MESH SMOOTHING USING A POSTERIORI ERROR ESTIMATES

RANDOLPH E. BANK* AND R. KENT SMITH†

Abstract. We develop a simple mesh smoothing algorithm for adaptively improving finite element triangulations. The algorithm makes use of a posteriori error estimates which are now widely used in finite element calculations. In this paper, we derive the method, present some numerical illustrations, and give a brief analysis of the issue of uniqueness.

Key words. moving finite elements, adaptive refinement, a posteriori error estimates.

AMS subject classifications. 65M50, 65N50

1. Introduction. In this paper, we consider an algorithm for optimizing the placement of nodes for triangulations used in finite element calculations. The problem we address here is fairly simple: given a finite element mesh $\mathcal{T}_h$ with a *fixed* connectivity structure, and the finite element solution u_h associated with this triangulation, we seek an algorithms to adjust the positions of the vertices which will result in minimum error. That is, we seek a new triangulation $\bar{\mathcal{T}}_h$ with the same topology as $\mathcal{T}_h$, such that the error in the new solution $\bar{u}_h$ will be minimized.

The algorithms this paper draw upon ideas from three related fields- general mesh generation, [11], [13], [23], [14], [24], [12], [15], [16] moving meshes [1], [2], [3], [18], [20], [19], [17], [4], and adaptive local mesh refinement using a posteriori error estimates [10], [21], [7], [9], [6], [5], [26], [25], [22]. Among the problems in the area of general mesh generation is that of computing a triangulation from some description of the boundary of the region, a set of coordinates (x_i, y_i) describing the region, or some mixture of the two. Among the techniques studied there are smoothing techniques, notably Laplacian smoothing [15],[11], in which vertex locations are sequentially updated in a Gauss-Seidel like fashion. Our algorithms also follow this Gauss-Seidel strategy, and are closest to the smoothing algorithm used by the mesh generator *TRIGEN* in the *PLTMG* package, which optimizes a geometric quality function. This algorithm is briefly described in Section 2, since the quality function plays an important role in our subsequent development. Through our use of a posteriori error estimates, the computed solution u_h also plays an important part in our smoothing algorithm. This aspect is similar to the work of D'Azevedo and Simpson [12], and Dyn *et al* [14] who compute the optimal shape of elements for a given function.

In the area of moving finite element methods, there are many schemes for evolving a triangulation in time, through the solution of systems of differential equations, direct use of a posteriori error estimates, and various other optimization schemes. Our algorithms might be applied in many of the same situations and make use of many of the same things, e.g., a posteriori error estimates, but we use these estimates in a somewhat non standard way. In particular, our methods don't make direct use of the details of the partial differential equation or the a posteriori error estimates. This should make the method easy to implement in a broad range of applications. Indeed, in Section 4 we first develop the method using interpolation errors, and defer the introduction of a posteriori error estimates until Section 5.

*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.

†AT&T Bell Laboratories, Murray Hill, New Jersey 07974

A posteriori error estimates are now widely used in adaptive methods for solving partial differential equations, and there are now many good a posteriori error estimators available. One of the nice features of our smoothing algorithm is that it is not linked to a particular a posteriori error estimator. In the case of linear elements, which is what we mainly consider here, we require that the error estimator yield a (local) error indicator as a piecewise quadratic polynomial, which can be continuous or discontinuous. Our basic strategy is to use whatever a posteriori error estimate is available to locally estimate the second derivatives of the true solution u, and then holding these second derivatives fixed, solve local minimization problems for the locations of the mesh points. If one were to generalize to piecewise polynomials of degree p, one would require an a posterior error estimator which yields a continuous or discontinuous piecewise polynomial of degree $p+1$.

We don't believe that a mesh smoothing algorithm, such as the one developed here, can be the sole basis of an effective adaptive method. The main shortcoming of such methods is that they do not change the topology (connectivity) in the triangulation, and the ability make such changes is often critical to the success of an adaptive method. We do believe that our method can be successfully employed in conjunction with methods which allow topology changes, such as adaptive local mesh refinement (and unrefinement). In such schemes, one begins with a coarse triangulation, and generates a sequence of nested, refined triangulations through a process of local mesh refinement. There are several widely used schemes, some dividing an given element into four similar elements by pairwise connecting the midpoints of the three edges, and others which divide an element into two triangles by bisecting a single edge according to some geometric criteria. Such schemes can easily create highly nonuniform meshes adapted to the singularities of a given solution. However, their chief drawback is that, being based on element refinement, exact locations of grid points depend on the geometry of the father element (e.g., the midpoint of an edge), and inductively, on the structure of the initial coarse triangulation. Often this presents no problem, but in other cases (e.g. that of a steep front), it might be the case that moving the grid points a little (to better align the mesh with the front) can reduce the error substantially. It is in such situations that we think mesh smoothing algorithms of the type developed here can make a difference. That is, one should rely on other methods, such as refinement and unrefinement to create a mesh with a proper density of mesh points and a reasonable topology, and then use the smoothing algorithm to "fine tune" the mesh. The mesh points won't generally move very far, and only a few sweeps should be required, but the effect of these small changes could be dramatic.

Although we focus exclusively on two dimensional triangular meshes, we expect that our basic approach to smoothing should apply to other situations, such as quadrilateral meshes in two dimensions or tetrahedral and other types of meshes in three dimensions. However, it is also clear that many important details will vary with each individual situation, and remain to be worked out.

The remainder of this paper is organized as follows. In Section 2, we describe a smoothing algorithm based on the geometric quality of the elements. In Section 3, we extend our smoothing algorithm to minimizing the error in piecewise linear interpolation. In Section 4, we show how to replace interpolation errors with a posteriori error estimates. In Section 5, we present some numerical illustrations of the smoothing procedure, while in Section 6, we consider the mathematical issue of uniqueness of the solutions of the local problems in the smoothing algorithm.

2. A Smoothing Algorithm Based on Element Geometry. In this section we consider a method for placing nodes based on optimizing a predefined quality function. Consider the element t shown in Figure 1 with vertices $\nu_i = (x_i, y_i)$, $1 \leq i \leq 3$ and area $|t|$. The vectors

$$\ell_1 = \begin{bmatrix} x_3 - x_2 \\ y_3 - y_2 \end{bmatrix}, \quad \ell_2 = \begin{bmatrix} x_1 - x_3 \\ y_1 - y_3 \end{bmatrix}, \quad \ell_3 = \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix}$$

are tangent vectors with counterclockwise orientation.

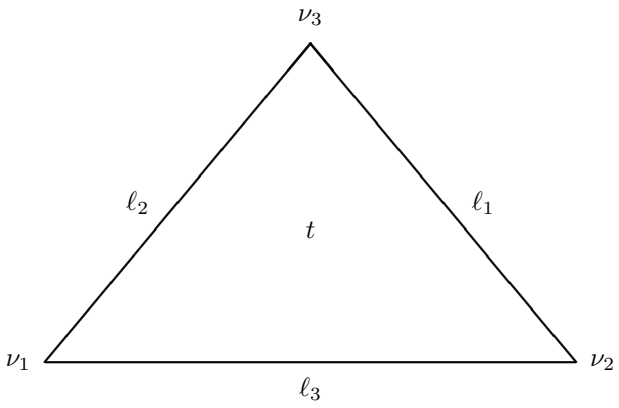

FIG. 1. *Local labels and orientation of triangle t.*

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

$$q(t) = \frac{4\sqrt{3}|t|}{|\ell_1|^2 + |\ell_2|^2 + |\ell_3|^2}. \tag{1}$$

The function $q(t)$ is normalized to equal one for an equilateral triangle and to approach zero for triangles with small angles. In particular, $q(t)$ is independent of the size of t. Note that

$$2|t| = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1), \tag{2}$$

provided the vertices are oriented counterclockwise. This is a very convenient formula to use for computing $|t|$; should a triangle become reoriented during our procedure and the vertices become ordered clockwise, the continued use of (2) will yield a negative area, signaling the reorientation.

To understand the geometric meaning of $q(t)$ in somewhat more detail, without loss in generality, we assume for the moment that $\nu_1 = (0, 0)$, $\nu_2 = (1, 0)$, and $\nu_3 = (x, y)$ with $y \geq 0$, and consider the dependence of the $q(t)$ on the location of the vertex ν_3. Noting that $0 \leq q(t) \leq 1$, we seek the set of points (x, y), for which $q(t) = \alpha$. From (1)

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

This can be manipulated to the form

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

which can be seen to be the equation of a circle.

For $\alpha = 1$, one can see that 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 $q(t) = 1$. 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. Some examples are shown in Figure 2.

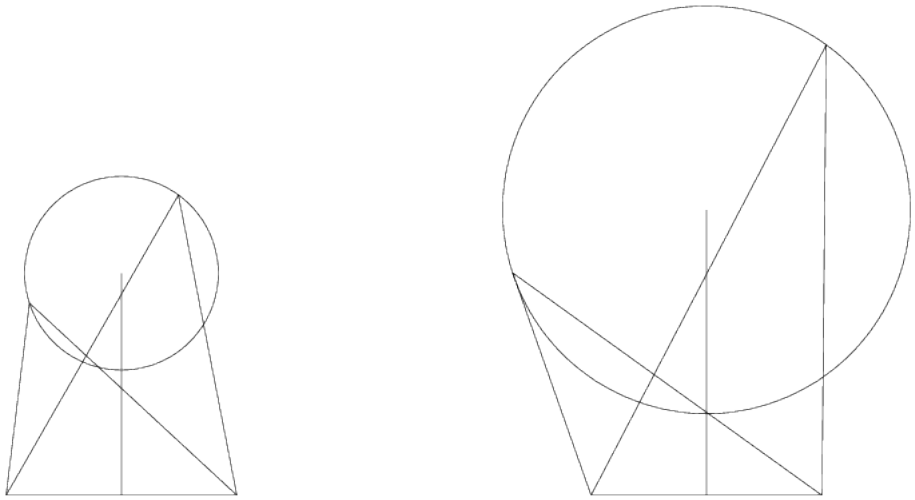

FIG. 2. *Two triangle geometries with quality* $q(t) = 0.9$ *(left) and* $q(t) = 0.7$ *(right).*

Let $\mathcal{T}$ be a fixed triangulation of the domain Ω. For purposes of mesh smoothing, the vertices of the triangulation will be decomposed into the direct sum of three sets, those with 0, 1, or 2 degrees of freedom with respect to vertex placement, which we will call *corners, boundary/interface*, and *interior* vertices, respectively. Roughly speaking, a corner is a vertex critical to defining the geometry of the region, the boundary conditions, the interfaces, etc., whose movement would compromise the integrity of the domain in some way. Such vertices should remain fixed. Boundary/interface points are those points lying along boundaries and interfaces and are constrained to move only along the boundary or interface. All other points are interior vertices, and will have the ability to move in all directions. It should be noted that to some extent the decision as to whether a given vertex should have 0, 1, or 2 degrees of freedom with respect to placement is governed by user and the particular circumstances of the application.

Let $\mathcal{F}$ be a family of triangulations of Ω with a fixed topology; that is, members of $\mathcal{F}$ can differ from one another only by the positions of the vertices in the mesh, but not by their connectivity structures. Additionally, the constraints for corner and boundary/interface vertices should be applied to each member of the family $\mathcal{F}$. Given such a family of triangulations, one can seek the solution of the optimization problem: find a triangulation $\mathcal{T} \in \mathcal{F}$ such that

$$\min_{t \in \mathcal{T}} q(t) = \max_{\mathcal{T}' \in \mathcal{F}} \min_{t \in \mathcal{T}'} q(t). \tag{4}$$

Our algorithm for computing a (generally nonunique) triangulation $\mathcal{T}$ is an iterative Gauss-Seidel like method, in which we sweep through the vertices, locally optimizing the position of a single vertex holding all others fixed. Assume for the moment that the local problems can be solved. Then the function

$$\min_t q(t)$$

can only be increased or unchanged by the solution of each local problem. As a practical matter, this (outer) Gauss-Seidel iteration converges very quickly in its initial stages, and very good triangulations can be obtained by very few sweeps. This makes the algorithm very attractive, provided that the local problems can be solved efficiently.

We now consider the solution of these local problems. We shall describe the case when a given vertex $\nu_i = (x_i, y_i)$ is an interior vertex; for boundary/interface vertices we can apply essentially the same algorithms, but with the appropriate constraints. It should be clear that moving the vertex ν_i can effect the quality $q(t)$ of only those triangles having ν_i as a vertex. Thus we define Ω_i as the union of those elements having ν_i as a vertex.

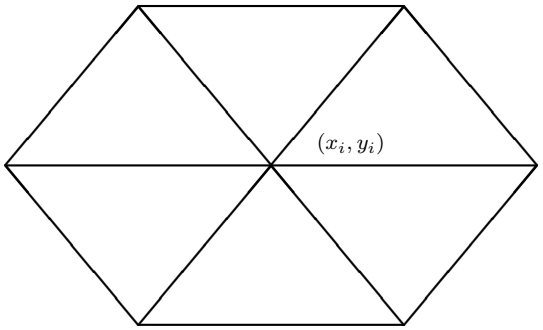

FIG. 3. *The region* Ω_i.

Suppose that initially we compute the qualities of the elements in Ω_i, and that

$$\hat{\alpha} = q(\hat{t}) = \min_{t \in \Omega_i} q(t).$$

Suppose the vertices of $\hat{t}$ are denoted $\hat{\nu}_a, \hat{\nu}_b, \nu_i$ in counterclockwise order. Note that $\hat{\nu}_a$ and $\hat{\nu}_b$ will remain fixed as the position of ν_i is optimized. Let $\hat{\nu}_i = (\hat{x}_i, \hat{y}_i)$ denote the point such that $\hat{\nu}_a, \hat{\nu}_b, \hat{\nu}_i$ are the vertices of an equilateral triangle given in counterclockwise order. As we move along the straight line segment connecting (x_i, y_i) and $(\hat{x}_i, \hat{y}_i)$ the quality of $\hat{t}$ will increase monotonically from $\hat{\alpha}$ to 1. On the other hand, the qualities of some other triangles in Ω_i are likely to decrease along this line; in extreme cases, $(\hat{x}_i, \hat{y}_i)$ may lie outside Ω_i and then some elements will have their qualities reduced to zero and then be reoriented; this is easily accounted for in practice if (2) is used to compute the area; if a triangle becomes reoriented the quality will become negative. In any event, one can employ a simple line search in which we find $0 \leq \theta \leq 1$ such that at the point $(\theta x_i + (1-\theta)\hat{x}_i, \theta y_i + (1-\theta)\hat{y}_i)$, $\hat{t}$ and at least one other element in Ω_i have equal qualities, and the remaining elements have better qualities. By replacing (x_i, y_i) with $(\theta x_i + (1-\theta)\hat{x}_i, \theta y_i + (1-\theta)\hat{y}_i)$, the minimum

quality of elements contained in Ω_i will generally increase (or at worst remain the same if $\theta = 1$). Sweeping through the mesh in this fashion will generally improve the overall quality of the triangulation.

A more sophisticated version of this procedure, which usually avoids the necessity of a line search, is to find two distinct elements, $\hat{t}$ and $\bar{t}$ satisfying

$$q(\hat{t}) = \min_{t \in \Omega_i} q(t)$$
$$q(\bar{t}) = \min_{\substack{t \in \Omega_i \\ t \neq \hat{t}}} q(t).$$

These are the two lowest quality triangles in Ω_i. There is a unique point (x_i', y_i') where the triangles corresponding to $\hat{t}$ and $\bar{t}$ with (x_i, y_i) replaced by (x_i', y_i') will have equal qualities $\alpha' \geq \hat{\alpha}$. This point is characterized as a point of tangency of the circles for the level curve α' for the two triangles, and can be computed directly from the geometry of the vertices which remain fixed in $\hat{t}$ and $\bar{t}$. If qualities of all other triangles are greater than α', then this is the exact solution of the local optimization problem; in practice this is almost always the case. When it is not the case, and the quality of some other triangle becomes less than α', one can then use a line search along the line segment $(\theta x_i + (1-\theta)x_i', \theta y_i + (1-\theta)y_i')$ as in the previous discussion.

3. A Smoothing Algorithm based on Local Interpolation Errors. In this section we consider a method for placing nodes based on approximately minimizing

$$\min_{\mathcal{T} \in \mathcal{F}} \int_\Omega |\nabla(u - u_L)|^2 \, dx. \tag{5}$$

Here $\mathcal{F}$ is the family of triangulations of fixed connectivity described in Section 2, $u \in \mathcal{H}^1(\Omega)$ is a given function, and u_L is the continuous piecewise linear interpolant of u defined relative to the triangulation $\mathcal{T} \in \mathcal{F}$. Since (5) is in general too expensive to solve, we shall instead consider algorithms for approximately minimizing

$$\min_{\mathcal{T} \in \mathcal{F}} \int_\Omega |\nabla(u_Q - u_L)|^2 \, dx, \tag{6}$$

where u_Q is the continuous piecewise quadratic interpolant of u. As will be seen below, we are effectively assuming that for each $t \in \mathcal{T}$, the second derivatives of u are constant.

We now consider the function $\nabla(u_Q - u_L)$ in some detail. Our discussion will adopt and extend the notation of Figure 1. Let c_i, $1 \leq i \leq 3$ denote the piecewise linear nodal basis functions (barycentric coordinates) on a given element t. Recall the defining relation

$$c_i(x_j, y_j) = \delta_{ij}.$$

The piecewise linear interpolant u_L on element t can be expressed as

$$u_L = u(x_1, y_1)c_1(x, y) + u(x_2, y_2)c_2(x, y) + u(x_3, y_3)c_3(x, y)$$

Let M_t denote the 2×2 matrix of second derivatives on t given by

$$M_t = -\frac{1}{2} \begin{bmatrix} u_{xx} & u_{xy} \\ u_{xy} & u_{yy} \end{bmatrix}. \tag{7}$$

We assume that the matrix M_t is constant on t. We define the quadratic "bump functions" b_i for element t by

$$\begin{aligned} b_1(x,y) &= c_2(x,y)\, c_3(x,y), \\ b_2(x,y) &= c_3(x,y)\, c_1(x,y), \\ b_3(x,y) &= c_1(x,y)\, c_2(x,y). \end{aligned}$$

(This differs from the standard definition by a factor of 4.) Then the piecewise quadratic interpolant u_Q can be expressed as

$$u_Q = u_L + \ell_1^t M_t \ell_1\, b_1(x,y) + \ell_2^t M_t \ell_2\, b_2(x,y) + \ell_3^t M_t \ell_3\, b_3(x,y).$$

If the second derivatives of u are not constant in t, we must replace $\ell_1^t M \ell_1$ with the undivided difference quotient

$$4u(\hat{x}_1, \hat{y}_1) - 2u(x_2, y_2) - 2u(x_3, y_3),$$

where $\hat{x}_1 = (x_2 + x_3)/2$ and $\hat{y}_1 = (y_2 + y_3)/2$. Similar replacements are used for $\ell_2^t M \ell_2$ and $\ell_3^t M \ell_3$.

Using these definitions, we can see that

$$\int_t |\nabla(u_Q - u_L)|^2 \, dx = v^t B v,$$

where

$$v = \begin{bmatrix} \ell_1^t M_t \ell_1 \\ \ell_2^t M_t \ell_2 \\ \ell_3^t M_t \ell_3 \end{bmatrix},$$

and

$$B_{ij} = \int_t \nabla b_i \cdot \nabla b_j \, dx.$$

By direct calculation, we find

$$B = \frac{1}{48|t|} \begin{bmatrix} \ell_1^t \ell_1 + \ell_2^t \ell_2 + \ell_3^t \ell_3 & 2\ell_1^t \ell_2 & 2\ell_1^t \ell_3 \\ 2\ell_2^t \ell_1 & \ell_1^t \ell_1 + \ell_2^t \ell_2 + \ell_3^t \ell_3 & 2\ell_2^t \ell_3 \\ 2\ell_3^t \ell_1 & 2\ell_3^t \ell_2 & \ell_1^t \ell_1 + \ell_2^t \ell_2 + \ell_3^t \ell_3 \end{bmatrix}.$$

It is well known that the matrix B is positive definite and comparable to its diagonal. Note that the diagonal entries of B are constant and can be expressed as $1/(4\sqrt{3}q(t))$, where $q(t)$ is the quality measure defined in (1). If we use this diagonal approximation for B, then we have

$$\int_t |\nabla(u_Q - u_L)|^2 \, dx \approx \frac{(\ell_1^t M_t \ell_1)^2 + (\ell_2^t M_t \ell_2)^2 + (\ell_3^t M_t \ell_3)^2}{4\sqrt{3} q(t)}. \tag{8}$$

For convenience, we define the function $s(t)$ by

$$s(t) = (\ell_1^t M_t \ell_1)^2 + (\ell_2^t M_t \ell_2)^2 + (\ell_3^t M_t \ell_3)^2. \tag{9}$$

Notice that on the right hand side of (8), the quality function $q(t)$ serves as a natural "barrier function" which will help prevent the triangle t from becoming degenerate as

the vertices of the triangulation are moved. On the other hand, the numerator $s(t)$ contains information about the size and directionality of the second derivatives of u.

Our smoothing algorithm for solving (6) is analogous to that given in Section 2 in that we will sweep through the vertices, locally optimizing the location of one vertex while holding the others fixed. As before, we partition the vertices into corners, which aren't allowed to move, boundary/interface vertices which can move along one dimensional manifolds, and interior vertices, which can move in all directions. As in Section 2, we focus on the procedure for updating interior vertices, as the boundary/interface case is just a constrained version of that algorithm.

The local problem for vertex $\nu_i = (x_i, y_i)$ has the form

$$\min_{x_i, y_i} \sum_{t \in \Omega_i} \frac{s(t)}{q(t)}. \tag{10}$$

Since we assume that M_t is constant, $s(t)$ is a quartic polynomial in x_i and y_i, while $q(t)$ is a rational function of quadratic polynomials. Thus (10) can be minimized in a straightforward fashion using a damped Newton's method. Although there are certain cases when the barrier function is canceled by terms in $s(t)$ (see Section 6), it is clear from the form of (10) that generally the functions $q(t)$ form a barrier which makes certain that the minimizing point (x_i^*, y_i^*) remains within Ω_i. If Ω_i is not convex, the barrier functions constrain the point (x_i^*, y_i^*) to lie in the subregion of Ω_i which is visible from all points on the boundary of Ω_i.

There is a potential for nonuniqueness of the solution; this could occur for example, if u is linear in Ω_i and $M_t = 0$. Unfortunately, this is not the only sort of pathology that can arise; we will explore this aspect of the problem in more detail in Section 6. For now, we note that one possible remedy for degeneracy would be to add a regularizing term like

$$\min_{x_i, y_i} \sum_{t \in \Omega_i} \frac{s(t) + \rho s_0(t)}{q(t)}, \tag{11}$$

where

$$s_0 = \{(\ell_1^t \ell_1)^2 + (\ell_2^t \ell_2)^2 + (\ell_3^t \ell_3)^2\},$$

and ρ is a nonnegative penalty parameter. In practice, we have found it more expedient to rely on the damping/line search to keep (x_i, y_i) well within Ω_i, and in the rare event of exact singularity of the Hessian matrix, to simply skip that point for the given sweep. Following the Newton sweeps, we can make $1-2$ sweeps of the type described in Section 2 to improve the geometry of the mesh if needed.

Before leaving this section, we note that global smoothing strategies can also be developed along these lines. For a given triangulation, a global set of coupled equations for each vertex is constructed from (6) and (8). The domain boundaries are described by a set of constraints that restrict the motion to one dimensional manifolds. These constraints could also provide a means to describe moving boundaries. The resulting system of coupled nonlinear equations could be solved using a damped Newton's method. A line search would insure that the integrity of the original triangulation is preserved. This could be advantageous when efficient sparse matrix solution methods are available. For example, the same sparse matrix data structures and solution methods can be used for mesh optimization that are used in finite element solution methods. However, the question of uniqueness is far more complex for global methods than for the local problems we have been considering.

4. A Smoothing Algorithm based on A Posteriori Error Estimates. In this section we consider a method for placing nodes based on approximately minimizing

$$\min_{\mathcal{T}\in\mathcal{F}} \int_\Omega |\nabla(u-u_h)|^2\,dx. \tag{12}$$

Here $\mathcal{F}$ is the family of triangulations of fixed connectivity described in Section 2, $u \in \mathcal{H}^1(\Omega)$ is the solution a partial differential equation, and u_h is the continuous piecewise linear finite element approximation of u relative to the triangulation $\mathcal{T} \in \mathcal{F}$. The algorithm we will develop will be quite close to that of Section 3, except that $u - u_h$ will be approximated by an a posteriori error estimate.

Since our goal here is not to develop new a posteriori error estimates, but rather to explain how they can be applied in this context, for expositional convenience we will consider only the standard self adjoint, positive definite, elliptic equation

$$\begin{aligned} -\nabla(a\nabla u) + bu &= f \qquad \text{for } x \in \Omega, \\ u &= 0 \qquad \text{for } x \in \partial\Omega, \end{aligned} \tag{13}$$

with $a > 0$, $b \geq 0$, and f smooth functions. We note that a posterior error estimates have been developed for much broader classes of linear and nonlinear equations.

The usual weak formulation (13) is: find $u \in \mathcal{H}_0^1(\Omega)$ such that

$$a(u,v) = (f,v)$$

for all $v \in \mathcal{H}_0^1(\Omega)$, where

$$\begin{aligned} a(u,v) &= \int_\Omega a\nabla u \cdot \nabla v + buv\,dx, \\ (f,v) &= \int_\Omega fv\,dx. \end{aligned}$$

Let $\mathcal{S}_h \subset \mathcal{H}_0^1(\Omega)$ be the usual space of continuous piecewise linear polynomials associated with the triangulation $\mathcal{T} \in \mathcal{F}$. Then the finite element approximation is: find $u_h \in \mathcal{S}_h$ such that

$$a(u_h, v) = (f, v)$$

for all $v \in \mathcal{S}_h$.

Our a posteriori error estimate requires the solution of a local Neumann problem in each element $t \in \mathcal{T}$. We will not give a derivation or an analysis of this estimate, as it is rather lengthy and not especially relevant to our current discussion. The interested reader is referred to [10] [7] for details. In any event, let $\mathcal{B}_t$ be the set of quadratic polynomials on t which are zero at the vertices of t. This is a three dimensional space spanned by the quadratic bump functions $b_i(x,y)$, $1 \leq i \leq 3$ defined in Section 3.

Suppose that $t \cap \partial\Omega = \emptyset$. Then the local a posteriori error estimate is computed for element t by solving the problem: find $e_t \in \mathcal{B}_t$ such that

$$a(e_t, v)_t = (f,v)_t - a(u_h, v)_t + \langle \left[\frac{\partial u_h}{\partial n}\right]_A, v\rangle_{\partial t} \tag{14}$$

for all $v \in \mathcal{B}_t$. Here

$$a(e_t, v)_t = \int_t a\nabla e_t \cdot \nabla v + be_t v \, dx,$$
$$(f, v)_t = \int_t fv \, dx,$$
$$\langle \left[\frac{\partial u_h}{\partial n}\right]_A, v\rangle_{\partial t} = \int_{\partial t} \left[\frac{\partial u_h}{\partial n}\right]_A v \, ds,$$

and $[\partial u_h/\partial n]_A$ is the average normal derivative of u_h along the boundary of element t.

Using integration by parts, one can also write (14) as: find $e_t \in \mathcal{B}_t$ such that

$$a(e_t, v)_t = (r, v)_t + \frac{1}{2}\langle \left[\frac{\partial u_h}{\partial n}\right]_J, v\rangle_{\partial t}$$

for all $v \in \mathcal{B}_t$, where r is the residual $r = f + \nabla(a\nabla u_h) - bu_h$, and $[\partial u_h/\partial n]_J$ is the *jump* in normal derivative in u_h along ∂t. In either formulation, (14) represents a 3×3, symmetric, positive definite set of linear equations to be solved in each element. For elements with edges on the boundary $\partial\Omega$, the condition $e_t = 0$ on $\partial\Omega$ should be imposed, with the corresponding modification of the space $\mathcal{B}_t$ and the boundary inner product in (14).

Globally, the function e_h is given by

$$e_h = \sum_{t \in \mathcal{T}} e_t,$$

(where it is understood that $e_t(x, y) = 0$ for $(x, y) \notin t$), so e_h is a *discontinuous* piecewise quadratic polynomial which is zero at the vertices of the triangulation $\mathcal{T}$. We note that, unlike the case of interpolation errors which express the true local error, most a posteriori error estimates yield only local error indicators. That is, while $\| u - u_h \|_{\mathcal{H}^1(\Omega)} \approx \| e_h \|_{\mathcal{H}^1(\Omega)}$ globally, this is not known to be true elementwise.

To see how such an error indicator can be used for mesh smoothing, we consider a particular element $t \in \mathcal{T}$ and adopt the notation used in Figure 1 and Section 3. On element t, our a posteriori error indicator has the form

$$e_t = e_1 b_1(x, y) + e_2 b_2(x, y) + e_3 b_3(x, y), \tag{15}$$

where the e_i were found by solving the 3×3 linear system associated with (14). To use the smoothing algorithm developed in Section 3, we would like to express this in the form

$$e_t = \ell_1^t M_t \ell_1 \, b_1(x, y) + \ell_2^t M_t \ell_2 \, b_2(x, y) + \ell_3^t M_t \ell_3 \, b_3(x, y), \tag{16}$$

where M_t is the 2×2 matrix of (approximate) second derivatives for the solution u given by (7). Equating coefficients in (15)-(16), and using (7), we are led to the 3×3 set of equations

$$\begin{bmatrix} (x_3 - x_2)^2 & 2(x_3 - x_2)(y_3 - y_2) & (y_3 - y_2)^2 \\ (x_1 - x_3)^2 & 2(x_1 - x_3)(y_1 - y_3) & (y_1 - y_3)^2 \\ (x_2 - x_1)^2 & 2(x_2 - x_1)(y_2 - y_1) & (y_2 - y_1)^2 \end{bmatrix} \begin{bmatrix} u_{xx} \\ u_{xy} \\ u_{yy} \end{bmatrix} = -2 \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \tag{17}$$

for the matrix elements of M_t. This system can be solved provided that t is not degenerate ($q(t) \neq 0$). Once computed, we will regard M_t as constant, and then simply apply the smoothing procedure as in Section 3. We note that the critical point of the a posteriori error estimate is that one obtains an expression like (15) for the error indicator in a given element t. Many a posteriori error estimators lead to such expressions, and the one described here also yields such expressions when applied to nonlinear, indefinite, and nonself-adjoint problems.

5. Numerical Illustrations. In this section we present two numerical illustrations of the smoothing algorithms described in Sections 3-4. Both examples were developed using the finite element package PLTMG7.0 [8], which includes these algorithms among its adaptive mesh generation options.

In our first example, we begin with a uniform 17×17 mesh on the unit square $\Omega = (0,1) \times (0,1)$, and adapt it to the function $u = \sqrt{x}$. This function has a singular gradient at $x = 0$, and we expect the mesh to move in response to this singularity.

Our algorithm is exactly that of Section 3. After every four Gauss-Seidel iterations, the quadratic interpolant u_Q is computed for each element in the usual way, using the values of u at the vertices and midpoints of each element. New values for the second derivatives used to define M_t are computed by solving a 3×3 linear system similar to (17), with second differences replacing the e_i, and these values are then used for the next four sweeps. Each time new second derivatives were computed, we evaluated the functional

$$\mathcal{E}^2 = \sum_{t \in \mathcal{T}} \frac{s(t)}{4\sqrt{3}q(t)}, \tag{18}$$

where $s(t)$ is defined in (9) and $q(t)$ is defined in (1).

In Table 1, we summarize the convergence history for 40 iterations in terms of $\mathcal{E}$, and in Figure 1 we show the initial mesh and the smoothed meshes after 4, 8, and 12 iterations.

iteration	$\mathcal{E}$
0	0.679468
4	0.569834
8	0.536575
12	0.523275
16	0.515238
20	0.511084
24	0.509720
28	0.510352
32	0.511416
36	0.512729
40	0.514082

TABLE 1
Convergence history for the first example.

As can be seen from the Table 1, the convergence of the the Gauss-Seidel sweeps is initially very rapid, and then slows down. In Figure 4, we can see that the mesh changes substantially during the first four sweeps, and much less in later iterations. In this example, after 12 iterations, neither $\mathcal{E}$ or the mesh changes very much. We

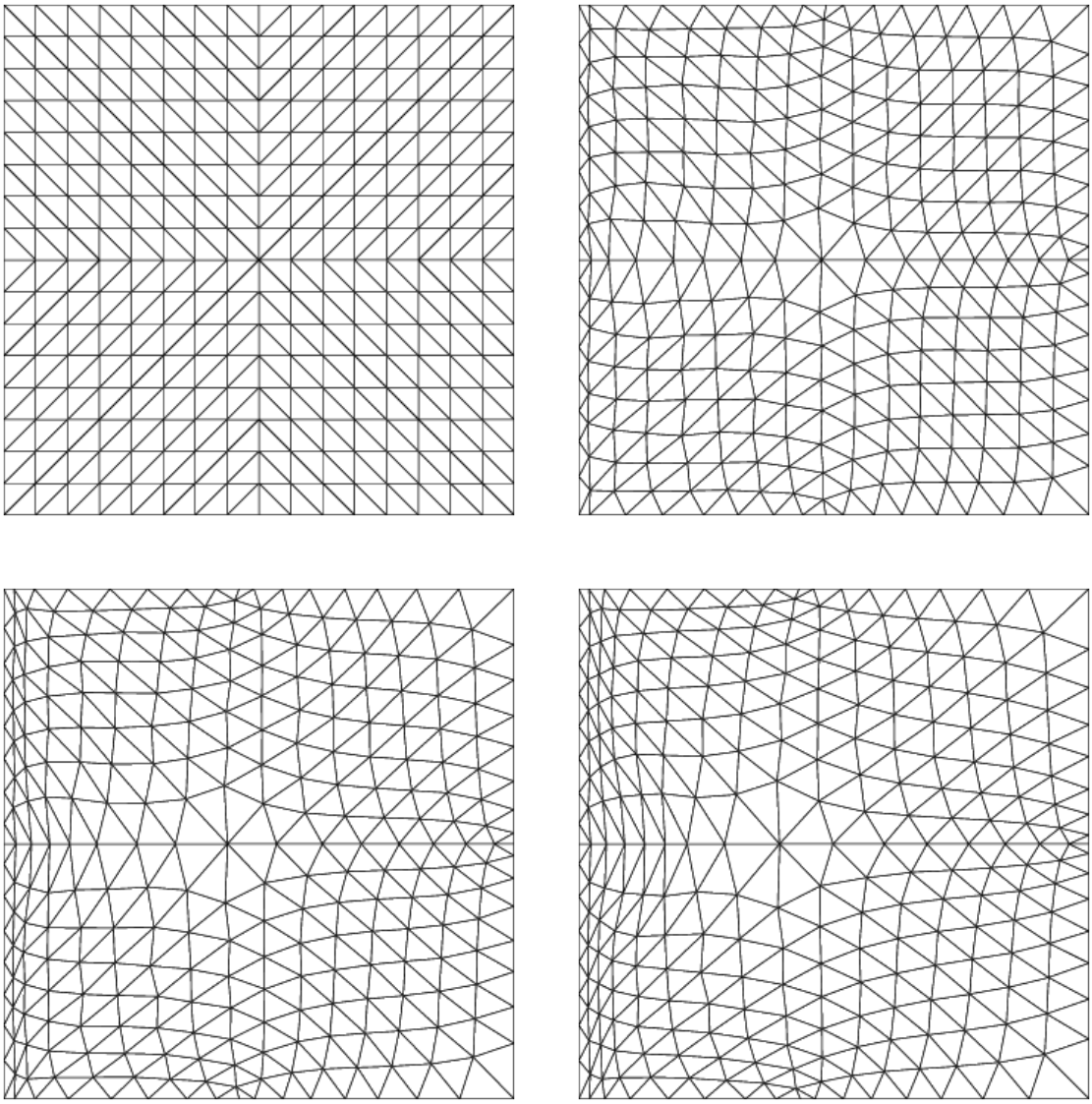

FIG. 4. *The triangulation after 0, 4, 8, 12 iterations.*

can also see from Figure 4 that mesh movement with a fixed topology by itself is not necessarily a good strategy for adapting the mesh. While the meshes are reasonable given the constraints, it is clear that changing the topology of the mesh (e.g., unrefining the mesh near $x = 1$ and refining near $x = 0$) would be beneficial.

In our second example, we employ the smoothing algorithm in conjunction with mesh refinement to illustrate this point. This is the arena where we believe smoothing will be most effective. We consider the function u defined by

$$u(x,y) = \begin{cases} 1 & \text{for } r \leq 1/3 \\ e^{10(1/3-r)} & \text{for } r > 1/3 \end{cases}$$

where $r = \sqrt{x^2 + y^2}$, and Ω is again the unit square. Beginning with a uniform 3×3 mesh made up of eight isosceles right triangles, the mesh was adaptively refined to a mesh with 644 vertices. Except for a small number of "green" triangles on the boundaries of the refined regions, all elements on the refined meshes are isosceles right triangles with edges which are either horizontal, vertical, or have slope ± 1. Our choice of coarse mesh has led to a highly nonuniform mesh of well-shaped elements. The topology and general distribution of the elements is good; the mesh is most refined along the curve $r = 1/3$ as one would expect. The mesh, and a contour map of the piecewise linear interpolant u_L of the function u are shown in Figure 5. A more detailed view of the mesh and contour map near $x = y = \sqrt{2}/6$ is shown in Figure 6.

iteration	$\mathcal{E}$
0	4.45347
4	4.09555
8	3.31294
12	3.14476
16	3.12519
20	3.12679
24	3.13368
28	3.13191
32	3.12501

TABLE 2
Convergence history for the second example.

The main shortcoming of the refinement procedure is that, being based on bisection, the elements are not aligned with the function u, but rather inherit their geometries from the elements in the initial mesh. The contours of u_L are a bit ragged, because the elements are not aligned with the "front" along the curve $r = 1/3$.

In Table 2, we summarize the convergence history of the smoothing algorithm. As with the first example, the most significant changes occur in the first few iterations. In Figures 5 and 6, we illustrate the mesh and the contour map after 12 iterations. Although not perfect, the elements are much better aligned with the function u near $r = 1/3$, and as a result the contours are much smoother.

We remark that in the first example u_x becomes infinite near $x = 0$, while in the second, ∇u is discontinuous at $r = 1/3$. In the first example, the problem is not very serious, since u_{xx} and u_{xy} are not needed along the line $x = 0$, and $u_{yy} = 0$. The problem is more serious in the second example, since assuming constant "second derivatives" is not completely meaningful in elements containing a portion of the arc $r = 1/3$. However, in both cases M_t is always well defined, because it is based on second differences using the endpoints and midpoint of each triangle edge rather than explicitly on second derivatives. It is interesting to note that discontinuities in ∇u present no problem for continuous piecewise linear interpolation in the special case where the elements are exactly aligned with the discontinuity.

6. Single Element Analysis. In practice, we expect that the local problems (10) to have unique solutions, except in the degenerate case where u is linear and $M_t = 0$. However, this seems difficult to prove in general. In a general patch Ω_i, there are too many parameters, the number of elements, the locations of the vertices, and possibly different second derivatives M_t for each element, to make a simple direct analysis. We had hoped to build a proof by restricting attention to a single element

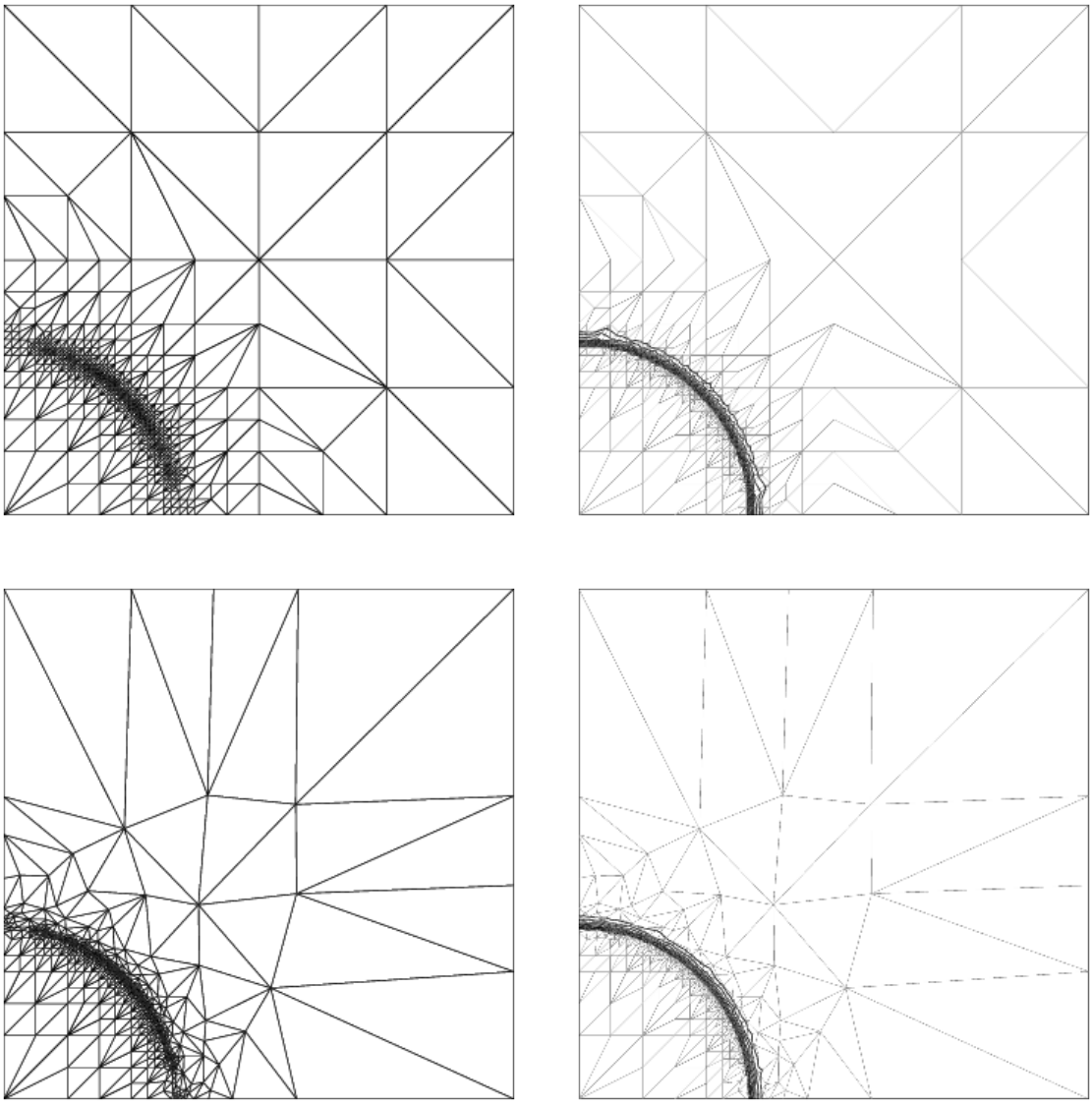

FIG. 5. *Left: the initial triangulation, and triangulation after 12 iterations. Right: a contour map for u_L for the initial triangulation, and for the triangulation after 12 iterations.*

in a patch, so the number of parameters would be more manageable. For example, if one could show that the Hessian matrix for the function $s(t)/q(t)$ was positive definite for each point in Ω_i and for each triangle $t \in \Omega_i$, then uniqueness would follow immediately. Unfortunately, the Hessian for a single element is not always positive definite. Worse yet, in some situations, when $q(t) \to 0$, $s(t)/q(t) \to 0$, in effect negating the impact of the "barrier function". However, even in these unfavorable cases, there is important cancellation when contributions for all elements in Ω_i are summed, so that the Hessian for Ω_i can be uniformly positive definite even when the

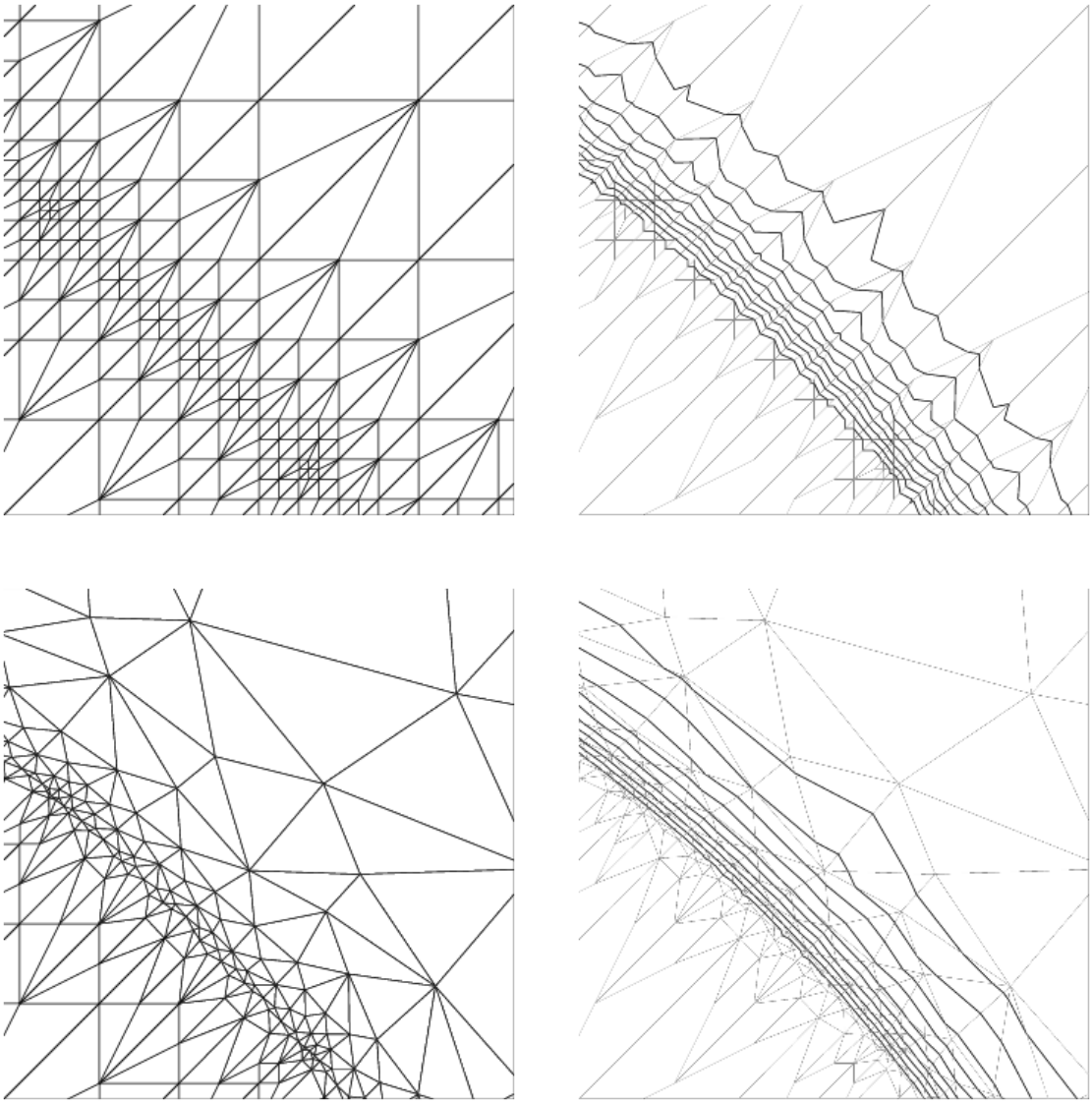

FIG. 6. *A detail from Figure 5.*

contributions from some of the individual elements are not.

In this section, we analyze the case of a single element $t \in \Omega_i$ in detail to illustrate these points. Without loss of generality, we can assume M_t is such that its largest eigenvalue is one and the other eigenvalue λ satisfies $-1 \leq \lambda \leq 1$. When $\lambda > 0$, the level sets for $\ell^t M_t \ell$ are ellipses, while for $\lambda < 0$, the level sets are hyperbolas. We next apply a rotation such that M_t is diagonal, and make a scaling and translation of t such that the vertices are $\nu_1 = (0,0)$ $\nu_2 = (\cos\theta, \sin\theta)$ and $\nu_3 = (x,y)$ with counterclockwise orientation. By symmetry, we may assume $0 \leq \theta \leq \pi/2$.

For this reference element

$$s(t) = (\cos^2\theta + \lambda\sin^2\theta)^2 + (x^2 + \lambda y^2)^2 + ((x-\cos\theta)^2 + \lambda(y-\sin\theta)^2)^2, \tag{19}$$

while

$$q(t) = \frac{2\sqrt{3}(y\cos\theta - x\sin\theta)}{1 + x^2 + y^2 + (x-\cos\theta)^2 + (y-\sin\theta)^2}. \tag{20}$$

To have a catastrophic failure of the barrier function of the type mentioned above, we must have $\cos^2\theta + \lambda\sin^2\theta = 0$ and the terms $x^2 + \lambda y^2$ and $(x-\cos\theta)^2 + \lambda(y-\sin\theta)^2$ both proportional to $y\cos\theta - x\sin\theta$. Solutions can be found for $-1 \le \lambda \le 0$, with $\cos\theta = \sqrt{-\lambda}\sin\theta$. We expect the worst case of this type to be $\lambda = -1$, $\theta = \pi/4$.

To continue our development, it seems best to simplify the model problem again, and we now focus only on this special choice of λ and θ. We let $\nu_1 = (0,0)$, $\nu_2 = (1,0)$, and $\nu_3 = (x,y)$, $0 \le y$, with

$$M_t = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{21}$$

(Note the eigenvalues of M_t are ± 1). This is just the previous problem in a frame of reference rotated by $\pi/4$. For this reference element

$$s(t) = y^2(x^2 + (x-1)^2), \tag{22}$$

while

$$q(t) = \frac{2\sqrt{3}y}{1 + x^2 + (x-1)^2 + 2y^2}. \tag{23}$$

The 2×2 Hessian matrix H_t for $s(t)/q(t)$ is

$$H_t = \frac{1}{\sqrt{3}} \begin{bmatrix} 2y(1 + 2x^2 + 2(x-1)^2 + 2(2x-1)^2 + 2y^2) & (2x-1)(1 + 2x^2 + 2(x-1)^2 + 6y^2) \\ (2x-1)(1 + 2x^2 + 2(x-1)^2 + 6y^2) & 6y(x^2 + (x-1)^2) \end{bmatrix}.$$

It is easy to see that H_t can become indefinite when $y \to 0$, so that H_t is not positive definite for $0 \le y$ and all x.

On the other hand, even in this case it appears that

$$\sum_{t\in\Omega_i} \frac{s(t)}{q(t)}$$

can have a positive definite Hessian, due to cancellations among the contributions from the various elements. For example, we consider the case where Ω_i is a square containing four triangles, with each element similar to this most disadvantageous case. We thus take Ω_3 to be the unit square with ν_i $1 \le i \le 3$ be defined as above, and $\nu_4 = (1,1)$, $\nu_5 = (0,1)$ as shown in Figure 7. M_t will be defined as in (21) for all $t \in \Omega_3$.

The triangle with vertices ν_1, ν_2, ν_3 has $s(t)$ given by (22) and $q(t)$ by (23). The triangle with vertices ν_4, ν_5, ν_3 has

$$s(t) = (1-y)^2(x^2 + (x-1)^2),$$
$$q(t) = \frac{2\sqrt{3}(1-y)}{1 + x^2 + (x-1)^2 + 2(1-y)^2}.$$

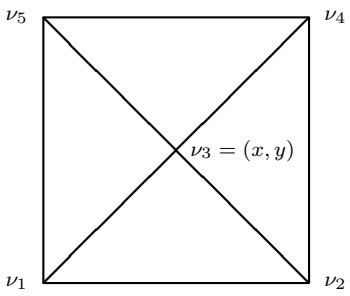

FIG. 7. *The region* Ω_3.

The triangle with vertices ν_5, ν_1, ν_3 has

$$s(t) = x^2(y^2 + (y-1)^2),$$
$$q(t) = \frac{2\sqrt{3}x}{1 + y^2 + (y-1)^2 + 2x^2}.$$

Finally, the triangle with vertices ν_2, ν_4, ν_3 has

$$s(t) = (1-x)^2(y^2 + (y-1)^2),$$
$$q(t) = \frac{2\sqrt{3}(1-x)}{1 + y^2 + (y-1)^2 + 2(1-x)^2}.$$

Note that although none of the single element Hessians is uniformly positive definite, each becomes indefinite in a different part of Ω_3. If we sum contributions from the four elements, we obtain

$$2\sqrt{3}\sum_t \frac{s(t)}{q(t)} = (x^2 + (1-x)^2)^2 + 6(x^2 + (1-x)^2)(y^2 + (1-y)^2) + (y^2 + (1-y)^2)^2.$$

This function has a positive definite Hessian throughout the unit square and a unique minimum at $(1/2, 1/2)$.

Finally, we consider the use of penalty functions as in (11), and show that with the penalty function we can guarantee positive definiteness for the element Hessians. We first show that the Hessian for $s_0(t)/q(t)$ is positive definite for each element t. We can interpret the penalty function $s_0(t)$ as a special case of $s(t)$ when $M_t = I$. For this calculation, it is advantageous to use a reference element with vertices $\nu_1 = (-1/2, 0)$, $\nu_2 = (1/2, 0)$ and $\nu_3 = (x, y)$, with counterclockwise orientation. For this reference element

$$\begin{aligned} s_0(t) &= 1 + \{(x-1/2)^2 + y^2\}^2 + \{(x+1/2)^2 + y^2\}^2 \\ &= 2x^4 + 4x^2y^2 + 2y^4 + 3x^2 + y^2 + 9/8, \end{aligned}$$

while

$$q(t) = \frac{2\sqrt{3}y}{3/2 + 2x^2 + 2y^2}.$$

Thus

$$\frac{s_0(t)}{q(t)} = \frac{64(x^2+y^2)^3 + 16(x^2+y^2)(9x^2+5y^2) + 12(9x^2+5y^2) + 27}{32\sqrt{3}y}.$$

The Hessian for the term $1/y$ is

$$H_0 = \frac{2}{y^3}\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

The Hessian for the term $(9x^2+5y^2)/y$ is

$$H_1 = \frac{18}{y^3}\begin{bmatrix} y^2 & -xy \\ -xy & x^2 \end{bmatrix}.$$

The Hessian for the term $(x^2+y^2)(9x^2+5y^2)/y$ is

$$H_2 = \frac{2}{y^3}\begin{bmatrix} 54x^2y^2+14y^4 & -18x^3y+14xy^3 \\ -18x^3y+14xy^3 & 9x^4+15y^4 \end{bmatrix}.$$

Finally, the Hessian for the term $(x^2+y^2)^3/y$ is

$$H_3 = \frac{2}{y^3}\begin{bmatrix} 15x^4y^2+18x^2y^4+3y^6 & -3x^5y+6x^3y^3+9xy^5 \\ -3x^5y+6x^3y^3+9xy^5 & x^6+9x^2y^4+10y^6 \end{bmatrix}.$$

Each of the Hessians H_i is either positive definite or positive semidefinite for $y > 0$. Thus the Hessian for $s_0(t)/q(t)$,

$$H_t = \frac{64H_3 + 16H_2 + 12H_1 + 27H_0}{32\sqrt{3}},$$

is positive definite, and the Hessian for

$$\frac{s(t) + \rho s_0(t)}{q(t)}$$

can be made positive definite for sufficiently large ρ.

In conclusion, we see that although the analysis of the single element problems is presently incomplete, the foundation seems solid. As for the outer Gauss-Seidel iteration, the situation is more or less completely open. It is certainly clear that the function $\mathcal{E}$ defined in (18) is nonincreasing as a function of the number of Gauss-Seidel sweeps. If we use only a small fixed number of sweeps, this is sufficient to demonstrate that the smoothing algorithm will improve (or at least not impair) the quality of the mesh as measured by $\mathcal{E}$. However, the true nature of the asymptotic behavior of the outer iteration from the theoretical point of view is unknown; we suspect that the best one can hope for is convergence to a (nonunique) local minimum of the function $\mathcal{E}$.

REFERENCES

[1] S. Adjerid and J. Flaherty, *A moving finite element method with error estimation and refinement for one-dimensional time dependent partial differential equations*, SIAM J. Numer. Anal., 23 (1986), pp. 778–796.

[2] ———, *A local refinement finite element method for two-dimensional parabolic systems*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 792–805.
[3] ———, *Second-order finite element approximations and a posteriori error estimation for two-dimension parabolic systems*, Numer. Math., 53 (1988), pp. 183–198.
[4] D. Arney and J. Flaherty, *An adaptive mesh-moving and local refinement method for time-dependent partial differential equations*, ACM Trans. on Math. Software, 16 (1990), pp. 48–71.
[5] I. Babuška, J. Chandra, and J. Flaherty, *Adaptive Computational Methods for Partial Differential Equations*, SIAM, Philadelphia, 1983.
[6] I. Babuška, O. C. Zienkiewicz, J. R. Gago, and E. R. A. e Oliveira, *Accuracy estimates and Adaptive refinements in Finite Element Computations*, John Wiley, London, 1986.
[7] R. E. Bank, *Analysis of a local a posteriori error estimator for elliptic equations*, in Accuracy Estimates and Adaptivity in Finite Element Computations, (eds. I. Babuška, O. C. Zienkiewicz, and E. Arantes e. Oliveira), J. Wiley and Sons, New York, 1986, pp. 119–128.
[8] ———, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations, Users' Guide 7.0*, Frontiers in Applied Mathematics, Vol. 15, SIAM, Philadelphia, 1994.
[9] R. E. Bank and R. K. Smith, *A posteriori error estimates based on hierarchical bases*, SIAM J. Numerical Analysis, 30 (1993), pp. 921–935.
[10] R. E. Bank and A. Weiser, *Some a posteriori error estimates for elliptic partial differential equations*, Mathematics of Computation, 44 (1985), pp. 283–301.
[11] M. Bern and D. Eppstein, *Mesh generation and optimal triangulation*, tech. report, Xerox Palo Alto Research Center, Palo Alto California, 1991.
[12] E. F. D'Azevedo and R. B. Simpson, *Optimal triangular meshes for minimizing the gradient error*, tech. report, Department of Computer Science, University of Waterloo, Ontario, Canada, 1991.
[13] A. R. Diaz, N. Kikuchi, and J. E. Taylor, *A method of grid optimization for finite element methods*, Comp. Meth. Appl. Mech. Eng., 41 (1983), pp. 29–45.
[14] N. Dyn, D. Levin, and S. Rippa, *Data dependent triangulations for piecewise linear interpolation*, IMA J. Numer. Anal., 10 (1990), pp. 137–154.
[15] D. A. Field, *Laplacian smoothing of Delaunay triangulations*, Comm. in Applied Numer. Anal., 4 (1988), pp. 709–712.
[16] W. H. Frey and D. A. Field, *Mesh relaxation: A new technique for improving triangulations*, Int. J. Numer. Meth. Eng., 31 (1991), pp. 1121–1133.
[17] J. M. Hyman, *Moving mesh methods for partial differential equations*, in Mathematics Applied to Science, Academic Press, Boston, 1986.
[18] K. Miller, *Moving finite elements. II*, SIAM J. Numer. Anal., 18 (1981), pp. 1033–1057.
[19] K. Miller, *Alternate modes to control the nodes im the moving finite element method*, in Adaptive Computational Methods for Partial Differential Equations, SIAM, Philadelphia, PA, 1983.
[20] K. Miller and R. N. Miller, *Moving finite elements. I*, SIAM J. Numer. Anal., 18 (1981), pp. 1019–1032.
[21] W. F. Mitchell, *A comparison of adaptive refinement techniques for elliptic problems*, ACM Trans. Math. Soft., 15 (1989), pp. 326–347.
[22] M. Schweingruber and E. Rank, *Adaptive mesh generation for triangular or quadrilateral elements*, in Proceedings of the First European Conference on Numerical Methods, Brussels, Belgium, 1992.
[23] M. S. Shephard, *Approaches to the automatic generation of finite element meshes*, Appl. Mech. Rev., 41 (1988), pp. 169–185.
[24] R. B. Simpson, *A survey of two dimensional finite element mesh generation*, in Proceeding of the Ninth Manitoba Conference on Numerical Math. and Comput., 1979, pp. 49–124.
[25] T. Strouboulis and J. T. Oden, *A posteriori error estimation of finite element approximations in fluid mechanics*, Comp. Meth. Appl. Mech. Eng., 78 (1990), pp. 201–242.
[26] R. Verfürth, *A review of a posteriori estimation and adaptive mesh refinement techniques*, tech. report, Institut für Angewandte Mathematik der Universität Zürich, 1993.