Adaptive Triangular Mesh Generation

Gordon Erlebacher*
NASA Langley Research Center, Hampton, Virginia
and
Peter R. Eiseman†
Columbia University, New York, New York

A general adaptive algorithm is developed on triangular meshes. The adaptivity is provided by a combination of node addition, dynamic node connectivity and a simple node movement strategy. While the local restructuring process and the node addition mechanism take place in the physical plane, the nodes are displaced on a monitor surface, constructed from the salient features of the physical problem. An approximation to mean curvature detects changes in the direction of the monitor surface, and provides the pulling force on the nodes. Solutions to the axisymmetric Grad-Shafranov equation demonstrate the capturing, by triangles, of the plasma-vacuum interface in a free boundary equilibrium configuration.

Nomenclature

A^B	= area of triangle labeled B
$A^{i+½}$	= area of triangle $i,i+1,c$
A_{cr}	= critical triangle area
$A_{\min}$	= minimum allowable triangle area
$A_{\max}$	= maximum allowable triangle area
c	= cell center
J_T	= toroidal current density
K	= approximate mean curvature
$\bar{r}$	= radius vector (x,y)
$r_{i+½}$	$= r_{i+1} - r_i$
r^i	$= r_{i+½} - r_{i-½}$
s	= arc length
S	= triangle area
t	= unit tangent vector
$u(x,z)$	= monitor surface function
u	= solution function
$w(x,y)$	= weight density function
x,y,z	= Cartesian coordinate directions
α	= parameter in range [0,1]
α_c	= curvature pull
α_i	= normalization coefficient of the ith feature
ϵ	= small parameter
$\phi(x,y)$	= arbitrary function
λ	= parameter in range [0,1]
ψ	= poloidal flux function
ξ	= computational coordinate

Subscripts

c	= cell quantity
i	= node i
E,F	= diagonals of quadrilaterals found by two adjacent triangles
I	= intersection point of diagonals E and F
x,xx	= first and second x-derivatives

Presented as Paper 84-1607 at the AIAA Seventh Fluid Dynamics, Plasma Dynamics and Lasers Conference, Snowmass, CO, June 25-27, 1984; received July 24, 1984; revision received Sept. 12, 1986. Copyright © American Institute of Aeronautics and Astronautics, Inc., 1987. All rights reserved.

*Computational Methods Branch.

†Department of Applied Physics and Nuclear Engineering. Member AIAA.

Introduction

WHEN physical phenomena exhibit features which vary rapidly over a wide range of spatial and temporal scales, the corresponding computational simulation is done best in a solution adaptive sense. In this study we concentrate our attention on an adaptive mesh strategy.

To solve the increasingly complicated problems of current interest (complex shock structures, reacting flows), the physical domain is usually subdivided into a discrete set of nodes with a chosen connectivity (triangles, quadrilaterals), which might or might not have any associated structure. Conventional curvilinear grids are one example of structured quadrilaterals. With any structure, the continuous problem is approximated in some sense on the discrete set of nodes. The tradeoff between computational speed and number of nodes usually leads to the search for higher-order numerical schemes that maximize the coarseness of the mesh cells, without sacrificing the accuracy of the computation. With any scheme, the same objective can be achieved by dynamically adjusting the mesh so that ample resolution is supplied for high gradients and curvatures in the solution. At other locations, larger cells may be taken without sacrificing accuracy provided that the gradation between large and small cells is sufficiently smooth and that the remaining more mildly varying parts of the solution are adequately represented.

Many adaptive mesh techniques have been proposed for moving curvilinear grids.[1-7] Each is aimed at the reduction of computational inaccuracies in regions with small-length scales. Embedded mesh algorithms which give local refinement without displacing the nodes, have also been examined.[8] In each of the previous methods, no real provision has been made for situations where the global pattern of small-length scales changes in a dynamical manner. It is this situation which motivates this study. A key element is the use of a general triangular mesh that can alter the connectivity between points to conform to dynamically changing configurations. This alteration is called restructuring.

As a first step toward this objective, an adaptive strategy on a triangular mesh is considered which combines local restructuring, node addition and mesh movement. No attempt is made to implement sophisticated numerical solution algorithms for specific problems. Rather, the goal is to find an adaptive mesh algorithm which is as much as possible problem-independent, and which focuses on maintaining a good structure devoid of excessively skewed triangles. It is observed that triangle skewness should be avoided, not on the physical plane, but rather on a "monitor surface," constructed from the features to be resolved.

Local restructuring allows the common edge between two adjacent triangles to be replaced by the second diagonal of the resulting quadrilateral.[9] If all the edges are periodically checked, and "flipped," according to some criterion that reduces the skewness of the triangles, the mesh retains a good structure. The numerical algorithms implemented at the present time only require the computation of the physical variables at triangle nodes. This eliminates the need for interpolation procedures when the edges are flipped.

In conjunction with a local restructuring process, additional nodes are inserted at triangle centroids when there are too few mesh points to resolve the salient features. On the other hand, within the context of curvilinear grids, inserting nodes entails the addition of a whole coordinate line which wastes grid points unnecessarily. This problem has been addressed by the use of partial coordinate curves.[10] However, the concurrent problem of treating the dangling endpoints then arises.

To complement the restructuring process, the mesh points are allowed to cluster about the salient features of the fluid. For applications with arbitrary mesh topology, a simple, explicit movement algorithm, with little logic, has been developed in a manner which prevents adjacent convex cells from overlapping. While not guaranteed, overlap of concave cells is avoided most of the time. A combination of restructuring and movement is necessary to maintain a good mesh structure with an overwhelming majority of convex cells. An alternative approach is a modification of the movement mechanism to directly ensure nonoverlap.

Basic pointwise movement stems from the establishment of a positive weight on each triangle. The weight is chosen to properly reflect the features with respect to which the nodes should cluster. These in turn can be interpreted as mass densities over the triangle area, so that each triangle acquires a positive mass. The position of the new mesh point then becomes the center of gravity of the polygon formed by that node and the triangles adjacent to it.

One of the main goals in the application of adaptive methods is to decrease the number of arbitrary parameters which must be specified by the user on the basis of intuition, and replace them with parameters which have a physical interpretation. For two-dimensional problems, the points are pulled along a general surface according to an approximation of its mean curvature. The general surface is used to monitor salient solution features. Its direct use eliminates the need to specify a gradient in the weight function since that information is implicitly contained in the triangle area on the monitor surface.

After the introduction of some basic notation required to efficiently describe triangular meshes and operations thereon, successively discussed are the associated data structure, the use of monitor surfaces, local restructuring, node addition, and finally the node movement algorithm. As a test case, the free boundary axisymmetric plasma equilibria in a tokamak configuration is computed. The free boundary is treated from a viewpoint that is directly analogous to shock capturing, as opposed to shock fitting. The monitor surface is carefully chosen to ensure that the mesh points cluster in regions of rapid changes in toroidal current density and in regions of low poloidal magnetic fields. The variations of current density define the main front, whereas the magnetic field represents a second physical effect.

Triangular Meshes

Among the commonly used triangular meshes, a distinction must be made between structured and unstructured meshes. A structured triangular mesh can be transformed into a uniform mesh of nearly equilateral triangles by an invertable coordinate transformation. Therein, each node lies at the intersection of three coordinate lines. While this regularity simplifies the treatment of rather complicated geometries (as demonstrated by Winslow[11]) and leads to simple movement algorithms, a local restructuring of such a mesh destroys the simple connectivity pattern by locally changing the number of edges connected to each node. For this reason, the physical domain is triangulated with no restrictions on the node connectivity, resulting in an unstructured triangular mesh. The main disadvantage is the more complex data structure and logic that is required to keep track of the various nodes, edges, and triangles. Once the data structure is established however, it does not change and is mostly transparent to the user.

For a general mesh format, a number of conventions and definitions are particularly advantageous and are used throughout this study. The nodes on an arbitrary triangle are labeled in a counterclockwise direction. Node numbers are indicated by subscripts, typically chosen from the set (i,j,k). A cell at a node c is formed from the triangles which contain it as a vertex. The cell boundary is defined by successively joining triangular centroids with midpoints of edges emanating from c until a closed loop is formed, or a boundary is encountered. This is illustrated in Fig. 1. A subscript c will either refer to such a cell or to the cell center. With the above definitions, the cell area is readily given by

$$A_c = \sum_{B(c)} A^B \tag{1}$$

where the summation index B includes all triangles that have the point c as a vertex. Triangle edges are defined by their endpoint indices. For example, the vector between nodes i and $i+1$ is expressed as $r_{i,i+1}$. In the case of successive indices, the shorter notation $r_{i+½}$ is simpler and is often preferred. A similar alternative notation for triangle areas is $A^{i+½}$, which is interpreted as the area of triangle c, i, $i+1$ in cell c. As a general rule, a superscript $i+½$ translates into "evaluated at the centroid of triangle c, i, $i+1$" where c is the node index whose cell is under discussion.

Discrete formulas for the gradient and Poisson operators, needed in the practical implementation of the adaptive scheme described later are easily obtained from appropriate discretizations of integral relations. Details are presented elsewhere.[12]

It has been shown[12] that maximum bounds on the truncation error from gradient and Laplacian operators are inversely proportional to the triangle area. Nonetheless, the truncation errors could still be orders of magnitude smaller than these bounds, even on very skewed meshes. It is the authors' contention that the triangles must be well structured on the solution surface itself rather than in the physical

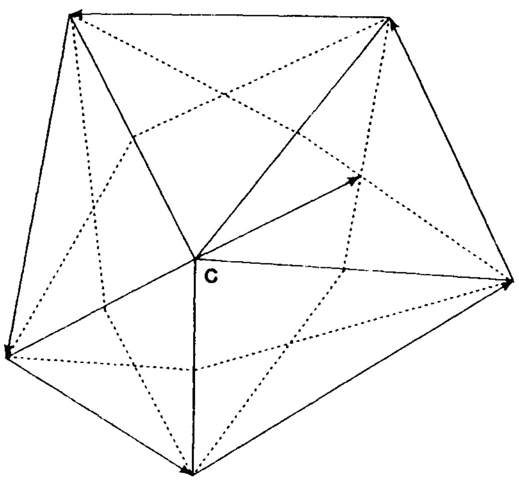

Fig. 1 Cell associated with node c (dotted region).

plane, although no formal proof of this has yet been given. There is, however, the intuitive notion that discrete formulas of a general nature are not as accurate as formulas that adjust to the object on which they are applied.

Data Stuctures

The algorithms developed for an unstructured triangular mesh depend heavily upon the chosen data structure. The advantages gained from the use of a triangular mesh come at the additional expense of an increased complexity in the code logic that is necessary to track the various triangles, edges and nodes. A library of high-level subroutine modules serves as an interface between the user and the actual low-level code. This library is created only once, and is independent of the numerical schemes adopted to solve the continuous problem. Examples include node addition routines, diagonal flipping routines, and various types of array updates. The main arrays used for the mesh data (excluding temporary storage and space allocation for solution variables), are defined as follows:

$X(I)$, $Y(I)$, $I=1$, NVT:	x,y coordinates of node I
$NGH(I,J)$, $I=1,10$; $J=1$, NVT:	Jth neighbor of node I
$NED(I,J)$, $I=1,2$; $J=1$, $NEDT$:	the node indices of the two extremities of edge J
$NTRI(I,J)$, $I=1,3$; $J=1$, NTR:	Ith node index of triangle J
$NTRIE(I,J)$, $I=1,3$; $J=1$, NTR:	Ith edge index of triangle J
$NTRIED(I,J)$, $I=1,2$; $J=1$, $NEDT$:	Ith triangle common to edge J
$KH(I)$, $I=1$, NVT:	number of neighbors of node I

NVT, *NEDT*, and *NTR* are, respectively, the total number of vertices, edges, and triangles on the physical domain. In this list, an implicit numbering of nodes, edges, and triangles is assumed. Consecutive boundary nodes and edges, and a node's neighboring indices are numbered counterclockwise.

An estimate of the required memory is possible in the limit of an infinite number of triangles. The number of edges is then three times, and the number of triangles twice the number of nodes. If we allow a maximum of 10 neighbors per node, the total amount of storage required is 37 *NVT* memory words, which is quite substantial compared to the 2 *NVT* words needed for curvilinear coordinates.

In view of the evolutionary trend in computer hardware development, a choice between striving for low-memory or high-speed would favor the latter. The decreasing cost per memory word shows no sign of abatement and moreover the advent of parallel processing computers promises to multiply execution speed by orders of magnitude. What then are the important factors to consider? As the number of mesh points decreases, so does the spectral radius of typical iterative methods, which reduces the total number of work units necessary to obtain a converged solution. This reduction in work, and consequently the faster execution times, comes both from a lower number of iterations, and from a lower amount of work per iteration, even though the latter is partially offset by the work associated with the adaptive scheme. However, the total CPU time associated with the node movement is only a function of the mesh geometry, the total number of mesh points, and the structure of the monitor surface. It does not depend on the original number of differential equations to be solved. Therefore, as the systems to solve increase in complexity, the fraction of CPU time allotted to the adaptive portion of the computation becomes less and less significant.

As a final note, no attempt has been made at this stage to optimize the movement algorithms with respect to computer architecture. This would certainly vary with each computer. In our context, the storage required for the bookkeeping consists mostly of positive integers less than 10,000 that can be stored in two-byte words thereby substantially reducing storage requirements. Moreover, a few mainframe computers have very fast gather and scatter routines that permit very efficient vectorization algorithms on unstructured meshes.

In view of such considerations, it is not the objective here to present sophisticated numerical schemes, but rather to concentrate on practical means of detecting and resolving flow features while maintaining a good mesh structure. The goal is feasibility rather than peak efficiency. In this spirit, most numerical schemes are made more useful.

Monitor Surfaces

The dynamic behavior of the solution is usually driven by a small number of critical variables. For example, the evaluation of safety factors and current profiles in plasma equilibria dictate that the poloidal current variations and the location of the magnetic axis be accurately determined. These critical features are loosely defined as quantities that, when properly resolved, generate numerical solutions of adequate accuracy. Another example is given by shock waves where pointwise locations are determined by accurately resolving variations in pressure. In many cases, the controlling feature is not the solution vector, but some suitable combination of its components. These form a vector in their own right which can be evaluated at each point in physical space. Collectively, all evaluations form an abstract surface called a monitor surface.[13] Herein, only two-dimensional problems are considered. The feature vector is a scalar function of the two independent variables. As the physical solution evolves in time, so does the monitor surface. Ideally, the mesh points are uniformly distributed on the monitor surface with local concentrations in regions of high curvature. With the monitor surface, the essential adaptive data have been extracted from the solution and have been expressed in this single purely geometric object typically considerably simpler than the solution itself. This surface contains all of the pertinent input information, quite unlike the formulation of Aniywo[14] which is similar in spirit but which is dispersed into many distinct objects.

Once a strategy is agreed upon to resolve the monitor surface, the solution vector must still be interpolated onto the new grid, in a manner consistent with the numerical scheme used to advance the solution to its next step. This last issue is separate from the adaptive strategy and is not addressed here. A number of issues immediately come to mind, and are partially addressed as follows:

1) How are the various features combined to form the monitor surface?

2) Should the surface be considered as a coordinate-free geometric entity, or as a function that spans the physical domain?

3) Armed with an answer to question 2, how are the grid points distributed across the monitor surface? Or stated differently, what are the appropriate controls that will ensure a satisfactory mesh structure that adequately dissects the surface?

These questions are addressed in the following sections.

Local Restructuring

To take full advantage of the general connectivity of the triangular mesh, and to avoid excessively distorted cells or

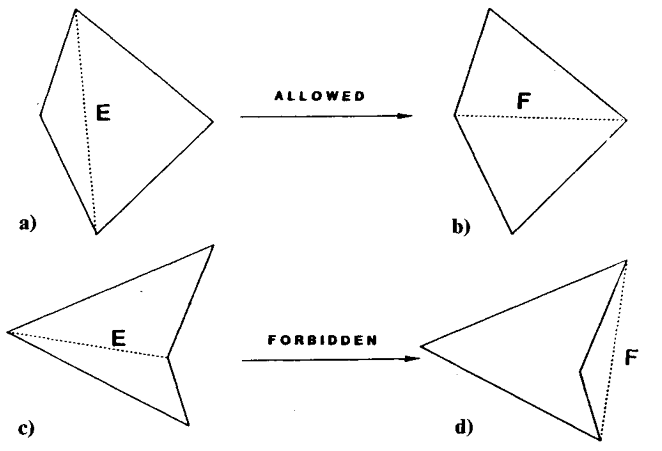

Fig. 2 Diagonal flipping technique: a), b) edge E is replaced by F; b) locally improved structure; c), d) triangle overlap, d) caused by switching when the parameter $(\lambda_E)_I$ is not in the unit interval.

triangles, the diagonal flipping technique first employed by Crowley[15] and illustrated in Fig. 2 is adopted. By replacement of the common edge E with the opposite diagonal F, two new triangles appear with F as the common edge and with an improvement in structure over the ones initially displayed. Various schemes have been proposed to determine the most appropriate conditions under which to flip the edge, the simplest being to take the shortest diagonal, which is the approach adopted. On the other hand, Fritts[9] takes the diagonal that minimizes the sum of the cotangents of its subtended angles. He has shown that this guarantees a diagonally dominant coefficient matrix for his discrete Laplace operator. The algorithm used here is as follows:

1) Express the diagonals E and F as parametric line segments with the respective parameters λ_E and λ_F linearly varying from 0 to 1 in going between the diagonal extremities.

2) Calculate the intersection, I, of the two diagonals. At this point, the parameters take on the values $(\lambda_E)_I$ and $(\lambda_F)_I$.

3) If $0.2 \leq (\lambda_E)_I \leq 0.8$ and $(\lambda_F)_I \leq 0.2$ or $(\lambda_F)_I \geq 0.8$, flip the diagonal.

4) When both $(\lambda_E)_I$ and $(\lambda_F)_I$ lie within the range 0.2, 0.8, pick the shortest diagonal.

If the intersection results in either parameter $(\lambda_E)_I$ or $(\lambda_F)_I$ being outside of the unit interval, then a diagonal falls outside of the quadrilateral. When $(\lambda_E)_I$ is out of range, the misplaced diagonal is F, and similarly for $(\lambda_F)_I$ and E. An illustration is given in Fig. 2. Since the misplacement is smoothly approached in a linear fashion, an excessively thin triangle would result near the endpoints. To maintain good structure by biasing against such thin triangles, preference is given to the midrange which we take as the interval from 0.2 to 0.8. Upon starting with E in place, we know that the corresponding $(\lambda_F)_I$ is in the unit interval but is possibly outside of the midrange. If it misses the midrange while the competing $(\lambda_E)_I$ lands there, then an improvement results with a diagonal flip from E to F. This is just step 3. If both $(\lambda_E)_I$ and $(\lambda_F)_I$ fall in the midrange, then each diagonal avoids the excessively thin triangle problem and the best structure results from the choice of the shortest diagonal. This is step 4.

Each interior edge is systematically checked for the need to flip. To reduce the influence of a particular ordering in the edge index storage, successive sweeps are performed until no more flips occur. However, the cost of more than a single sweep through the mesh is wasteful, given that only in those regions where the movement is substantial, or where nodes are added, does the structure deteriorate. A limitation to one overall edge sweep is not overly restrictive unless these

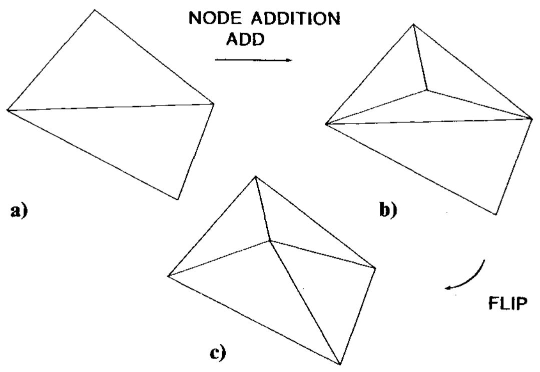

Fig. 3 Node addition: a), b) node is inserted at the centroid of A triangle that requires refinement; c) mesh structure is recovered by local restructuring.

regions cover a large fraction of the computational domain. At present, the edges are scanned once after every application of the movement algorithm.

Node Addition

As nodes cluster around regions of high information content, the intermediate regions might become depleted. Node addition, followed by local restructuring, reduces local truncation errors, as well as the number of skewed triangles that result because of the mesh topology. Node enrichment is useful in shock structures, in boundary layers, and in strategic locations such as sonic points. Although node insertion is important and has many uses, its only application in the code so far is in the initialization process where the mesh must adapt to an initial solution profile.[12]

Criteria to decide whether a triangle needs refinement are always converted into area comparisons. The refinement takes place with the addition of an extra node at the triangle centroid, which is then joined to the original three triangle vertices. This generates two new triangles and three new edges. After all the additional nodes have been inserted, a sweep through the edges is executed for possible diagonal flips (Fig. 3). An absence of this second scan quickly results in extremely skewed cells.

Clearly, a wide variety of schemes exist to assess whether or not a node should be inserted. The simplest criterion is to stipulate that each triangle area vary inversely with some positive weight function. Herein, the weight is the magnitude of the gradient of the initial solution guess. Other possible weight functions include monotone functions of triangle skewness, measured as a departure from equilateral triangles, or functions that increase sharply when critical parameters vary rapidly over a short distance.

An ideal numerical mesh should have a triangle area that increases as the weight evaluated at its centroid decreases, and vice versa. However, the finite amount of available computer storage dictates that the area should not be allowed to decrease below a prescribed minimum ($A_{\min}$), while accuracy considerations restrict the area to lie below some maximum value $A_{\max}$, usually chosen as the ratio of the total area of the physical domain over the strict minimum number of triangles necessary to cover the domain if the solution were smooth. The critical triangle area is chosen to vary linearly between $A_{\min}$ and $A_{\max}$ according to

$$A_{cr} = \alpha A_{\min} + (1-\alpha) A_{\max} \tag{3}$$

where α is the weight function, normalized to unity and calculated at the triangle centroid. A node is inserted if the triangle area is greater than A_{cr}. This of course leads to triangles with areas less than $A_{\min}$ because of the fourfold reduction in area after the node has been inserted.

To ensure that the variation of adjacent triangle areas is not too severe in the rapidly varying regions of the monitor surface, the weights are shifted toward higher values. This induces the addition of nodes in triangles whose actual weight is below the critical threshold. With an assumed smooth transition, these extra nodes become part of a buffer region that surrounds the small length scale regions. Consequently the critical local disturbances are prevented from leaving the resolved domain before the next mesh adjustment.

Grid Movement

Many of the node movement concepts implemented on triangular meshes are direct extensions of their one-dimensional counterparts. Grid movement concepts and algorithms from the one-dimensional viewpoint are examined first. This is followed by the two-dimensional generalization where the main differences between the two formulations are pointed out.

As stated earlier, the objective pursued is to properly relocate a discrete set of nodes on a monitor surface $u(x)$. Confusion of $u(x)$ with the solution of the original set of PDEs is avoided since the original problem is now decoupled into two independent components: 1) the numerical algorithm that solves the PDE, and 2) the resolution of the monitor surface, which in one-dimension becomes a monitor curve. In reality, these two components are coupled through the interpolation of the solution vector back onto the new mesh, which leads to an even wider range of possible algorithms.[12,16]

Local geometric properties of curves and surfaces are easily derived from a Taylor expansion of the vector $r=(x,u)$ with respect to some independent variable, for which the two obvious choices are the abscissa x and the arc length s. Choosing x as an expansion parameter isolates a particular coordinate direction and does not take full advantage of the curve's intrinsic properties. Keeping terms up to second-order in Δx_j, $u_{j+1}=u(x_{j+1})$ is, approximately,

$$u_{j+1}=u_j+(u_x)_j\Delta x_j+\tfrac{1}{2}(u_{xx})_j\Delta x_j^2+\mathcal{O}(\Delta x^3) \tag{4}$$

where $\Delta x_j - x_{j+1}-x_j$. The coefficients of Δx_j and Δx_j^2 are respectively proportional to the first and second derivatives of u. These are often used as weights to attract nodes toward regions where u varies rapidly. As u_x and u_{xx} increase, it

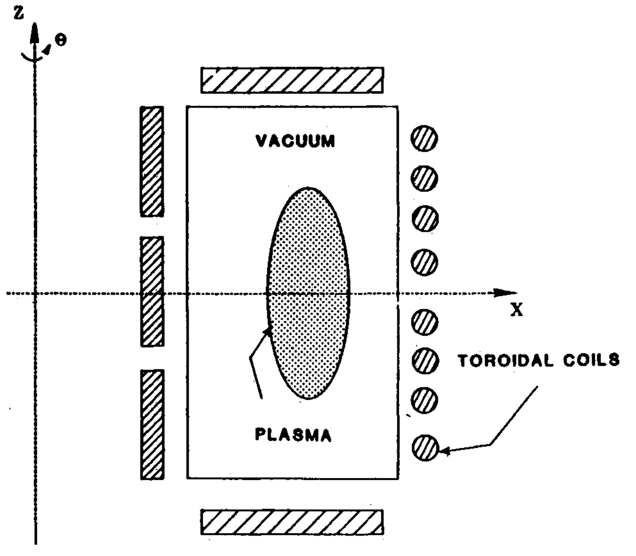

Fig. 4 Cross section of an axisymmetric toroidal device. The toroidal coils are placed symmetrically with respect to the midplane $z=0$.

seems natural to decrease the node interspacing so that the remainder of the Taylor series remains below a specified error tolerance. The generalization of Eq. (4) to two-dimensions is straightforward: replace u_x by the gradient of u, and u_{xx} by the coefficients u_{xx}, u_{xy}, u_{yy}. To simplify matters somewhat, one can combine the last three coefficients into $\nabla^2 u$.

However, u_{xx} is not always appropriate as a measure of attraction. A more precise measure is given by the actual curvature, which is the change in direction of the unit tangent vector with respect to arc length. The curvature K is defined by

$$K=\frac{\partial t}{\partial s}, \qquad K=\frac{u_{xx}}{(1+u_x^2)^{3/2}} \tag{5}$$

where $t=(\partial r/\partial s)$ is the unit tangent vector along $u(x)$, and s is the cumulative arc length. It is also interesting to note that the curvature appears as the coefficient of the second derivative in a Taylor expansion of r with respect to arc length. To emphasize the importance of curvature for proper feature capturing, consider the function $u(x)=x^2$. First derivative clustering would concentrate the nodes in the regions furthest away from the origin. The second derivative is constant, so it has no effect. Nonetheless, it seems intuitively obvious that the origin of u should attract more nodes than the outer regions. This is the case if second derivative pull is replaced by curvature pull, since

$$K=2/(1+4x^2)^{3/2}$$

It is maximum at the origin and decreases toward zero as x tends to infinity. Second derivatives and curvature are approximately equal only in regions where $u_x^2 \ll 1$.

Movement Algorithm

To allow a natural extension to two-dimensional triangular configurations, the movement algorithm must satisfy the following conditions:

1) The maximum displacement of any node is constrained to the region formed by its nearest connecting neighbors.

2) The weight function (defined below) must only depend on properties of the monitor curve (or surface) within the area defined in condition 1.

3) In the absence of curvature, the nodes must be uniformly distributed on the monitor surface.

Condition 2 is included for simplicity, because it is difficult to increase the domain of dependency of a node on an unstructured triangular mesh beyond its nearest neighbors. Condition 3 immediately suggests the mapping

$$\mathrm{d}\xi=[1+w(r)]\,\mathrm{d}s \tag{6}$$

where $w(r)$ is a positive weight function. This formula is obtained from the observation that if $\mathrm{d}\xi$ is kept fixed, the spacing of nodes with respect to arc length s decreases with increasing weight, and is constant in the limit of vanishing weight. The arc length is related to the x coordinate by

$$\mathrm{d}s^2=(1+u_x^2)\,\mathrm{d}x^2 \tag{7}$$

which automatically includes gradient effects without the need to introduce an extra free parameter to control its pulling strength. If ds is replaced by dx in Eq. (6), and if $w(r)$ is a linear combination of the two derivative magnitudes, Dwyer's formulation is obtained.[2] Motivated by the discussion on curvature, the weight function is chosen to be proportional to the curvature magnitude as performed by Ablow and Schecter[17] in one dimension, and Eiseman[3] in higher dimensions. Clearly, conditions 1 and 2 place restrictions on the discretization of Eq. (6). After dividing both sides of Eq.

(6) by dξ and differentiating them with respect to ξ, the discretization of the resulting equation is given by

$$0=(1+\alpha K_{j+½})(s_{j+1}-s_j)-(1+\alpha K_{j-½})(s_j-s_{j-1}) \tag{8}$$

where the subscript $j+½$ denotes the midpoint between j and $j+1$ and where α is an adjustable parameter. Extracting s_j from Eq. (8) leads to

$$s_j^{\text{new}}=\left[\frac{s_{j+1}(1+\alpha K_{j+½})+s_{j-1}(1+\alpha K_{j-½})}{(1+\alpha K_{j+½})+(1+\alpha K_{j-½})}\right]^{\text{old}} \tag{9}$$

where the superscripts new and old emphasize values before and after displacement, the curvatures being held constant. Equation (8) is a tridiagonal system of equations that is easily solved for the s_j. New values of s_j are interpolated onto the old grid, and new values of (x_j,u_j) are derived from knowledge of cumulative arc length s_j along the monitor curve prior to the node displacements. Equation (9) clearly satisfies conditions 1 and 2. Also, when $\alpha=0$,

$$s_j^{\text{new}}=½(s_{j+1}+s_{j-1})^{\text{old}}$$

and the solution to Eq. (9) is a uniform arc length distribution.

Two-Dimensional Movement

Two of the formulas derived in the one-dimensional context cannot be easily extended to two dimensions. These are 1) the curvature and 2) the geometrical interpretation of the movement algorithm.

The problem of resolving a two-dimensional monitor surface with a finite number of mesh points, in a coordinate-free environment is complicated by the fact that there are very few intrinsic properties of that surface akin to curvature. The best known among these is the Gaussian curvature, which is simply the product of the two principle curvatures.[3] Its expression in terms of the Riemanian metric is rather expensive to compute on a triangular mesh. On the other hand, the average of the principle curvatures, called the mean curvature, can be approximated to yield a simple formula, which is a natural two-dimensional generalization of the curvature of one-dimensional functions

$$K=\frac{|\nabla^2 u|}{2[1+(\nabla u)^2]^{3/2}} \tag{10}$$

When gradients are small relative to unity, K is well approximated by the Laplacian of u. Otherwise it is always scaled down. The basic property of Eq. (10) is that K approximates the mean curvature. By contrast, the Gaussian curvature is inappropriate because it vanishes along steep fronts if their properties vary in only one primary direction. For convenience, the term curvature, when applied to a two-dimensional triangulated surface, will refer to the approximate formula of Eq. (10).

The second point that must be addressed is the extension of Eq. (9) to a two-dimensional formulation where arc length becomes the area of the triangles on the monitor surface. However, an update of triangle areas is not easily translated into an update of node positions. Equation (9) is therefore modified, and replaced by

$$r_c^{\text{new}}-r_c^{\text{old}}=\left[\frac{\Sigma_i S^{i+½}(1+\alpha K^{i+½})r^{i+½}}{\Sigma_i S^{i+½}(1+\alpha K^{i+½})}\right]^{\text{old}} \tag{11}$$

where $r_j=(x_j,y_j)$ is the radius vector of node j, $S^{i+½}$ is the area of triangle c, i, $i+1$ on the monitor surface, and $r^{i+½}$ is the vector that joins node c to the centroid of triangle c, i, $i+1$. Equation (11) can be physically interpreted on the surface as moving node c to the centroid of its surrounding polygon, composed of triangles of area $S^{i+½}$, and of mass density $(1+\alpha K^{i+½})$. When the curvature vanishes, nodes converge to the geometrical center of gravity of the polygons that enclose them. This is the definition adopted for mesh uniformity, which is now a function of the mesh convectivity. To support this idea of uniformity, a uniform mesh of equilateral triangles, after distortion, drift back to its original state, if K is set to zero. Gradient effects are implicitly included in $S^{i+½}$ through the relation

$$S^{i+½}=A^{i+½}[1+(\nabla u)^2]^{½}$$

Upon downward projection, this produces gradient clustering in the physical region. By its very construction, Eq. (11) guarantees that the nodes remain within their cells in physical space when their nearest neighbors form convex, or not too severely concave, polygons. After each movement sweep over the nodes, the local restructuring algorithm checks all the interior nodes for possible diagonal flipping. This helps ensure that the cells retain an adequate geometrical structure.

Plasma Equilibrium

To test the adaptive mesh strategy, free-boundary plasma equilibrium configurations in axisymmetric toroidal devices on an unstructured triangular mesh are calculated. Fixed-boundary equilibrium configurations are preferably solved in flux coordinates because they automatically resolve the structure of the magnetic surface.[18] However, since the presence of a separatrix in the vacuum region can greatly complicate the numerical algorithm, flux coordinates are not practical for free-boundary calculations. Furthermore, by their very nature, flux coordinates cannot properly treat geometries with multiple magnetic axes such as those found in multipole machines.[19] At the other end of the spectrum, fixed grid algorithms are relatively simple to implement for free- or fixed-boundary configurations with single or multiple singularities, but they require a large number of grid points to resolve the crowded flux surfaces at high beta.[20] Unless the structure of the plasma equilibrium is initially known at the onset of the computation, one must provide a fine gridding over the entire physical domain.

Adaptive triangular meshes retain the better features of both the flux- and the fixed-coordinate systems, at the expense of more complicated data structures. Since all calculations are performed on the physical plane with dynamically reconnecting triangular meshes that are free from overlap, the numerical algorithm adopted to solve the equilibrium equation is easily implemented in both fixed- and free-boundary modes.[12] Excessive nodal points in the mildly varying regions are avoided by properly defining the monitor surface.

Monitor Surface

Many of the stability results for high-beta equilibrium configurations under small perturbations can be traced to the structure of the safety factor profile, particularly its value at the magnetic axis. Accordingly, to accurately calculate the position of the magnetic axis, the first term in the monitor surface definition is inversely proportional to the magnitude of the flux gradient, which is zero at the magnetic axis. With only a concern for the axis resolution, the monitor surface amplitude is given by

$$M(x,z)=\frac{1}{|\nabla\psi(x,z)|+\epsilon} \tag{12}$$

which decreases away from the axis. The variables x, z are the radial and axial coordinates in the poloidal plane (Fig. 4) and ϵ is a small positive parameter to prevent division by zero.

As beta, the ratio of internal to magnetic energy in the plasma increases, the magnetic flux and the current density vary rapidly in localized regions. This results in a loss of accuracy in the numerical solution if the mesh cannot resolve the small spatial length scales. Rather than directly using the flux function as is most commonly done or even the gradient as above, it is found that better solutions are obtained when the nodes move to resolve the current density surface. The current density J_T is proportional to the source term in the equilibrium equation. It is calculated by applying a second-order differential operator to the flux function, which accentuates the gradients already present in the flux profile. On an intuitive level, the need to resolve the current is also evident since it rapidly changes from zero in the vacuum to nearly peak values just inside the plasma. From a more analytical standpoint, a good resolution of the current profile leads to a good discrete representation of the differential operator, and consequently of the flux, obtained from its inversion. Furthermore, the sharp current profile near the outer edge of the plasma delineates the plasma boundary much better than the flux, so the plasma-vacuum interface can be captured by node clustering. Both the magnetic axis and the current density can be readily detected by defining the monitor surface

$$M(x,z) = \alpha_1\beta_1 \frac{1}{|\nabla\psi(x,z)| + \epsilon} + \alpha_2\beta_2 |J_T(x,z)| \tag{13}$$

where the two coefficients β_1 and β_2 normalize their respective terms to unity. This normalization is chosen arbitrarily in the absence of other criteria. The two remaining parameters α_1 and α_2 are chosen by the user to control the respective importance attributed to the two terms.

Results

To test the accuracy of the discrete representation of the Grad-Shafranov equation, we reproduce the analytic solution obtained by Solove'v, in which the toroidal current density is proportional to the radial coordinate, x.[12] The plasma boundary calculated from the exact solution, is kept fixed during the iterative process. The maximum relative error between the computed and exact flux profiles is 3%. Figures 5 and 6 compare the solution on a fixed and adaptive mesh, respectively. Even though the current profile is linear in x, and constant in the vertical direction z, it falls abruptly to zero at the boundary.

After every five iterations of the main equilibrium solver, the grid adapts to the monitor surface [Eq. (13)]. This is accomplished by two applications of Eq. (11) at every interior grid point. If the number of iterations between node movements is less than five, the convergence rate is extremely low, while if much greater than five, the high curvature regions can escape the buffer regions. With $\alpha_1 = 2$, and $\alpha_2 = 1$ in Eq. (13), the magnetic axis is well resolved (Fig. 6a). The flux contours form concentric curves about

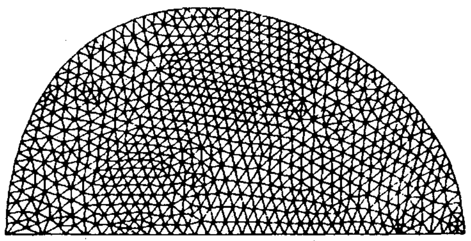
Fig. 5a Fixed mesh (Solove'v equilibrium).

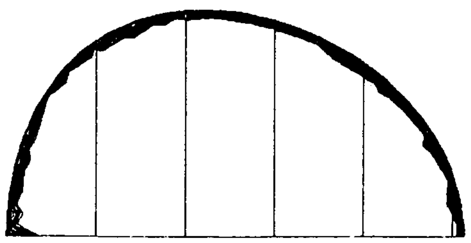
Fig. 5b Toroidal plasma current density on fixed mesh (Solove'v equilibrium).

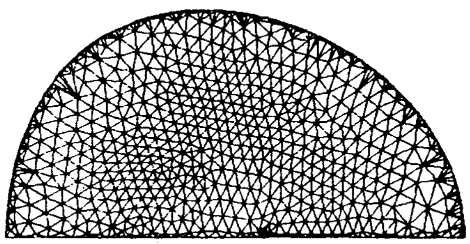
Fig. 6a Adapted mesh (Solove'v equilibrium).

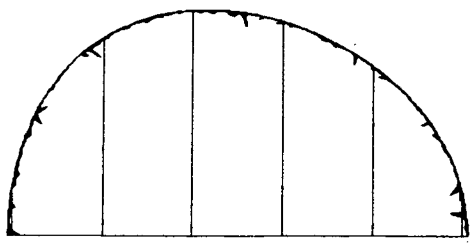
Fig. 6b Toroidal plasma current density on adapted mesh (Solove'v equilbirum).

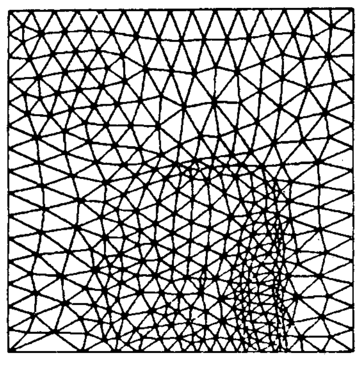
Fig. 7a Mesh adapted to initial current density in free-boundary equilibrium calculation.

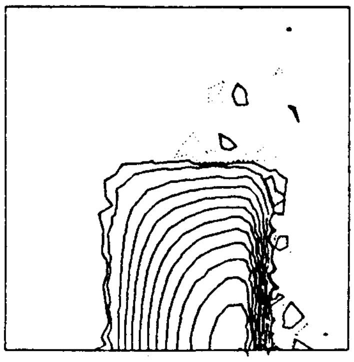
Fig. 7b Initial current density for free-boundary equilibrium calculation.

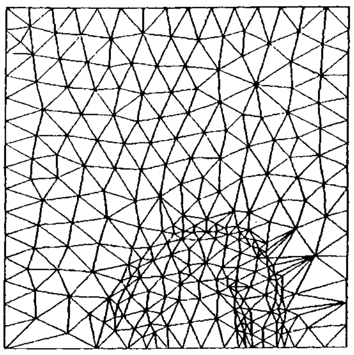
Fig. 8a Converged current density (free-boundary equilibrium).

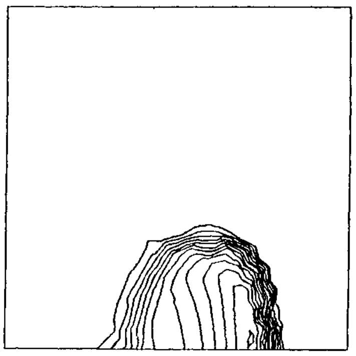
Fig. 8b Converged mesh (free-boundary equilibrium).

the magnetic axis. Notice the clustering of the triangles at the plasma boundary in Fig. 6a. A comparison of Figs. 5b and 6b, which show the current profile on the fixed and adaptive mesh, respectively, clearly exhibits the sharper current discontinuity in the adaptive case. The spikes in Fig. 6b correspond to triangle edges that are badly oriented relative to the boundary, and could be removed by better restructuring routines. The extremely flattened triangles at the boundary do not deteriorate the solution, contrary to expectation. It is conjectured that the triangles must be well structured on the monitor surface, and not on the physical plane, although this has not been proven at the present time.

The free-boundary calculation is performed in a rectangular domain limited by the vacuum vessel boundaries (Fig. 4). Exterior toroidal current coils surround the vessel and generate the magnetic fields necessary to balance the outward pressure force in the plasma. A self-consistent solution to the equilibrium is calculated. The poloidal flux at the vacuum walls is due to the exterior current coils and to the induced current in the plasma. When the distribution of current in the plasma and the location of the plasma boundary have both converged to steady states, an equilibrium is achieved. The exterior coils are symmetrically located about the midplane ($z=0$), so that the solution need only be displayed in the upper plane $z>0$. Starting with an initial current profile, 10 successive applications of Eq. (11) over the mesh results in the node distribution presented in Fig. 7a. Most noticeable is the vertical double band of triangles due to the high curvature regions of the monitor surface. The first corresponds to the current peak at the midplane, which tapers off at higher values of z, while the second is located just inside the plasma boundary, where the current abruptly falls to zero (Fig. 7b). The 400 nodes used in the computation were not sufficient to resolve the magnetic axis, perhaps because the curvature pull, equal to 50, was too strong. However, when it was reduced, the double band disappeared. At the present time, it seems that the method is very sensitive to the pulling parameter values. In its converged state, the plasma boundary is rounded off, and occupies a smaller fraction of the vacuum chamber (Fig. 8). However, the outer edge is still well defined by the current gradients. The two bands, present in Fig. 7a, have contracted, and the triangles are smaller. Larger triangles fill the vacuum region, which is for the most part devoid of interesting physics. An insufficient number of nodes in the plasma region prevents the magnetic axis from being seen. This problem could easily be solved with the addition of a couple of hundred points in the plasma region, but a lack of computer resources prevented the testing of this effect.

From the weight function definition, the curvature of the monitor surface determines the pulling strength on the nodes. Therefore, regions of lesser curvatures, e.g., the plasma boundary to the left of the magnetic axis, are not well resolved.

Conclusion

A general adaptive strategy was developed for dynamically moving and restructuring triangular meshes. It was applied to the study of plasma equilibria where the capability to capture free boundaries was demonstrated. In conjunction with better restructuring algorithms, the adaptive techniques developed herein are readily applicable to the study of compressible flows around complicated geometries, where internal shocks and boundaries are found. Moreover, from a more general perspective, these methods are also applicable to a wide range of additional physical problems.

Acknowledgments

This study was supported by NASA under Grant NAG1-355, the U.S. Air Force under Grant AFOSR-82-0176, and by DOE under Grant DE-AC02-76753016.

References

[1]Anderson, D. A., "Adaptive Mesh Schemes Based on Grid Speeds," *Proceedings of the AIAA 6th Computational Fluid Dynamics Conference*, AIAA, New York, 1983, pp. 311-318.

[2]Dwyer, H. A., "A Discussion of Some Criteria for the Use of Adaptive Gridding," *Proceedings of the AIAA 6th Computational Fluid Dynamics Conference*, AIAA, New York, 1983, pp. 319-322.

[3]Eiseman, P. R., "Alternating Direction Adaptive Grid Generation," *AIAA Journal*, Vol. 23, April 1985, pp. 551-560.

[4]Ghia, K., Ghia, U., and Shin, C. T., "Adaptive Grid Generation for Flows with Local High Gradient Regions," *Advances in Grid Generation*, edited by K. N. Ghia and U. Ghia, FED-Vol. 5, ASME, New York, 1983, pp. 35-48.

[5]Miller, K. and Miller, R. N., "Moving Finite Element I," *SIAM Journal of Numerical Analysis*, Vol. 18, No. 6, Dec. 1981, pp. 1019-1032.

[6]Salzman, J., "A Variational Method for Generating Multidimensional Adaptive Grids," Ph.D. Thesis, New York Univ., 1981.

[7]White, A. B., "On the Numerical Solution of Initial Boundary-Value Problems in Space Dimension," *SIAM Journal of Numerical Analysis*, Vol. 19, Aug. 1982, pp. 683-697.

[8]Berger, M., Gropp, W., and Oliger, J., "Grid Generation for Time-Dependent Problems: Criteria and Methods," Numerical Grid Generation Techniques, *Proceedings of NASA Langley Workshop*, NASA CP2166, Oct. 1980, pp. 181-188.

[9]Fritts, M. J. and Boris, J. P., "The Lagrangian Solution to transient Problems in Hydrodynamics Using a Triangular Mesh," *Journal of Computational Physics*, Vol. 31, No. 2, May 1979, pp. 173-215.

[10]Dannenhoffer III, J. F. and Baron, J. R., "Adaptive Procedure for Steady State Solution of Hyperbolic Equations," AIAA Paper 84-0005, Jan. 1984.

[11]Winslow, A. M., "Numerical Solution of the Quasilinear Poisson Equation in a Nonuniform Triangle Mesh," *Journal of Computational Physics*, Vol. 1, No. 2, Nov. 1966, pp. 149-172.

[12]Erlebacher, G., "Solution Adaptive Triangular Grids with Application to Plasma Equilibrium," Ph.D. Thesis, Columbia Univ., 1984.

[13]Eiseman, P. R., "Grid Generation for Fluid Mechanics Computations," *Annual Review of Fluid Mechanics*, Vol. 17, Jan. 1985, pp. 487-522.

[14]Anyiwo, J. A., "Idealized Dynamic Grid Computation of Physical Systems," *Numerical Grid Generation*, edited by J. F. Thompson, North-Holland, Nashville, April 1982.

[15]Crowley, W. P., "Free Lagrange Method for Numerically Simulating Hydrodynamic Flows in Two Dimensions," *2nd International Conference on Numerical Methods in Fluid Dynamics*, Berkeley, 1970.

[16]Harten, A. and Hyman, J. M., "Self-Adjusting Grid Methods for One-Dimensional Hyperbolic Conservation Laws," *Journal of Computational Physics*, Vol. 50, No. 2, May 1983, pp. 235-269.

[17]Ablow, C. M. and Schecter, S., "Campolytropic Coordinates," *Journal of Computational Physics*, Vol. 27, No. 3, June 1978, pp. 351-362.

[18]Delucia, J., Jardin, S. C., and Todd, A. M., "An Iterative Metric Method for Solving the Inverse Tokamak Equilibrium Problem," *Journal of Computational Physics*, Vol. 37, No. 2, Sept. 1980, pp. 183-204.

[19]Feneberg, W. and Lackner, K., "Multipole Takamak Equilibria," *Nuclear Fusion,* Vol. 13, No. 4, Aug. 1973, pp. 549-556.

[20]Johnston, J. L., Dalhed, H. E., Greene, J. M., et al., "Numerical Determination of Axisymmetric Toroidal MHD Equilibria," Rept. PPPL-1463, Dept. of Energy, Princeton Plasma Physics Laboratory, 1978.