\delta})$, such that, for $\bar{\epsilon}$ sufficiently small,*

$$|a(u,w)_{\Omega_\ell}| \leq C\left(\bar{\gamma}^{|i-k|/2} + \bar{\epsilon}\right)|||u|||_{\Omega_\ell}|||w|||_{\Omega_\ell}. \tag{20}$$

Proof. Our proof follows closely the proof of Lemma 4.1. For convenience, suppose $i < k$, and let $t \in \mathcal{T}_i$ and $\tilde{t}$ denote the set of elements in $\mathcal{T}_\ell$ resulting from the refinement of t. Let $\tilde{u}$ be the linear polynomial characterized by the values of u at the vertices of t, and set $u = \nabla\tilde{u} \cdot (v - e) = \tilde{u} + z$, as in the proof of Lemma 4.1. As in the proof of Lemma 4.1, we may set the constant $\tilde{u}_0 = 0$.

We first consider the term $a(\tilde{u}, w)_t$ and note

$$a(\tilde{u}, w)_{\tilde{t}} = \int_{\partial\tilde{t}} \frac{\partial\tilde{u}}{\partial n} w \, dx$$

where n is the unit normal. Note that $w \equiv 0$ on edges of $\partial\tilde{t}$ which are level $k-1$ or less, and $w = 0$ at all vertices in $\tilde{t}$ of level $k-1$ or less. Thus $w \equiv 0$ on any element in $\tilde{t}$ not containing at least one vertex of level k or larger, and where $w \neq 0$, it must be oscillatory. Let h_i denote the size of $\tilde{t}$, and h_k denote size of elements of level k which came from the refinement of t. Then $h_k/h_i \leq C\bar{\gamma}^{k-i}$, and

$$
\begin{aligned}
|a(\tilde{u},w)_{\tilde{t}}| &\le \left(\int_{\partial\tilde{t}}\left\{\frac{\partial\tilde{u}}{\partial n}\right\}^2 dx\right)^{1/2}\left(\int_{\partial\tilde{t}} w^2 dx\right)^{1/2}\\
&\le \frac{c}{\sqrt{h_i}}|||\tilde{u}|||_{\tilde{t}}\frac{h_k}{\sqrt{h_i}}|||w|||_{\tilde{t}}\\
&\le c\bar{\gamma}^{|i-k|/2}|||u|||_{\tilde{t}}\,|||w|||_{\tilde{t}}.
\end{aligned}
$$

Here we have estimated $|||\tilde{u}|||_{\tilde{t}}$ using Lemma 4.1 assuming that $1-c\bar{\epsilon}>0$ in (14), and used standard trace and inverse estimates for the terms involving w. Using the arguments of Lemma 4.1 once more, we have

$$
|a(z,w)_{\tilde{t}}| \le C\bar{\epsilon}|||u|||_{\tilde{t}}\,|||w|||_{\tilde{t}}.
$$

Combining these two estimates, and summing over $t\in\mathcal{T}_i$ (in effect, summing over the refined patches $\tilde{t}$) leads to the estimate (20). □

Our final theorem is

THEOREM 4.7. *Let $u\in\mathcal{V}_k^\ell$, $u=\sum_i U_i\psi_i^\ell$, where the ψ_i^ℓ are the hierarchical basis functions described above, and $k>1$. Then there exist constants $C_i=C_i(\bar{\theta},\bar{\epsilon},\bar{\delta})$, $1\le i\le 2$, such that*

$$
C_1|||u|||_{\Omega_\ell}^2 \le \sum_i U_i^2|||\psi_i^\ell|||_{\Omega_\ell}^2 \le C_2|||u|||_{\Omega_\ell}^2. \tag{21}
$$

Proof. The left inequality in (21) is a simple exercise using the triangle inequality, and the fact the support of ψ_i^ℓ can intersect the support of only a fixed number of hierarchical basis functions of the same level, and is true even in the case $k=1$. The right inequality is more difficult, and will be proved elementwise for $t\in\mathcal{T}_k$. Since $\mathcal{T}_k$ came from the refinement of $\mathcal{T}_{k-1}$, all of the elements must be of the patterns shown in Figure 1. As usual, we will let $\tilde{t}$ denote the patch of elements in $\mathcal{T}_\ell$ which arise from the subsequent refinement of t.

In Figure 1 A, no edges were refined and the single element has all vertices with level less than k; thus $u\equiv 0$ on $\tilde{t}$, and there is nothing to prove. In Figure 1 B, there is one level k vertex, and hence one nonzero basis function, say $\psi_{i_1}^\ell$. If $\tilde{t}$ corresponds to one of the two elements $t\in\mathcal{T}_k$ arising from such a refinement then $|||u|||_{\tilde{t}}^2 = U_{i_1}^2|||\psi_{i_1}^\ell|||_{\tilde{t}}^2$, and again the proof is trivial.

In Figure 1 C, one of the three elements has only one nonzero basis function, and is similar to the case of Figure 1 B. The remaining elements have two nonzero basis functions, say $\psi_{i_1}^\ell$ and $\psi_{i_2}^\ell$. To obtain a local bound, we are led to study the 2×2 eigenvalue problem

$$
\begin{bmatrix} |||\psi_{i_1}^\ell|||_{\tilde{t}}^2 & a(\psi_{i_1}^\ell,\psi_{i_2}^\ell)_{\tilde{t}} \\ a(\psi_{i_1}^\ell,\psi_{i_2}^\ell)_{\tilde{t}} & |||\psi_{i_2}^\ell|||_{\tilde{t}}^2 \end{bmatrix}\begin{bmatrix} U_{i_1}\\ U_{i_2}\end{bmatrix} = \lambda\begin{bmatrix} |||\psi_{i_1}^\ell|||_{\tilde{t}}^2 & 0 \\ 0 & |||\psi_{i_2}^\ell|||_{\tilde{t}}^2. \end{bmatrix}\begin{bmatrix} U_{i_1}\\ U_{i_2}\end{bmatrix}
$$

The eigenvalues are $\lambda=1\pm\alpha$, where

$$
\alpha = \frac{|a(\psi_{i_1}^\ell,\psi_{i_2}^\ell)_{\tilde{t}}|}{|||\psi_{i_1}^\ell|||_{\tilde{t}}|||\psi_{i_2}^\ell|||_{\tilde{t}}}.
$$

We will show $\alpha<1$, depending only on $\bar{\theta}$, $\bar{\epsilon}$ and $\bar{\delta}$. Following the pattern of Lemma 4.1, we let $\psi_{i_1}^\ell=\tilde{\psi}_{i_1}^\ell+z_{i_1}$, where $\tilde{\psi}_{i_1}^\ell$ is the linear polynomial based on the

values at the vertices of t (in this case, $\tilde{\psi}^\ell_{i_1} = 0$ at two nodes and $\tilde{\psi}^\ell_{i_1} = 1$ at the third). We make an analogous decomposition for $\psi^\ell_{i_2}$. Note that $\tilde{\psi}^\ell_{i_2} = 1$ at one of the two vertices where $\tilde{\psi}^\ell_{i_1} = 0$. Now let $\hat{t} \in \mathcal{T}_\ell$ be one of the triangles in the set of elements making up $\tilde{t}$. Then $\nabla\psi^\ell_{i_1}$ and $\nabla\psi^\ell_{i_2}$ are constant on $\hat{t}$ and essentially small perturbations of $\nabla\tilde{\psi}^\ell_{i_1}$ and $\nabla\tilde{\psi}^\ell_{i_2}$, respectively. By shape regularity combined with their definitions we must have the strengthened Cauchy inequality

$$|\nabla\psi^\ell_{i_1} \cdot \nabla\psi^\ell_{i_2}| \leq \alpha_{\hat{t}} \sqrt{\nabla\psi^\ell_{i_1} \cdot \nabla\psi^\ell_{i_1}} \sqrt{\nabla\psi^\ell_{i_2} \cdot \nabla\psi^\ell_{i_2}}$$

leading directly to the estimate

$$|a(\psi^\ell_{i_1}, \psi^\ell_{i_2})_{\hat{t}}| \leq \alpha_{\hat{t}} |||\psi^\ell_{i_1}|||_{\hat{t}} |||\psi^\ell_{i_2}|||_{\hat{t}}.$$

Taking $\alpha = \max_{\hat{t}} \alpha_{\hat{t}}$ and summing over $\hat{t}$ complete the estimate in this case.

For refinement as in Figure 1 D, all four elements have two nonzero basis functions and are covered by the preceding argument. For refinement as in Figure 1 E, three of the four elements in $\mathcal{T}_k$ have two nonzero basis functions, while the middle element has three nonzero basis functions. The argument above will fail for such elements, as the analogous 3×3 eigenvalue problem will have one zero eigenvalue and two positive eigenvalues. On the other hand, such elements only occur in the situation shown in Figure 1 E, where they are surrounded by three elements, each having only two nonzero basis functions. Let t_0 denote the center element with three nonzero basis functions, and let t_1, t_2 and t_3 denote the other three triangles, each with two nonzero basis functions. Let $\psi^\ell_{i_1}$, $\psi^\ell_{i_2}$, and $\psi^\ell_{i_3}$ denote the three basis functions. Suppose $\psi^\ell_{i_1}$ has support in t_2 and t_3 as well as t_0. Then using arguments similar to those in Lemma 4.1, it is straightforward to show

$$c_1 |||\psi^\ell_{i_1}|||^2_{\tilde{t}_0} \leq |||\psi^\ell_{i_1}|||^2_{\tilde{t}_2} + |||\psi^\ell_{i_1}|||^2_{\tilde{t}_3} \leq c_2 |||\psi^\ell_{i_1}|||^2_{\tilde{t}_0}$$

so that we can control the behavior of the center element though the surrounding elements which have only two nonzero basis functions. Similar estimates apply to the other two basis functions. These can be combined to show

$$U^2_{i_1} |||\psi^\ell_{i_1}|||^2_{\tilde{t}_0} + U^2_{i_2} |||\psi^\ell_{i_2}|||^2_{\tilde{t}_0} + U^2_{i_3} |||\psi^\ell_{i_3}|||^2_{\tilde{t}_0} \leq C |||u|||^2_{\tilde{t}_0 \cup \tilde{t}_1 \cup \tilde{t}_2 \cup \tilde{t}_3}.$$

Now, by combining all cases, and summing over $t \in \mathcal{T}_k$, we obtain the right hand inequality in (21). □

We close this section with a few remarks. The first concerns the issue of dynamic edge swapping. We note that under appropriate assumptions, one can show $|||\mathcal{S}u|||_D \approx |||u|||_D$ where $\mathcal{S}$ is an "edge swapping" operator. For example, if we were to allow dynamic edge swapping once, say only on the finest triangulation $\mathcal{T}_J$, we would have growth like $C(1 + c\bar{\epsilon})$, $C > 1$ in Lemma 4.1. This could be generalized to any fixed number p of levels on which edge swapping is allowed with growth like $C^p(1 + c\bar{\epsilon})^p$. However, if we allowed dynamic edge swapping on *all* levels ($p = J$), clearly the growth would not be acceptable. We emphasize that we do not believe that dynamic edge swapping is harmful to the coarsening (or refinement) process; indeed, it is a critical part of the restart step of Algorithm Coarsen, and from our practical experience in this context, has little if any observed effect on the convergence of iterations using hierarchical basis preconditioners. It just seems that the theory developed in this section is not yet mature enough to adequately explain this observed behavior.

Our second remark concerns the effect of a variable coefficient in the enery inner product. Suppose that

$$\tilde{a}(u,w)_D = \int_D a\nabla u \cdot \nabla w \, dx$$

where $a = a(x,y)$ is a piecewise smooth, positive, bounded function on D. Let $t \in \mathcal{T}_1$ and $\tilde{t}$ be the set of elements in $\mathcal{T}_J$ resulting from the refinement of t. Let $0 < \underline{a}_{\tilde{t}} \leq a(x,y) \leq \bar{a}_{\tilde{t}}$ on $\tilde{t}$ and set $\alpha = \max_{\tilde{t}} \bar{a}_{\tilde{t}}/\underline{a}_{\tilde{t}}$. Then most constants in our analysis would depend on α as well as $\bar{\theta}$, $\bar{\epsilon}$ and $\bar{\bar{\delta}}$. We note that α is a locally defined quantity, and depends only on the variation of $a(x,y)$ on fine grid patches of elements corresponding to coarse grid triangles. For example, if a is a piecewise constant, it will have *no* effect on our estimates as long as the interfaces where a has jumps are modeled in the coarse triangulation. This is independent of the sizes of the jumps, and is of course exactly similar to the behavior of hierarchical basis decompositions on sequences of nested meshes. However, it is interesting to note that we do not have to approximate these interfaces exactly on the coarse grids, just as we do not have to approximate the boundary $\partial\Omega$ exactly on the coarse grids. Indeed the same basic refinement rule applies to both interfaces and boundaries: when an interface (boundary) edge is refined, the new vertex should be placed on the interface (boundary). Neither the original interface (boundary) edge or its two children edges are required to coincide exactly with the interface (boundary). This explains our motivation for giving boundary and interface vertices similar special treatment in assigning tentative vertex parents in Algorithm Coarsen.

5. Some Hierarchical Basis Preconditioners. In this section, we will analyze block Jacobi and block symmetric Gauss-Seidel iterations using the hierarchical decomposition

$$\mathcal{M}_J = \mathcal{V}_1^J \oplus \mathcal{V}_2^J \oplus \cdots \oplus \mathcal{V}_J^J$$

defined in Section 4. However, for convenience in this section, we assume that the space $\mathcal{M}_J \subset \mathcal{H}_0^1(\Omega)$. Satisfying the Dirichlet boundary conditions amounts to excluding basis functions associated with boundary vertices from the subspace. In order to avoid the notational explosion involved with this change, we adopt the same notation as in Section 4, and simply remark that many symbols have slightly altered meanings to accommodate the boundary conditions. One important change is that $|||\cdot|||$ becomes a strong norm instead of a seminorm.

As before, we let $\{\psi_i^k\}_{i=N_{k-1}+1}^{N_k}$ denote the hierarchical basis functions for the level-k vertices in $\mathcal{T}_J$. Then the stiffness matrix A, represented in the hierarchical basis, is expressed as the symmetric, positive definite block $J \times J$ matrix

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1J} \\ A_{21} & A_{22} & & A_{2J} \\ \vdots & & \ddots & \vdots \\ A_{J1} & A_{J2} & \cdots & A_{JJ} \end{bmatrix} \tag{22}$$

where A_{kk} is the $(N_k - N_{k-1}) \times (N_k - N_{k-1})$ matrix of energy inner products involving just the level-k basis functions. We set

$$A = L + D + L^t \tag{23}$$

where

$$D = \begin{bmatrix} A_{11} & & & \\ & A_{22} & & \\ & & \ddots & \\ & & & A_{JJ} \end{bmatrix}$$

and

$$L = \begin{bmatrix} 0 & & & \\ A_{21} & 0 & & \\ \vdots & & \ddots & \\ A_{J1} & A_{J2} & \cdots & 0 \end{bmatrix}.$$

The block diagonal matrix D is further decomposed as $D = d + \ell + \ell^t$, where

$$d = \begin{bmatrix} A_{11} & & & \\ & d_{22} & & \\ & & \ddots & \\ & & & d_{JJ} \end{bmatrix}$$

$d_{ii} = \mathrm{Diag} A_{ii}$, and

$$\ell = \begin{bmatrix} 0 & & & \\ & \ell_{22} & & \\ & & \ddots & \\ & & & \ell_{JJ} \end{bmatrix}$$

is lower triangular.

We first analyze the block Jacobi preconditioner d. This is similar to the hierarchical basis method of Yserentant [29].

THEOREM 5.1. *Let $A = L + D + L^t$ and $D = \ell + d + \ell^t$ as defined above. Then there exist positive constants $C_i = C_i(\bar{\theta}, \bar{\epsilon}, \bar{\delta})$, $i = 1, 2$, such that*

$$C_1(1 + \bar{\epsilon} J)^{-1} \leq \frac{U^t d U}{U^t A U} \leq C_2 J^2 \tag{24}$$

for $U \neq 0$.

Proof. The upper bound in (24) is similar to the hierarchical basis preconditioner for the case of nested meshes. The lower bound in the nested case does not have the term $(1 + \bar{\epsilon} J)^{-1}$. However, for typical choices of $\bar{\epsilon}$ and J, this term is effectively bounded by a constant, so as a practical matter, the bound (22) does not differ significantly from the nested mesh case. We begin our proof by writing the Rayleigh quotient in (24) as

$$\frac{U^t d U}{U^t A U} = \frac{U^t D U}{U^t A U} \frac{U^t d U}{U^t D U}$$

and will estimate each of the terms separately. Let $u \in \mathcal{M}_J$ correspond to U in (24). Then the Rayleigh quotient $U^t D U / U^t A U$ can be written as

$$\frac{U^t D U}{U^t A U} = \frac{\sum_{k=1}^{J} |||z_k|||^2}{|||u|||^2}$$

where $z_k = (\tilde{\mathcal{I}}_k^J - \tilde{\mathcal{I}}_{k-1}^J)u$.

Applying Theorem 4.3 for the upper bound, and Theorem 4.6 for the lower bound, we have

$$c_1(1+\bar{\epsilon}J)^{-1} \leq \frac{U^t DU}{U^t AU} \leq c_2 J^2.$$

For the Rayleigh quotient $U^t dU / U^t DU$, we can apply Theorem 4.7 to see

$$c_1 \leq \frac{U^t dU}{U^t DU} \leq c_2.$$

Combining these estimates proves (24). □

We next consider the symmetric block Gauss-Seidel preconditioner using symmetric Gauss-Seidel "inner iterations", an algorithm similar to the Hierarchical Basis Multigrid Method [4] [1]. We begin with a few preliminary lemmas.

LEMMA 5.2. *Let $A = L+D+L^t$ and let $\tilde{B} = (D+L^t)D^{-1}(D+L) = A+L^tD^{-1}L$. Then there exists a positive constant $C = C(\bar{\theta}, \bar{\epsilon}, \bar{\delta})$, such that*

$$1 \leq \frac{U^t \tilde{B} U}{U^t AU} \leq 1+\mu_0 \leq CJ^2 \tag{25}$$

for $U \neq 0$, and

$$\mu_0 = \max_{U \neq 0} \frac{W^t DW}{U^t AU}.$$

$$DW = LU.$$

Proof. The preconditioner $\tilde{B}$ is essentially that for block symmetric Gauss-Seidel, assuming that all linear systems involving the diagonal blocks A_{ii} are solved exactly. The lower bound and upper bound $1+\mu_0$ follow immediately from the definition of $\tilde{B}$ and the fact that $L^tD^{-1}L$ is positive semidefinite. To estimate μ_0, let $u \in \mathcal{M}_J$ correspond to U in (25). Then, in finite element notation,

$$\mu_0 = \max_{u \neq 0} \frac{\sum_{k=2}^J |||w_k|||^2}{|||u|||^2}$$

where

$$a(w_k, \chi) = a(\tilde{\mathcal{I}}_{k-1}^J u, \chi)$$

for all $\chi \in \mathcal{V}_k^J$.

Taking $\chi = w_k$ in this relation and applying Theorem 4.3 leads to

$$|||w_k||| \leq |||\tilde{\mathcal{I}}_{k-1}^J u||| \leq C\sqrt{J-k+1}|||u|||$$

which immediately implies $\mu_0 \leq CJ^2$. □

LEMMA 5.3. *Let $D = \ell + d + \ell^t$ and let $\tilde{D} = (d+\ell^t)d^{-1}(d+\ell) = D + \ell^t d^{-1}\ell$. Then there exists a positive constant $C = C(\bar{\theta}, \bar{\epsilon}, \bar{\delta})$, such that*

$$1 \leq \frac{U^t \tilde{D} U}{U^t DU} \leq 1+\mu_1 \leq C \tag{26}$$

for $U \neq 0$, and

$$\mu_1 = \max_{U \neq 0} \frac{W^t dW}{U^t D U}$$

$$dW = \ell U.$$

Proof. The preconditioner $\tilde{D}$ summarizes the symmetric Gauss-Seidel inner iterations. The proof follows the pattern of that for Lemma 5.2, using Theorem 4.7 in place of Theorem 4.3. □

The preconditioner for the Hierarchical Basis Multigrid Method using symmetric Gauss-Seidel inner iterations is given by

$$\begin{aligned} B &= (\tilde{D} + L)^t (2\tilde{D} - D)^{-1} (\tilde{D} + L) \\ &= A + (D - \tilde{D} + L)^t (2\tilde{D} - D)^{-1} (D - \tilde{D} + L) \\ &= A + (L - Z)^t (D + 2Z)^{-1} (L - Z) \end{aligned} \tag{27}$$

where $Z = \ell^t d^{-1} \ell$ [4] [1].

THEOREM 5.4. *Let $A = L + D + L^t$ and let B be given by (27). Then there exists a positive constant $C = C(\bar{\theta}, \bar{\epsilon}, \bar{\delta})$, such that*

$$1 \leq \frac{U^t B U}{U^t A U} \leq C J^2 \tag{28}$$

for $U \neq 0$.

Proof. We begin by noting that the matrix $(L - Z)^t (D + 2Z)^{-1} (L - Z)$ is symmetric, positive semidefinite. This gives a lower bound of of $1 \leq U^t B U / U^t A U$ and an upper bound of the form $U^t B U / U^t A U \leq 1 + \mu$, where

$$\begin{aligned} \mu &= \max_{U \neq 0} \frac{U^t (L - Z)^t (D + 2Z)^{-1} (L - Z) U}{U^t A U} \\ &= \| \, (D + 2Z)^{-1/2} (L - Z) A^{-1/2} \, \|_{\ell^2}^2 \\ &\leq \Big(\| \, (D + 2Z)^{-1/2} D^{1/2} \, \|_{\ell^2} \| \, D^{-1/2} L A^{-1/2} \, \|_{\ell^2} \\ &\quad + \| \, (D + 2Z)^{-1/2} Z D^{-1/2} \, \|_{\ell^2} \| \, D^{1/2} A^{-1/2} \, \|_{\ell^2} \Big)^2 . \end{aligned}$$

Now $\| \, (D+2Z)^{-1/2} D^{1/2} \, \|_{\ell^2} = 1$, since $Z = \ell^t d^{-1} \ell$ is symmetric, positive semidefinite, while $\| \, D^{-1/2} L A^{-1/2} \, \|_{\ell^2} = \mu_0 \leq C J^2$ and $\| \, D^{1/2} A^{-1/2} \, \|_{\ell^2} \leq C J^2$ using Lemma 5.2 and Theorem 5.1, respectively. Finally, from Lemma 5.3,

$$\| \, (D + 2Z)^{-1/2} Z D^{-1/2} \, \|_{\ell^2} \leq \frac{1}{\sqrt{2}} \| \, D^{-1/2} Z D^{-1/2} \, \|_{\ell^2} = \frac{\mu_1}{\sqrt{2}} \leq C.$$

Combining these estimates, we see that $\mu \leq C J^2$, just as in the basic block symmetric Gauss-Seidel iteration. □

We next formally define algorithms similar to BPX [10] and regular multigrid algorithms based on our decomposition. We will use the ideas of Griebel [15] [14], who characterizes these methods as particular iterations applied to an enlarged, semidefinite system. For each vertex $v_i \in \mathcal{X}_J$, let $L(v_i) \equiv L_i$ denote the vertex level of v_i

and $E(v_i) \equiv E_i$ denote the maximum edge level of any edge in $\mathcal{E}_J$ having v_i as an endpoint. Clearly $1 \leq L_i \leq E_i \leq J$. With each vertex $v_i \in \mathcal{X}_J$, we associate a set of one or more basis functions given by

$$\xi_i^k = \tilde{\mathcal{I}}_k^J \phi_i$$

for $L_i \leq k \leq E_i$. Here $\phi_i \equiv \phi_i^J (= \xi_i^{E_i})$ are just the nodal basis functions for $\mathcal{M}_J$, and $\xi_i^{L_i} \equiv \psi_i^J$ are the usual hierarchical basis functions for $\mathcal{M}_J$. We next define the subspaces $\tilde{\mathcal{V}}_k^J$ by

$$\tilde{\mathcal{V}}_k^J = \text{span}\left\{\xi_i^k\right\}.$$

Note that $\mathcal{V}_k^J \subset \tilde{\mathcal{V}}_k^J$ and

$$\mathcal{M}_J = \tilde{\mathcal{V}}_1^J + \tilde{\mathcal{V}}_2^J + \cdots + \tilde{\mathcal{V}}_J^J.$$

We note that this is *not* a direct sum decomposition as in the case of hierarchical basis methods.

We can now form a block $J \times J$, symmetric, positive semidefinite stiffness matrix $\mathcal{A}$ given by

$$\mathcal{A} = \begin{bmatrix} \mathcal{A}_{11} & \mathcal{A}_{12} & \cdots & \mathcal{A}_{1J} \\ \mathcal{A}_{21} & \mathcal{A}_{22} & & \mathcal{A}_{2J} \\ \vdots & & \ddots & \vdots \\ \mathcal{A}_{J1} & \mathcal{A}_{J2} & \cdots & \mathcal{A}_{JJ} \end{bmatrix} \tag{29}$$

where $\mathcal{A}_{kk}$ involves energy inner products of the ξ_i^k.

If the enlarged finite element system is $\mathcal{A}\mathcal{U} = \mathcal{B}$, then Griebel shows that *any* solution $\mathcal{U}$ of the singular system corresponds to the *unique* finite element solution $u \in \mathcal{M}_J$. Within this framework, the BPX method is just a block Jacobi preconditioner analogous to the Hierarchical Basis preconditioner analyzed in Theorem 5.1, but applied to the matrix $\mathcal{A}$. Similarly, the block symmetric Gauss Seidel Preconditioner, using symmetric Gauss Seidel as an inner iteration as in Theorem 5.4, is just a particular multigrid V-cycle using symmetric Gauss-Seidel as smoother when applied to the matrix $\mathcal{A}$.

In the case of nested meshes, both BPX and regular multigrid V-cycle have been shown to have condition numbers which are bounded independent of the number of levels J [23] [24] [28] [30] [8]. Furthermore, those bounds apply in three as well as two space dimensions. Our analysis of these methods, based on the results of Section 4, is essentially the same as Theorems 5.1 and 5.4, and has neither of these properties.

On the one hand, one should expect these BPX and multigrid like preconditioners to outperform hierarchical basis and hierarchical basis multigrid preconditioners, respectively, since unknowns associated with certain grid points are processed on several levels, rather than only one. On the other hand, our subspace decomposition is based on the approximate pointwise interpolation operators $\tilde{\mathcal{I}}_k^J$ rather than $\mathcal{L}^2$ projections, so it is problematic that these preconditioners could be shown to achieve the same sorts of condition numbers as their counterparts based on nested refinement.

6. Implementation Issues. In this section we discuss several important aspects of the practical implementation of the hierarchical basis multigrid and other hierarchical basis preconditioners. The most difficult issue is that, in typical applications, the stiffness matrix is assembled using the nodal basis functions rather than the hierarchical basis. The reason is that the nodal basis functions have small supports, only three such functions are nonzero in any given element of the fine mesh, and therefore the stiffness matrix represented in the nodal basis is both easy to assemble and very sparse. In contrast, since hierarchical basis functions have supports of varying sizes, depending on the level, the stiffness matrix represented in the hierarchical basis is more complicated to assemble and much less sparse than that of the nodal basis (although better conditioned).

To give a theoretical analysis of the method as in Sections 4 and 5, it is sufficient to assume the linear system is assembled in the hierarchical basis. However, to efficiently implement the method, one must assemble the stiffness matrix with respect to the nodal basis, and then implement the hierarchical basis iteration in an *implicit* fashion. It is through this aspect of their implementation that the connection between hierarchical basis methods and multigrid methods becomes clear. We will give a simple illustration of how this is done, by considering in detail the case of two levels.

We write the linear system $\hat{A}\hat{U} = \hat{F}$, constructed with respect to the standard nodal basis, in block form as

$$\begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} \\ \hat{A}_{21} & \hat{A}_{22} \end{bmatrix} \begin{bmatrix} \hat{U}_1 \\ \hat{U}_2 \end{bmatrix} = \begin{bmatrix} \hat{F}_1 \\ \hat{F}_2 \end{bmatrix}. \tag{30}$$

Here we have assumed a partitioning corresponding to the two level hierarchical decomposition. The system (30) is related to the hierarchical basis system

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} \tag{31}$$

via a nonsingular matrix S which transforms the representation of a finite element function with respect to the hierarchical basis into its representation with respect to a nodal basis. S has the block structure

$$S = \begin{bmatrix} I & 0 \\ R & I \end{bmatrix}. \tag{32}$$

The off-diagonal entries of S are zero except for at most two nonzeroes in each row of R. Let v_i have $L_i = 2$, and let v_ℓ and v_r be the vertex parents of v_i. Then the two nonzero coefficients in R for row i are $S_{ir} = \theta$ and $S_{i\ell} = 1 - \theta$, where θ is the interpolation coefficient defined in (4) and (1). Using (32) we relate the matrices A and $\hat{A}$, the right hand sides F and $\hat{F}$, and solutions U and $\hat{U}$ by

$$\begin{aligned} A &= S^t \hat{A} S \\ F &= S^t \hat{F} \\ \hat{U} &= SU. \end{aligned}$$

Blockwise this gives

$$\begin{aligned} A_{11} &= \hat{A}_{11} + R^t \hat{A}_{21} + \hat{A}_{12} R + R^t \hat{A}_{22} R \\ A_{12} &= \hat{A}_{12} + R^t \hat{A}_{22} \end{aligned}$$

$$
\begin{aligned}
A_{21} &= \hat{A}_{21} + \hat{A}_{22}R \\
A_{22} &= \hat{A}_{22} \\
F_1 &= \hat{F}_1 + R^t\hat{F}_2 \\
F_2 &= \hat{F}_2 \\
\hat{U}_1 &= U_1 \\
\hat{U}_2 &= RU_1 + U_2.
\end{aligned}
$$

A single iteration cycle of the hierarchical basis multigrid method, expressed in the current notation, is given below. First, we approximately solve

$$A_{22}W_2 = F_2 \tag{33}$$

using a symmetric Gauss-Seidel inner iteration. To avoid introducing more notation, we denote the *approximate* solution by W_2. We then form the level-1 hierarchical basis residual $G_1 = F_1 - A_{12}W_2$ as

$$G_1 = (\hat{F}_1 - \hat{A}_{12}W_2) + R^t(\hat{F}_2 - \hat{A}_{22}W_2). \tag{34}$$

Next, we solve

$$A_{11}W_1 = G_1 \tag{35}$$

by a direct method, and set $\hat{W}_1 = W_1$. Next, form the level-2 hierarchical basis residual $G_2 = F_2 - A_{21}W_1 - A_{22}W_2$ as

$$G_2 = \hat{F}_2 - \hat{A}_{21}\hat{W}_1 - \hat{A}_{22}(RW_1 + W_2) \tag{36}$$

Next, we approximately solve

$$A_{22}Z_2 = G_2 \tag{37}$$

using a symmetric Gauss-Seidel inner iteration, denote the approximate solution by Z_2, and set

$$\hat{W}_2 = RW_1 + W_2 + Z_2. \tag{38}$$

Careful inspection of (33)-(38) shows the hierarchical basis method to be closely related to a standard multigrid V-cycle. Equations (33) and (37) correspond to multigrid smoothing steps, except that only the level-2 unknowns (W_2 and Z_2) are smoothed, rather than all unknowns. After the first (pre) smoothing step in (33), the approximate solution with respect to the nodal basis is

$$\begin{bmatrix} \hat{U}_1 \\ \hat{U}_2 \end{bmatrix} \approx \begin{bmatrix} 0 \\ W_2 \end{bmatrix}.$$

In (34), forming the residual vector G_1 consists of forming the fine grid residuals after smoothing with respect to the nodal basis

$$\begin{bmatrix} \hat{F}_1 - \hat{A}_{12}W_2 \\ \hat{F}_2 - \hat{A}_{22}W_2 \end{bmatrix}$$

and making a "fine-to-coarse" grid transfer (multiply by R^t) to compute the residual with respect to the hierarchical basis for the level-1 unknowns. Equation (35) is the

standard multigrid coarse grid correction, done by direct methods for our case $J = 2$, but more generally handled by recursion for $J > 2$. In (36) we update the nodal basis solution as

$$\begin{bmatrix} \hat{U}_1 \\ \hat{U}_2 \end{bmatrix} \approx \begin{bmatrix} \hat{W}_1 \\ RW_1 + W_2 \end{bmatrix}$$

using "coarse-to-fine" grid interpolation (multiply by R), and form the fine grid residual $\hat{G}_2$. After the second (post) smoothing step (37), the approximate solution with respect to the nodal basis is

$$\begin{bmatrix} \hat{U}_1 \\ \hat{U}_2 \end{bmatrix} \approx \begin{bmatrix} \hat{W}_1 \\ \hat{W}_2 \end{bmatrix}.$$

To carry out this iteration, we need the matrices $\hat{A}_{22} = A_{22}$, $\hat{A}_{12}$, and A_{11}, all of which are sparse. The matrices $\hat{A}_{12}$ and $\hat{A}_{22}$ are just blocks of the original stiffness matrix computed with respect to the nodal basis. In the case of nested refinement, A_{11} is just the stiffness matrix formed from energy inner products of the coarse grid nodal basis functions. In our case, A_{11} has a similar sparsity pattern to that of the nested case, but the basis functions involved are the nodal basis functions for the coarse grid interpolated onto the fine grid using (4). However, we never have to explicitly form or compute with these basis functions; rather we compute $\hat{A}$ as in (30), using the usual nodal basis, and form $A_{11} = \hat{A}_{11} + R^t\hat{A}_{21} + \hat{A}_{12}R + R^t\hat{A}_{22}R$.

The matrix R is also sparse; its sparsity structure is known through the vertex parents for each vertex, and the numerical values for each row (θ and $1 - \theta$) can be conveniently stored in one or two linear arrays of order N_J.

For more than two levels, the procedure is implemented recursively, which affects only equation (35). Matrix A_{11} is recursively decomposed into pieces corresponding to coarser levels, but with the same structure as the basic two level matrices described above.

We close this section with a numerical example, using the hierarchical basis multigrid method applied to the Lake Superior example of Section 3. The partial differential equation we solved was

$$\begin{aligned} -\Delta u &= 1 && \text{in } \Omega, \\ u &= 0 && \text{on } \partial\Omega. \end{aligned}$$

The application of Algorithm Coarsen with a target value of 300 vertices in the coarse grid triangulation resulted in eight hierarchical basis levels, which we defined using the function $L(v) \equiv \bar{L}(v)$.

The graphs of the resulting stiffness matrices (similar to A_{11} above, but now generated recursively for each of the hierarchical basis levels) are shown in Figures 10 and 11. As is standard in sparse matrices, each edge in the graph corresponds to a nonzero in the upper triangular part of the stiffness matrix. Perhaps the most striking feature of these graphs is that, with the exception of case of the original mesh, none of the graphs are triangulations. This is partly due to a suboptimal choice of levels, and partly due to edge swapping. For example, if the mesh with 1412 vertices in Figure 7 was one of the meshes (it is not) then the graph of the mesh would be the graph of the stiffness matrix, with the remark that for each quadrilateral element, *both* diagonal edges must be added. In effect, edge swapping creates nonzeroes in the stiffness matrix

connecting all four vertices involved in the swap. This effect propagates through subsequent edge swaps, so the matrix graphs become progressively more distant from the underlying triangulations. The extra edges due to swapping also increase slightly the overall work and storage involved in the algorithm, as noted in Section 3.

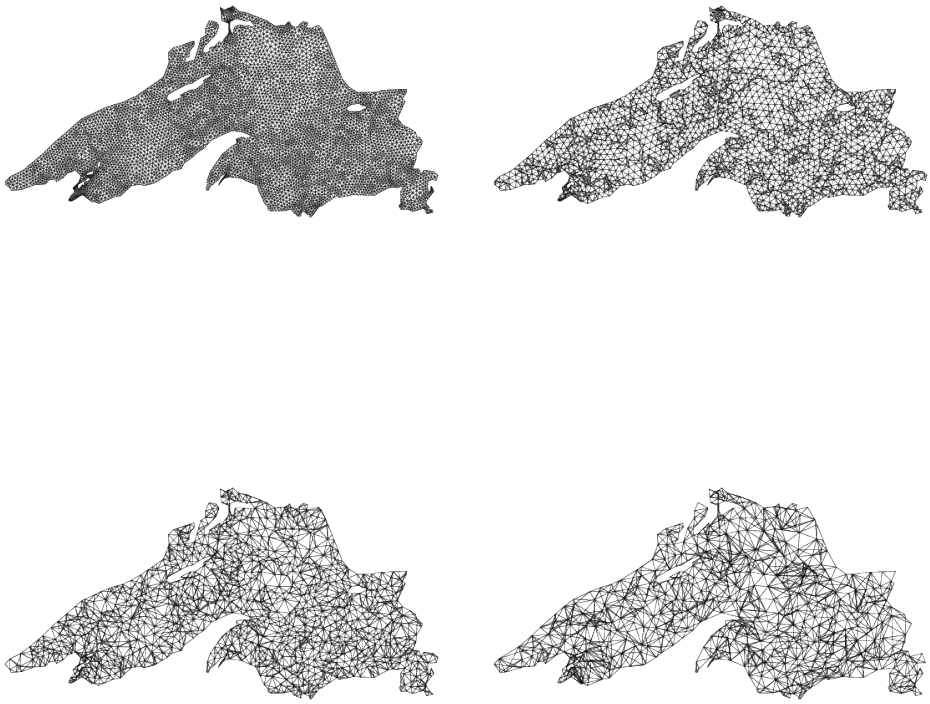

FIG. 10. *Graph of the initial stiffness matrix with 5927 vertices, and graphs of the hierarchical basis matrices with 2195, 1178 and 670 vertices.*

The convergence history is shown on the right in Figure 12. The quantity plotted is

$$\sigma_k = \log \left\{ \frac{\| r_k \|}{\| r_0 \|} \right\},$$

where r_k is the residual at iteration k and $\| \cdot \|$ is the ℓ^2 norm. Standard conjugate gradient acceleration was used. From the data points, we estimate by least squares that the convergence rate is approximately .37, which is fairly typical of this particular iterative method applied to a similar problem on a sequence of refined meshes.

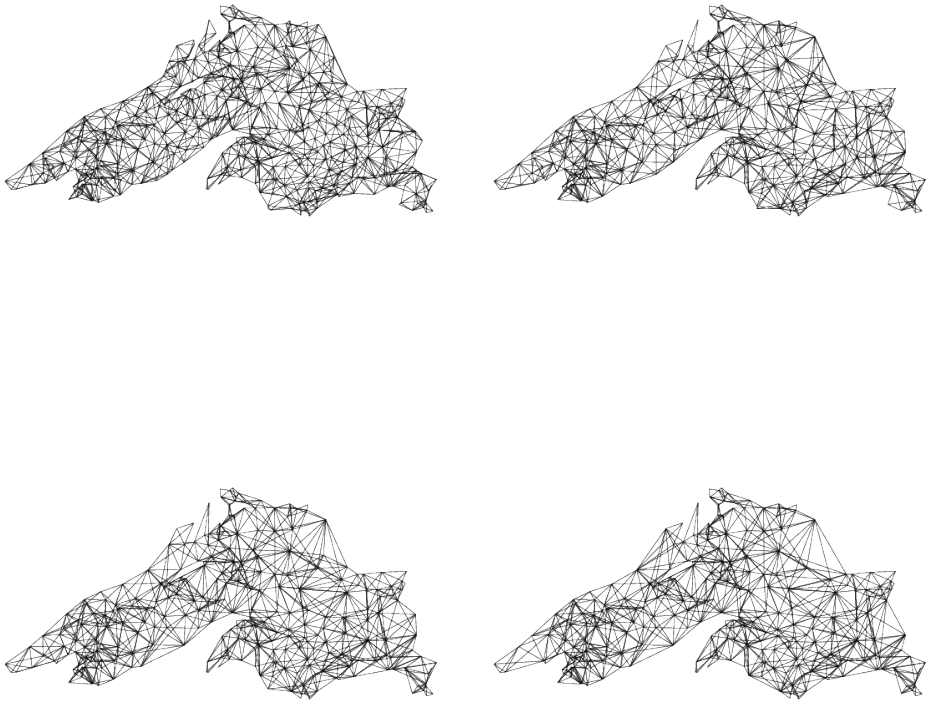

FIG. 11. *Graphs of the hierarchical basis matrices with 426, 337, 309, and 300 vertices.*

[1] R. E. BANK, *Hierarchical preconditioners for elliptic partial differential equations*, in Large Scale Matrix Problems and the Numerical Solution of Partial Differential Equations (J. Gilbert and D. Kershaw, eds.), Oxford University Press, 1994, pp. 121–155.

[2] ———, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations, Users' Guide 7.0*, Frontiers in Applied Mathematics, SIAM, Philadelphia, 1994.

[3] R. E. BANK AND T. F. DUPONT, *Analysis of a two level scheme for solving finite element equations*, Tech. Rep. CNA-159, Center for Numerical Analysis, University of Texas at Austin, 1980.

[4] R. E. BANK, T. F. DUPONT, AND H. YSERENTANT, *The hierarchical basis multigrid method*, Numer. Math., 52 (1988), pp. 427–458.

[5] R. E. BANK, A. H. SHERMAN, AND A. WEISER, *Refinement algorithms and data structures for regular local mesh refinement*, in Scientific Computing (Applications of Mathematics and Computing to the Physical Sciences) (ed. R. S. Stepleman), North Holland, 1983, pp. 3–17.

[6] R. E. BANK AND J. XU, *The hierarchical basis multigrid method and incomplete LU decomposition*, in Proceedings of Seventh International Conference on Domain Decomposition. (ed. D. Keyes and J. Xu), AMS, Providence, Rhode Island, 1994, pp. 163–173.

[7] ———, *A hierarchical basis multigrid method for unstructured meshes*, in Fast Solvers for Flow Problems. Proceedings of the Tenth GAMM-Seminar Kiel, (Notes on Numerical Mathematics, vol. 49, W. Hackbusch and G. Wittum, eds.), Vieweg-Verlag, Braunschweig, 1995, pp. 1–13.

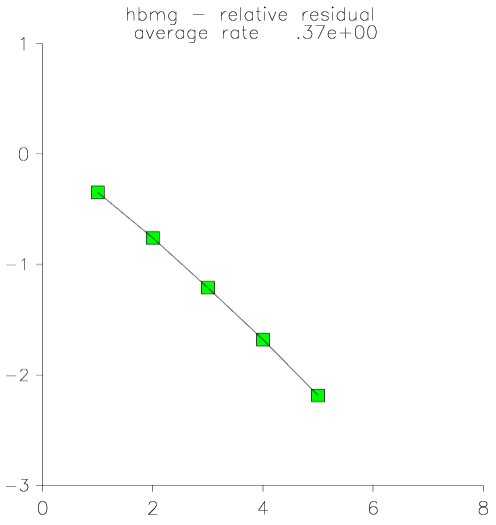

FIG. 12. *Convergence history for HBMG with conjugate gradient acceleration.*

[8] F. BORNEMANN AND H. YSERENTANT, *A basic norm equivalence for the theory of multigrid methods*, Numer. Math., 64 (1993), pp. 455–476.

[9] J. BRAMBLE, J. PASCIAK, AND J. XU, *The analysis of multigrid algorithms with non-imbedded spaces or non-inherited quadratic forms*, Math. Comp., 56 (1991), pp. 1–43.

[10] J. H. BRAMBLE, J. E. PASCIAK, AND J. XU, *Parallel multilevel preconditioners*, Math. Comp., 55 (1990), pp. 1–22.

[11] T. F. CHAN AND B. F. SMITH, *Domain decomposition and multigrid algorithms for elliptic problems on unstructured meshes*, in Proceedings of Seventh International Conference on Domain Decomposition. (ed. D. Keyes and J. Xu), AMS, Providence, Rhode Island, 1994, pp. 175–189.

[12] V. EIJKHOUT AND P. VASSILEVSKI, *The role of the strengthened Cauchy-Buniakowskii-Schwarz inequality in multilevel methods*, SIAM Review, 33 (1991), pp. 405–419.

[13] A. GEORGE AND J. LIU, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice Hall, Englewood Cliffs, NJ, 1981.

[14] M. GRIEBEL, *Multilevel algorithms considered as iterative methods on semidefinite systems*, SIAM J. Sci. Comput., 15 (1994), pp. 547–565.

[15] ———, *Multilevelmethoden als Interationsverfahren Erzeugendensystemen*, B. G. Teubner, Stuttgart, 1994.

[16] M. GRIEBEL AND P. OSWALD, *Remarks on the abstract theory of additive and multiplicative Schwarz algorithms*, Tech. Rep. TUM-19314, Institut für Informatik, Technische Universität München, 1993.

[17] W. HACKBUSCH, *Multigrid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[18] W. HACKBUSCH AND G. WITTUM, EDS, *Incomplete Decompositions - Algorithms, Theory and Applications*, Notes on Numerical Fluid mechanics, vol. 41, Vieweg, Braunschweig, 1993.

[19] R. H. W. HOPPE AND R. KORNHUBER, *Adaptive multilevel methods for obstacle problems*, SIAM J. Numer. Anal., (to appear).

[20] R. KORNHUBER, *Monotone multigrid methods for variational inequalities I*, Numer. Math., (to appear).

[21] D. J. MAVRIPLIS, *Multigrid techniques for unstructured meshes*, Tech. Rep. 95-27, Institute for Computer Applications in Science and Engineering, Hampton, Virginia, 1995.

[22] W. F. MITCHELL, *A comparison of adaptive refinement techniques for elliptic problems*, ACM Trans. Math. Soft., 15 (1989), pp. 326–347.

[23] P. OSWALD, *On discrete norm estimates related to multigrid preconditioners in the finite element method*, in Proceedings of the International Conference on Constructive Theory of Functions, Varna, 1991.

[24] ———, *Multilevel Finite Element Approximation. Theory and Applications*, B. G. Teubner, Stuttgart, 1994.
[25] M. C. Rivara, *Algorithms for refining triangular grids suitable for adaptive and multigrid techniques*, Int. J. Numer. Meth. Eng., 20 (1984), pp. 745–756.
[26] D. J. Rose, *A graph theoretic study of the numeric solution of sparse positive definite systems*, in Graph Theory and Computing, Academic Press, New York, 1972.
[27] J. Xu, *Theory of Multilevel Methods*, PhD thesis, Cornell University. Report AM-48, Penn State, 1989.
[28] ———, *Iterative methods by space decomposition and subspace correction*, SIAM Review, 34 (1992), pp. 581–613.
[29] H. Yserentant, *On the multi-level splitting of finite element spaces*, Numer. Math., 49 (1986), pp. 379–412.
[30] ———, *Old and new convergence proofs for multigrid methods*, in Acta Numerica, Cambridge University Press, 1992.
[31] S. Zhang, *Multilevel Iterative Techniques*, PhD thesis, Pennsylvania State University, Department of Mathematics Report 88020, 1988.