JOURNAL OF COMPUTATIONAL PHYSICS 67, 124–144 (1986)

A Two-Dimensional Mesh Moving Technique for Time-Dependent Partial Differential Equations*

DAVID C. ARNEY

Department of Mathematics, United States Military Academy, West Point, New York 10996

AND

JOSEPH E. FLAHERTY

Department of Computer Science, Rensselaer Polytechnic Institute, Troy, New York 12180-3590

Received April 17, 1985; revised January 15, 1986

We discuss an adaptive mesh moving technique that can be used with a finite difference or finite element scheme to solve initial-boundary value problems for systems of partial differential equations in two space dimensions and time. The mesh moving technique is based on an algebraic node movement function determined from the geometry and propagation of regions having significant discretizaion error indicators. Our procedure is designed to be flexible, so that it can be used with many existing finite difference and finite element methods. To test the mesh moving algorithm, we implemented it in a system code with an initial mesh generator and a MacCormack finite difference scheme on quadrilateral cells for hyperbolic vector systems of conservation laws. Results are presented for several computational examples. The moving mesh scheme reduces dispersive errors near shocks and wave fronts and thereby reduces grid requirements necessary to compute accurate solutions while increasing computational efficiency.

1. INTRODUCTION

Mesh moving is an adaptive technique that has been used successfully to improve the accuracy of both finite element and finite difference schemes for a variety of time-dependent problems in one (cf., e.g., [1, 2, 4, 14, 15, 17, 21, 24, 27, 33]) and two (cf., e.g., [12, 32, 33, 35]) space dimensions. The essential idea is to move the

* The authors were partially supported by the U.S. Army Research Office under Contract DAAG 29-82-K-0197 and the U. S. Air Force Office of Scientific Research, Air Force Systems Command, U.S.A.F., under Grant AFOSR 80-0192. This work was used to partially fulfill the Ph.D. requirements of the first author at the Rensselaer Polytechnic Institute.

0021-9991/86 $3.00

mesh either to minimize some quantity, such as the discretization error, or to follow some local nonuniformity, such as a wave front. This generally reduces dispersive errors and Courant number restrictions.

In one dimension, Hyman [27] described a mesh moving scheme for hyperbolic conservation laws that minimized the time variation of the finite difference solution at the nodes. Davis and Flaherty [14] and Adjerid and Flaherty [1, 2] developed finite element codes for parabolic systems that moved a mesh so as to equidistribute the spatial component of the discretization error. Miller *et al.* [21, 29, 30] simultaneously determined the numerical solution and the node positions using a finite element method that minimized the residual for parabolic problems. Bell and Shubin [5] solved the Euler–Lagrange equations of an extremizing functional and used a finite difference scheme to solve hyperbolic conservation laws. All of these schemes have demonstrated that mesh moving can reduce discretization errors and provide improvements in computational efficiency.

With some modification, the methods of Adjerid and Flaherty [1, 2], Hyman [27], and Miller *et al.* [21, 29, 30] can be extended to higher dimensions; however, many other mesh moving techniques are not directly applicable to two- and three-dimensional problems. One difficulty is that equidistribution strategies fail to produce unique solutions. Brackbill and Saltzman [12, 35] have overcome this problem by adding the constraints of mesh smoothness and orthogonality to a variational problem.

A successful mesh moving scheme for higher dimensional problems, that is, somewhat similar to the method presented here, is the algorithm of Rai and Anderson [32, 33, 34]. Their technique is based on a gravitational potential where nodes with higher- or lower-than-average discretization errors, respectively, attract or repel other nodes. The strength of the attraction or repulsion diminishes with increasing distance between nodes. Since each node affects all other nodes, a global calculation is necessary to determine each node's velocity.

Local mesh refinement is a different adaptive technique that consists of dividing or refining elements in regions where the solution is not adequately resolved. The advantage of this technique relative to mesh moving is that enough fine grids can be added to resolve the small-scale structures of the solution and provide solutions to within user-prescribed error tolerances. The local mesh refinement schemes of Berger [7, 8, 9], Flaherty and Moore [20], Gannon [22], and Bieterman and Babuska [10, 11] successfully satisfied prescribed error tolerances for different problems using finite element or finite difference schemes. The methods of Berger [7, 8, 9] and Gannon [22] have also been applied to two-dimensional problems.

The most promising algorithms appear to be those that combine both mesh moving and local mesh refinement. It is our intention to consider such schemes; however, except for calculating an initial mesh, the computational procedures discussed here do not contain local refinement.

The mesh moving technique that we have developed is simple, efficient, and independent of the numerical method being employed to discretize the partial differential equations. At each time-step, it uses the node locations and the nodal

values of an error indicator, such as an estimate of the local discretization error or the solution gradient to control grid motion. Nodes with "significant error indicators" (cf. Sect. 2) are grouped into rectangular error clusters which separate spatially distinct phenomena of the solution. As time evolves, the clusters can move, change size, change orientation, collide, separate, reflect off boundaries, or pass through boundaries. At each time-step, new clusters can be created and old ones can vanish. The clustering algorithms we use are briefly described in Section 2 and were developed by Berger [7, 9] for a mesh refinement scheme.

Mesh movement is determined by a node's relationship to the error clusters. Movement is done in two steps, each in a direction along a principal axis of a cluster rectangle. The amount of movement in each direction is determined by a movement function which moves the center of the cluster according to a differential equation suggested by Coyle, Flaherty, and Ludwig [13]. Additionally, the movement function smoothes mesh motion, reduces distortion and mesh tangling, and prevents nodes from moving outside the domain boundaries.

In Section 2, we discuss error clustering, movement of the center of mass of the cluster, the node movement function, and the initial mesh generator. In Section 3, we discuss the MacCormack finite difference scheme for hyperbolic equations and the error indicators used in the computational examples. The results of the computational examples are given in Section 4, and Section 5 contains a discussion of the results of the experiments and the status of our algorithm.

2. Mesh Moving Scheme and Initial Mesh Generation

We discuss a mesh moving scheme and an initial mesh generator that can be used in conjunction with a numerical procedure to solve time-dependent partial differential systems on a rectangular domain. Suppose that the domain is discretized into a moving mesh of quadrilateral cells having vertices (or nodes), $(x_i(t), y_i(t))$, $i = 1, 2, \ldots, N$, that are numbered in a row sequential fashion. A representative cell and a sample mesh are shown in Figs. 3 and 4, respectively. We further suppose that an approximate solution vector $\mathbf{u}_i(t)$, $i = 1, 2, \ldots, N$, of the partial differential system and a nonnegative scalar error indicator, $e_i(t)$, $i = 1, 2, \ldots, N$, are to be calculated at each node at time $t > 0$. The error indicator can be related to the local discretization error at a node; however, quantities proportional to the solution gradient, curvature, etc., can also be used. The error indicator serves to attract nodes, and, thus, it should be large where the mesh should be fine and small where the mesh should be coarse. The mesh moving scheme is discussed first in Section 2.1 and the discussion of the initial mesh generator follows in Section 2.2.

2.1. *Mesh Moving Scheme*

Suppose the mesh, solution, and error indicator described above have been calculated up to a time $t > 0$. We scan the mesh at time t and flag "significant error nodes" as nodes having error indicators greater than twice the mean nodal error

indicator and also greater than a user-supplied tolerance. This empirical strategy avoids having the mesh respond to fluctuations when error indicators are too small, but is sensitive enough to avoid missing dynamic phenomena associated with large error indicators. If there are no significant error nodes, computation is performed on a stationary mesh. The nearest neighbor clustering algorithm of Berger [7, 9] is used next to cluster the significant error nodes. In this iterative algorithm, a cluster is first defined to consist of one arbitrary significant error node. Other significant error nodes are added to the cluster if they are within a specified minimum intercluster distance from the nearest node in the cluster. New clusters are established for nodes that do not belong to any existing cluster. Clusters are united when a node is determined to belong to more than one of them. Upon completion of the algorithm, (i) nodes in different clusters will be separated by at least the minimum intercluster distance, and (ii) no node in a cluster with more than one node will be further than the minimum intercluster distance from its nearest neighbor in the cluster. The minimum intercluster distance is chosen to be the length of a cell diagonal.

Berger [7, 9] shows that near minimum area rectangles that contain a cluster can be easily generated. The principal axes of shuch a rectangle are the major and minor axes of an enclosed ellipse with the same first and second moments as the clustered nodes. Thus, if x_m and y_m are the mean coordinates of the clustered nodes, then the axes of the rectangle are in the directions of the eigenvectors of the symmetric (2×2) matrix

$$\begin{bmatrix} \sum (x_i^2 - x_m^2) & \sum (x_i y_i - x_m y_m) \\ \sum (x_i y_i - x_m y_m) & \sum (y_i^2 - y_m^2) \end{bmatrix}. \tag{2.1}$$

The summations range over all nodes in the cluster.

For problems with significant error nodes located on a long curved arc, the entire region may belong to one unacceptably large cluster. To prevent this ineffeciency and provide better alignment with curved fronts, the rectangular clusters are checked for efficiency by determining the percentage of significant error nodes in the cluster. If a 50% efficiency is not achieved, the rectangle is iteratively bisected in the directon of the major axis. This is repeated until all clusters have a 50% efficiency or more. This nearest neighbor clustering separates spatially distinct phenomena, as shown by the dashed line error clusters on the mesh of Fig. 13, and provides some alignment with long curved error regions, as shown by the clusters in Figs. 14 and 15.

We determine node movement from the velocity of propagation, the orientation, and the size of error clusters. We do not follow or track surfaces of discontinuities exactly. We assume that nodes in the same cluster have related solution characteristics, so that we can determine individual node movement from the propagation of the center of mass of the error cluster.

Hyman [27] and Harten and Hyman [24] developed a mesh moving algorithm

for hyperbolic systems that moved the mesh with a weighted average of the characteristic velocities when multiple waves interacted. The same principle applies to our algorithm when several error clusters merge due to, e.g., wave interaction. The mesh moves with a velocity given by a weighted average of the velocities of the intersecting error clusters. Comparisons between the center of mass propagation of an error cluster and the characteristic path of the center of the cluster are given in Example 4.2.

Coyle *et al.* [13] showed that mesh movement can be unstable in certain situations. Following one of their suggestions, we move clusters according to the differential equation

$$\ddot{\mathbf{r}} + \lambda \dot{\mathbf{r}} = 0, \tag{2.2}$$

where $\mathbf{r}(t)$ is the position vector of the center of an error cluster and a superimposed dot denotes differentiation with respect to t. Equation (2.2) is conditionally stable (cf. Coyle *et al.* [13]), and when solved numerically with reasonable choices of $\lambda > 0$ instabilities and oscillations in the mesh motion were not present. The choice of the parameter λ can be critical in certain situations. If λ is selected too large, the system (2.2) will be stiff and computationally expensive. On the other hand, if λ is selected too small, the mesh can oscillate from time step-to-time step. Coyle *et al.* [13] and Adjerid and Flaherty [2] suggest some adaptive procedures for choosing λ; however, we found no appreciable differences in results or computation times when λ was varied for the example considered in Section 4. These results were all calculated with $\lambda = 1$, but more study of the effects of this parameter are needed.

We solve (2.2) from t_{n-1} to t_n and then for each cluster determine $\mathbf{r}(t_{n+1})$ and the vector $\mathbf{r}(t_{n+1}) - \mathbf{r}(t_n)$ which is projected onto the two principal directions of the rectangular cluster at t_n. These projected distances are the amounts by which the center of mass of the error cluster moves in each principal direction. Let Δr_1 and Δr_2 denote these projections and let CM denote the center of mass of the error cluster. We create a one-dimensional mesh movement function to move the nodes of the mesh along the two axial directions of the error clusters. A profile of our movement function is shown in Fig. 1; however, the algorithm is designed to be used with any one-dimensional movement function. The slope of the movement function depends on the length of the side of the cluster which is denoted as w in Figs. 1 and 2.

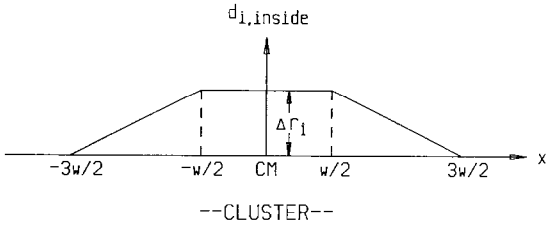

FIG. 1. Profile of the node movement function (cf. Eq. (2.3)).

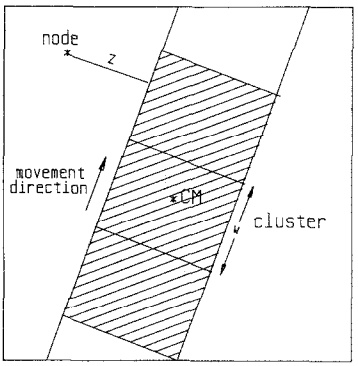

FIG. 2. A node shown outside the range of an error cluster. The distance z is used in Eq. (2.4) to determine the amount of movement for this node.

As shown in Fig. 1, nodes inside the range of the cluster (shaded in Fig. 2) are moved a distance $d_{i,\text{inside}}$, $i = 1, 2$, in each principal direction given by

$$d_{i,\text{inside}} = \begin{cases} \Delta r_i(3/2 - x/w) & \text{if } w/2 \leqslant x \leqslant 3w/2 \\ \Delta r_i & \text{if } -w/2 < x < w/2 \\ \Delta r_i(3/2 + x/w) & \text{if } -3w/2 \leqslant x \leqslant -w/2, \quad i = 1, 2. \end{cases} \tag{2.3}$$

Here, x is the projected distance in a principal direction of a node relative to the center of mass of the error cluster. To provide smooth node movement throughout the domain, nodes outside the range of the cluster move in a reduced amount determined by

$$d_{i,\text{outside}} = d_{i,\text{inside}}[1 - (2z/D)], \tag{2.4}$$

where z is the shortest distance to the range of the cluster (cf. Fig. 2) and D is the diagonal distance of the entire domain. Node movement distances $d_{i,\text{inside}}$ and $d_{i,\text{outside}}$ are reduced near boundaries to prevent nodes from leaving the domain. In particular, for nodes moving towards the edge of the domain, we recalculate $d_{i,j}$ as $d_{i,j}[\min(1, b/c)]$, $i = 1, 2$, $j = \text{inside, outside}$, where b is the distance of the node to the boundary and c is twice the length of a cell diagonal on a uniform mesh having the same number of cells as the moving mesh. Nodes on domain boundaries, except corner nodes which are not moved, are restrained to move along the boundary.

2.2. *Initial Mesh Generation*

The generation of a proper initial mesh is critical to the success of the mesh moving scheme. Without refinement, the mesh moving algorithm cannot provide suitable error control unless the initial mesh spacing properly resolves initial data. An initial error measure appropriate for the finite difference scheme of Section 3 is the error in interpolating the prescribed initial condition $\mathbf{u}_0(x, y)$ on each cell by a

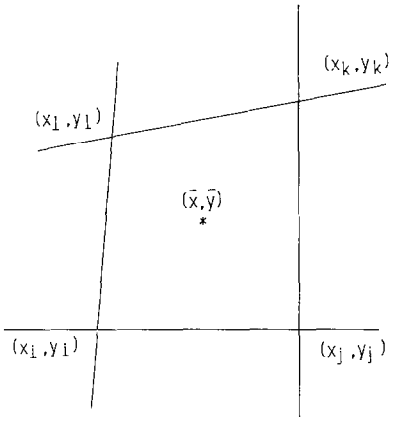

FIG. 3. Node labelling of an arbitrary quadrilateral cell.

bilinear polynomial. The error on each cell is determined as the difference between the value of the initial function and its bilinear interpolant at the center of each cell. Therefore, the initial mesh must be generated so that the condition

$$\|\tfrac{1}{4}\{\mathbf{u}_0(x_i, y_i) + \mathbf{u}_0(x_j, y_j) + \mathbf{u}_0(x_k, y_k) + \mathbf{u}_0(x_l, y_l)\} - \mathbf{u}_0(\bar{x}, \bar{y})\|_\infty < \text{TOL} \quad (2.5)$$

holds on each cell when using the vertex and center point labelling of a cell, as shown in Fig. 3 and a prescribed tolerance TOL. We satisfy condition (2.5) using an iterative scheme that begins by generating a uniform mesh and computing an initial error estimate from the left side of (2.5), we cluster nodes of high error and move them towards the centers of the clusters, and recompute the initial error. We then iteratively add rows and columns to the mesh wherever the error tolerance is exceeded, compute a new mesh by the scheme of Brackbill and Saltzman [12], and recompute the error until the prescribed tolerance is satisfied.

Initial meshes generated with this algorithm are shown in Figs. 4, 6, 13, and 14 for the initial conditions of the computational examples of Section 4. Any initial mesh generator that satisfies condition (2.5) could be used with our mesh moving scheme, and several such algorithms can be found in Thompson [36].

3. MacCormack Finite Difference Solver and Error Indication

To test our mesh moving scheme, we used the explicit finite difference MacCormack scheme on nonuniform quadrilateral grids for hyperbolic vector systems of conservation laws having the form

$$\mathbf{u}_t + \mathbf{f}_x(x, y, \mathbf{u}, t) + g_y(x, y, \mathbf{u}, t) = 0, \quad (3.1)$$

$$\mathbf{u}(x, y, 0) = \mathbf{u}_0(x, y), \quad (3.2)$$

with appropriate well-posed boundary conditions.

To discretize (3.1), we introduce a transformation

$$\xi = \xi(x, y, t), \qquad \eta = \eta(x, y, t), \qquad \tau = t, \tag{3.3}$$

from the physical (x, y, t) domain to a computational (ξ, η, τ) domain where a uniform rectangular grid will be used. Under this transformation, (3.1) becomes

$$\mathbf{u}_\tau + \mathbf{u}_\xi \xi_t + \mathbf{u}_\eta \eta_t + \mathbf{f}_\xi \xi_x + \mathbf{f}_\eta \eta_x + \mathbf{g}_\xi \xi_y + \mathbf{g}_\eta \eta_y = 0. \tag{3.4}$$

The transformation metrics $(\xi_x, \xi_y, \xi_t, \eta_x, \eta_y, \eta_t)$ are related to the metrics $(x_\xi, x_\eta, x_\tau, y_\xi, y_\eta, y_\tau)$ of the inverse mapping of the computational domain to the physical domain by the identities

$$\begin{aligned} &\xi_x = y_\eta/J, \qquad \xi_y = -x_\eta/J, \qquad \xi_t = (y_\tau x_\eta - x_\tau y_\eta)/J, \\ &\eta_x = -y_\xi/J, \qquad \eta_y = x_\xi/J, \qquad \eta_t = (y_\xi x_\tau - x_\xi y_\tau)/J, \\ &\qquad\qquad J = y_\eta x_\xi - x_\eta y_\xi. \end{aligned} \tag{3.5}$$

Using (3.5) in (3.4) gives

$$\begin{aligned} &\mathbf{u}_\tau + \mathbf{u}_\xi (y_\tau x_\eta - x_\tau y_\eta)/J + \mathbf{u}_\eta (y_\xi x_\tau - x_\xi y_\tau)/J \\ &\quad + \mathbf{f}_\xi y_\eta/J + \mathbf{f}_\eta(-y_\xi/J) + \mathbf{g}_\xi(-x_\eta/J) + \mathbf{g}_\eta x_\xi/J = 0. \end{aligned} \tag{3.6}$$

We discretized (3.6) by the MacCormack scheme using first-order-forward difference approximations in the predictor step and first-order-backward differences in the corrector step. It was shown by Hindman [25, 26] that this differencing of Eqs. (3.4) or (3.6) produces consistent and conservative approximations. We automatically adjust the time step so as to satisfy the Courant, Friedrichs, Lewy Theorem.

As previously noted, several quantities can be used as error indicators. In the computational examples of Section 4, we used either solution gradients of differences between the predicted and corrected solutions of MacCormack's method as error indicators. The latter error indicator is actually an estimate of the local discretization error of the predicted solution and not the second order corrected solution; however, it does have the proper propagation characteristics. Other more reliable error estimates will be needed when local mesh refinement is introduced. Error estimators that we are investigating are based on combining extrapolation and the difference between predicted and corrected solutions. Results of this method are reported in Arney [3]. Other possibilities are to use hierarchical approximations as done by, e.g., Adjerid and Flaherty [1, 2] and Zienkiewicz *et al.* [39].

4. Computational Examples

We solved a sequence of hyperbolic equations using the initial mesh generator of Section 2 and the MacCormack scheme of Section 3 as tests of our mesh moving technique. Significant error nodes are marked with an asterisk and rectangular error clusters are outlined with dashed lines in the figures accompanying the examples. As noted in Section 2, the parameter λ of Eq. (2.2) was selected as unity.

Example 4.1. Consider the linear scalar hyperbolic differential equation

$$u_t + u_x + u_y/4 = 0,\ t > 0, \quad -0.2 \leqslant x \leqslant 0.8,\ 0 \leqslant y \leqslant 1, \tag{4.1}$$

with initial conditions

$$u(x, y, 0) = \begin{cases} 0.8 & \text{if } y < -4x + 1.2 \\ 0.0 & \text{if } y > -4x + 1.6 \\ -8x - 2y + 3.2 & \text{otherwise,} \end{cases} \tag{4.2}$$

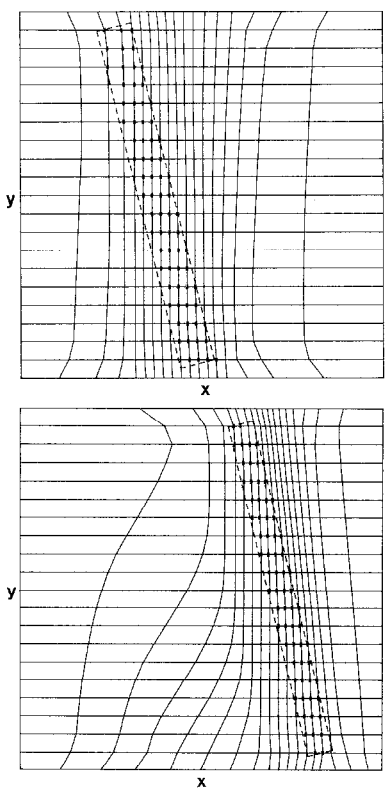

Fig. 4. Meshes of Example 4.1 at $t = 0.0$ (top) and at $t = 0.4$ (bottom).

and with Dirichlet boundary conditions

$$u(x, y, t) = \begin{cases} 0.8 & \text{if } y - 0.25t < -4(x-t) + 1.2 \\ 0.0 & \text{if } y - 0.25t > -4(x-t) + 1.6 \\ -8(x-t) - 2(y - 0.25t) + 3.2 & \text{otherwise.} \end{cases} \quad (4.3)$$

The solution of this problem is an oblique wave front that moves at an angle of 14 degrees across the domain. We selected it to show the concentration of the initial mesh within the front, the partial alignment of the initial mesh with the front, the propagation of the refined region of the mesh with the moving front, and the reduction of dispersive errors with a moving mesh. The magnitude of the gradient of the solution was used as an error indicator and a value of TOL = 0.01 was used in Eq. (2.5).

The computational meshes at $t = 0$ and $t = 0.4$ are shown in Fig. 4. At each time-step, the nodes are moving with the wave front at nearly the characteristic speed. This reduces dispersive errors. as shown in the surface plots of the solution in Fig. 5, which compare the solution calculated on the moving mesh with one calculated on a stationary uniform mesh having the same number of nodes. Observe

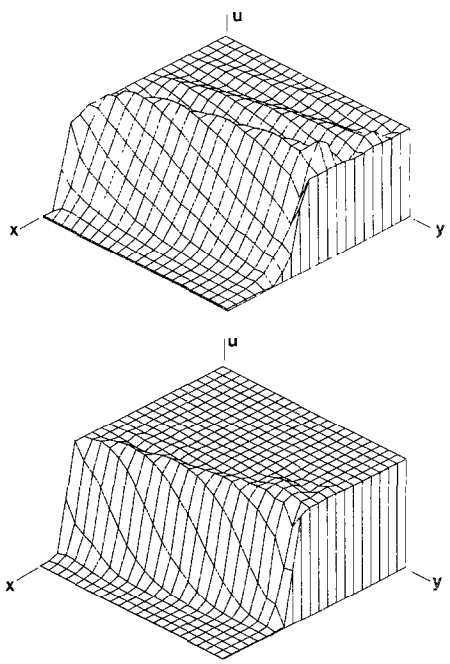

FIG. 5. Surface plots of the solution of Example 4.1 at $t = 0.4$ using a stationary uniform mesh (top) and a moving mesh (bottom). The moving mesh reduces numerical oscillations behind the propagating wave.

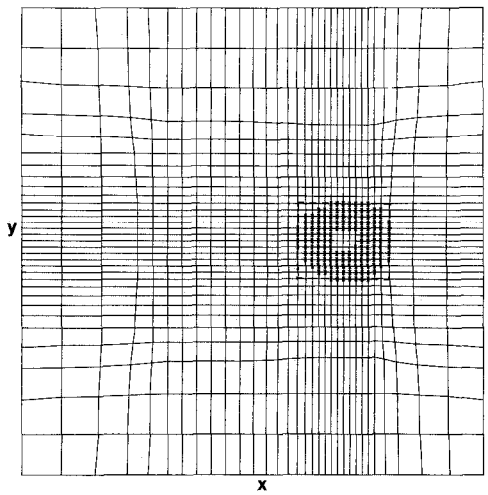

FIG. 6. Initial mesh for the rotating cone of Example 4.2.

EXAMPLE 4.2. Consider the initial boundary value problem

$$u_t - yu_x + xu_y = 0, \qquad t > 0,\ -1.2 \leqslant x,\ y \leqslant 1.2, \tag{4.4}$$

$$u(x, y, 0) = \begin{cases} 0 & \text{if } (x-1/2)^2 + 1.5y^2 \geqslant 1/16 \\ 1 - 16((x-1/2)^2 + 1.5y^2) & \text{otherwise,} \end{cases} \tag{4.5}$$

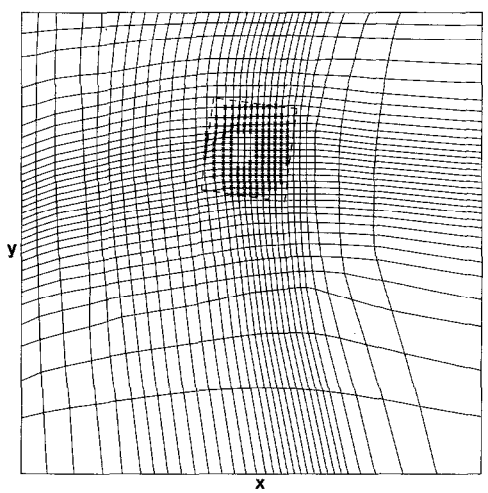

FIG. 7. Mesh of Example 4.2 at $t = 1.6$. Nodes are moving with the rotating cone.

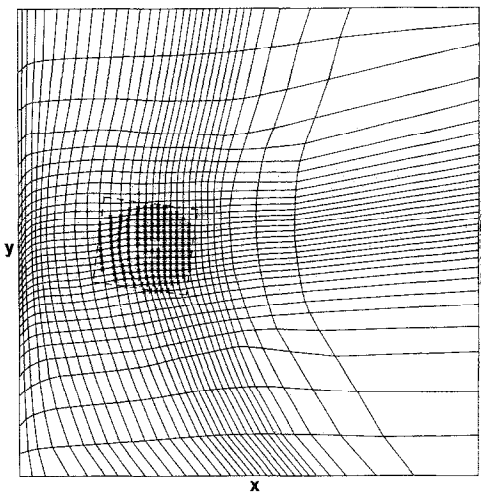

FIG. 8. Mesh of Example 4.2 at $t = 3.2$. Nodes continue to move with the rotating cone with some crowding near the boundary.

and

$$u(1.2, y, t) = u(-1.2, y, t) = u(x, -1.2, t) = u(x, 1.2, t) = 0. \tag{4.6}$$

The exact solution of this problem is

$$u(x, y, t) = \begin{cases} 0 & \text{if } C < 0 \\ C & \text{if } C \geqslant 0, \end{cases} \tag{4.7}$$

where

$$C = 1 - 16[(x \cos t + y \sin t - 1/2)^2 + 1.5(y \cos t - x \sin t)^2]. \tag{4.8}$$

Equations (4.7) and (4.8) represent a moving elliptical cone rotating counterclockwise around the origin with period 2π. This problem was proposed as a test problem by Gottlieb and Orszag [23] and we selected it because the rotational quality of the error region is a good test of a mesh moving scheme.

The initial mesh (cf. Fig. 6) has an interpolation error less than 0.08 and (as in Examples 4.3 and 4.4) the difference between the predicted and corrected steps of MacCormack's method was used as an error indicator. Figures 7 and 8 show the mesh at $t = 1.6$ and $t = 3.2$ respectively. The nodes follow the moving cone and keep it within the refined region. Figures 9 and 10 compare the contour and surface plots, respectively, of the solution at $t = 3.2$ on the moving mesh with one on a 32×32 uniform stationary mesh. The dispersive error distorts the cone and leaves a wake behind it. These errors are significantly reduced by the mesh moving scheme.

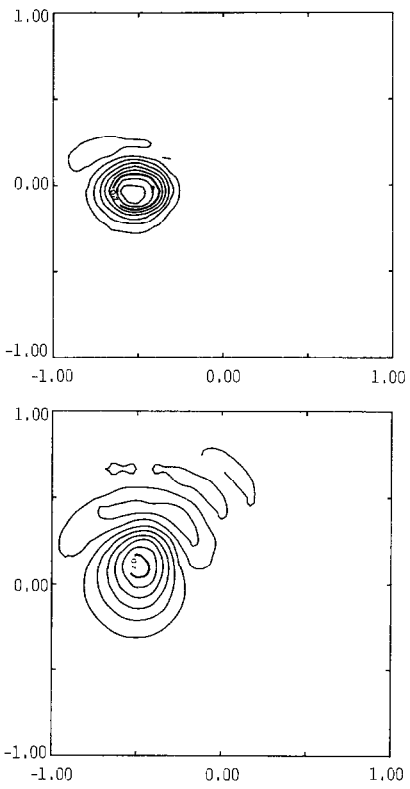

FIG. 9. Contour plots of solutions of Example 4.2 on a moving mesh (top) and a stationary mesh (bottom) at $t = 3.2$.

Figure 11 compares the path of the center of mass of the single error cluster using (2.2) and the real characteristic path of the peak of the cone. As expected for this scalar hyperbolic problem, the movement of the center of error mass determined by (2.2) closely approximates the characteristic path of the peak of the cone with a maximum difference of 4 percent in length and direction.

EXAMPLE 4.3. This problem is similar to Example 4.2 except there are now two symmetric cones rotating counterclockwise about the origin. The problem is given by Eq. (4.4), Eq. (4.6), and new initial conditions provided by

$$u(x, y, 0) = \begin{cases} 1 - 16((x-1/2)^2 + 1.5y^2) & \text{if } (x-1/2)^2 + 1.5y^2 \leqslant 1/16 \\ 1 - 16((x+1/2)^2 + 1.5y^2) & \text{if } (x+1/2)^2 + 1.5y^2 \leqslant 1/16 \\ 0 & \text{otherwise.} \end{cases} \quad (4.9)$$

Figure 12 shows the mesh at $t = 1.05$, which has poor aspect ratios and severe

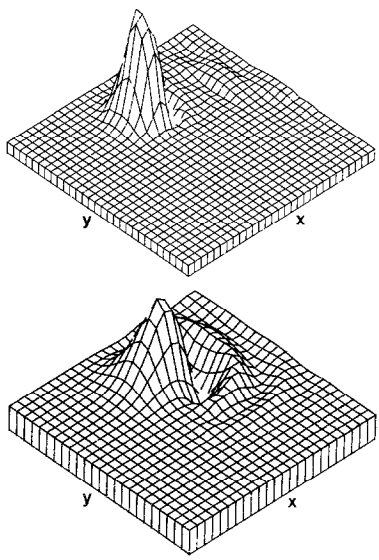

FIG. 10. Surface plots of solutions of Example 4.2 on a moving mesh (top) and a stationary mesh (bottom) at $t = 3.2$.

mesh distortion caused by the rotation of the error regions. The mesh tangles as the cones rotate further. When such mesh tangling occurs, a static rezone is necessary to create a new mesh. The rezoning can use an algorithm similar to the one that generated the initial mesh. The data at the new nodes must be obtained by interpolation from the calculated solution at the old nodes by a conservative rezoning as

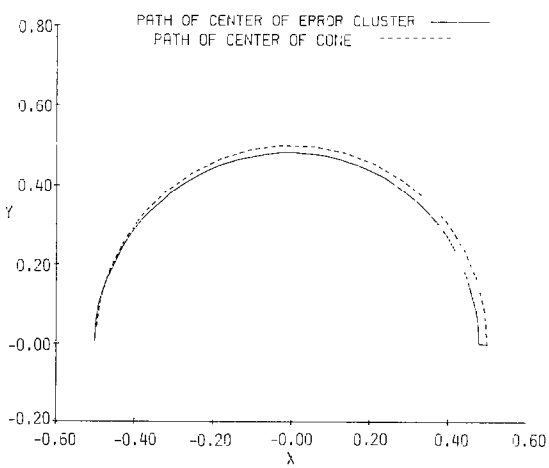

FIG. 11. Comparison of the characteristic path of the peak of the cone and the path of the center of error as determined by Eq. (2.2) for Example 4.2.

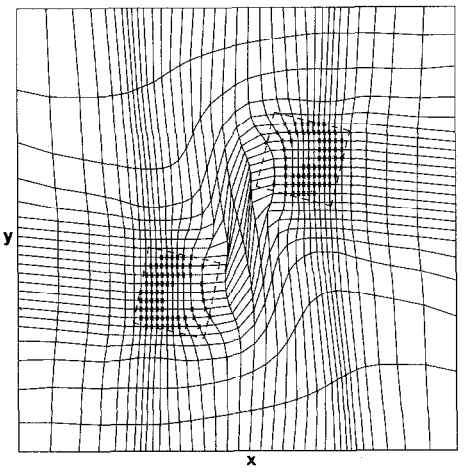

FIG. 12. Distorted mesh of Example 4.3 at $t = 1.05$ showing the need for static rezoning.

presented in Dukowicz [16]. A static rezone procedure accomplishing these goals is described in Arney [3].

EXAMPLE 4.4. Consider the uncoupled system

$$u_t + u_x = 0, \qquad v_t - v_x = 0, \qquad t > 0,\ -1 \leqslant x,\ y \leqslant 1, \tag{4.10}$$

$$u(x, y, 0) = \begin{cases} 1 - 16((x-1/2)^2 + 1.5y^2) & \text{if } (x-1/2)^2 + 1.5y^2 \leqslant \frac{1}{16} \\ 0 & \text{otherwise,} \end{cases} \tag{4.11a}$$

$$v(x, y, 0) = \begin{cases} 1 - 16((x+1/2)^2 + 1.5y^2) & \text{if } (x+1/2)^2 + 1.5y^2 \leqslant \frac{1}{16} \\ 0 & \text{otherwise} \end{cases} \tag{4.11b}$$

and

$$u(x, y, t) = v(x, y, t) = 0 \quad \text{on the boundary of the domain} \tag{4.12}$$

The solution of this problem is two moving cones that pass through one another. This causes the error clusters to collide and merge, and then later separate. Figure 13 shows the meshes used to solve this problem at $t = 0$, $t = 0.35$, $t = 0.9$, and $t = 1.3$. Initially there are two spatially distinct error clusters (cf. the upper left mesh of Fig. 13). At $t = 0.35$, the two clusters have collided and merged into a single cluster (cf. the upper right mesh of Fig. 13). From $t = 0.35$ to $t = 0.9$, the single cluster stays centered at the origin so the mesh does not move during this time. At $t = 0.9$, the cones have passed completely through one another and the single cluster has separated into two clusters which move towards the boundary of the region (cf. the lower left mesh of Fig. 13). Finally, at $t = 1.3$. The cones and error clusters have

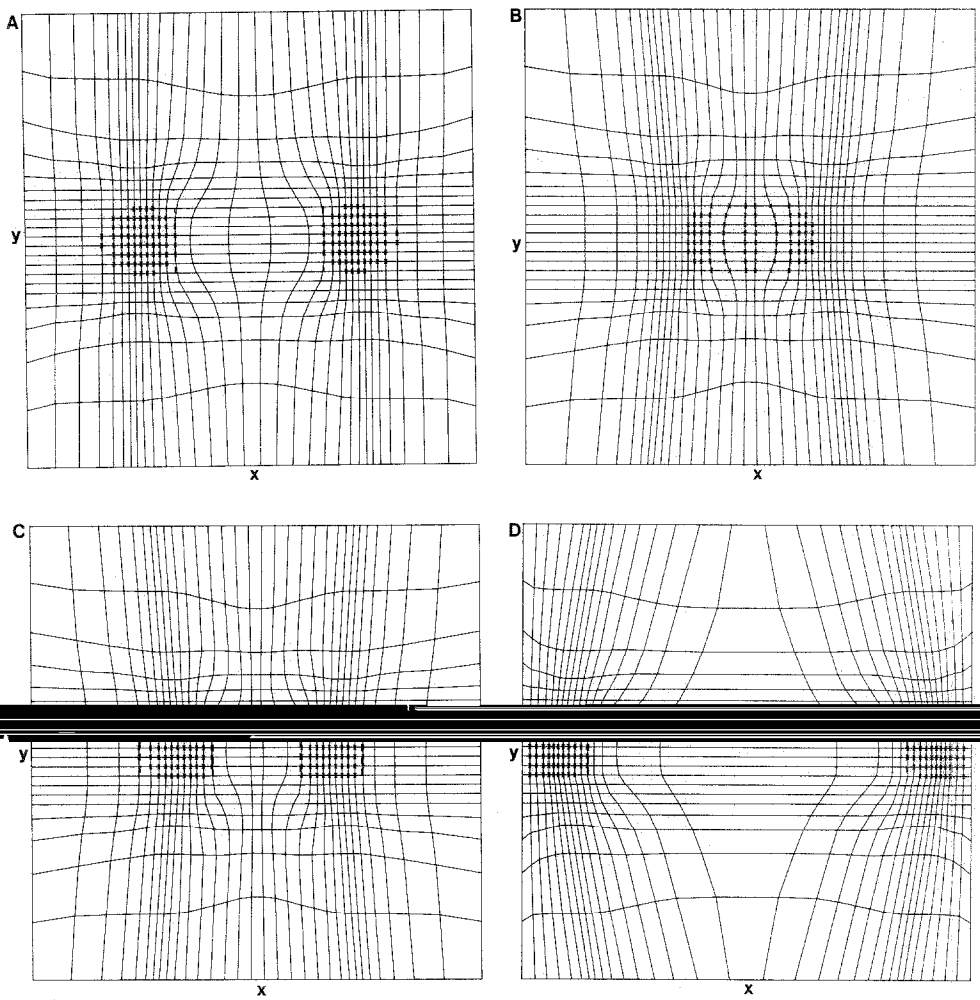

FIG. 13. Meshes of Example 4.4 at (A) $t=0$, (B) $t=0.35$, (C) $t=0.9$, and (D) $t=1.3$. Initially there are two spatially distinct error clusters, at $t=0.35$ the two clusters have merged into a single cluster, at $t=0.9$ the single cluster separates into two clusters, and at $t=1.3$ the two clusters have reached the domain boundary.

reached the domain boundary (cf. the lower right mesh of Fig. 13) and no further movement of the mesh will take place as the cones exit the domain.

EXAMPLE 4.5. Consider the Euler equations for a perfect inviscid fluid, which have the form of Eq. (3.13) with

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \tag{4.13a}$$

$$\mathbf{f}(\mathbf{u}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e+p) \end{bmatrix}, \tag{4.13b}$$

$$\mathbf{g}(\mathbf{u}) = \begin{bmatrix} \rho u \\ \rho uv \\ \rho v^2 + p \\ v(e+p) \end{bmatrix}. \tag{4.13c}$$

In the above equations, u and v are the velocity components in the x and y directions, ρ is the density, e is the total energy per unit volume, and p is the pressure which is given by

$$p = (\gamma - 1)[e - \rho(u^2 + v^2)/2]. \tag{4.13d}$$

We solve a problem where a Mach 10 shock moves down a channel containing a wedge. The computational domain, $-0.3 \leqslant x \leqslant 1.9$, $0 \leqslant y \leqslant 1$, is oriented along the

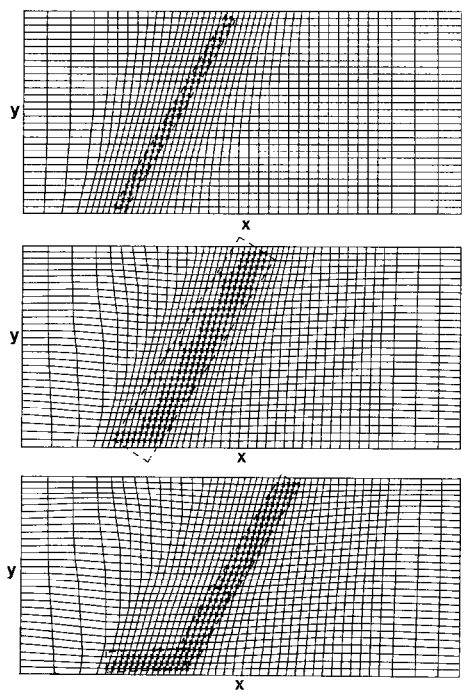

FIG. 14. Meshes of Example 4.5 at $t = 0$ (top), $t = 0.01$ (center), $t = 0.02$ (bottom).

wedge so that the wedge lies on the bottom boundary in the region $y=0$, $\frac{1}{6} \leqslant x \leqslant 1.9$. The initial conditions

$$\rho = 8.0, \quad p = 116.5, \quad e = 563.5, \quad u = 4.125\sqrt{3}, \quad v = -4.125, \quad \text{if} \quad y < \sqrt{3}(x - 1/6), \tag{4.14a}$$

and

$$\rho = 1.4, \quad p = 1.0, \quad e = 2.5, \quad u = 0, \quad v = 0, \quad \text{if} \quad y \geqslant \sqrt{3}(x - 1/6), \tag{4.14b}$$

represent a Mach 10 shock in air ($\gamma = 1.4$), which initially makes a 60 degree angle with the reflecting wall and moves into undisturbed air. Along the left boundary ($x = -0.3$) and the bottom boundary to the left of the wedge ($y = 0$, $-0.3 \leqslant x \leqslant \frac{1}{6}$), we prescribe Dirichlet boundary conditions according to (4.14): along the top boundary ($y = 1$), values are set to describe the exact motion of an undisturbed Mach 10 shock flow; along the right boundary ($x = 1.9$), all normal derivatives are set to 0; and along the wedge ($y = 0$, $\frac{1}{6} \leqslant x \leqslant 1.9$) reflecting boundary conditions are used.

This problem was used as a test problem by Woodward and Collela [38] to compare several finite difference schemes on uniform grids for the Euler Equations.

The MacCormack finite difference scheme needs artificial viscosity to "capture" the shocks of this problem. We used the artificial viscosity developed by Lapidus [28] which is velocity dependent and was used with the MacCormack scheme by Woodward and Collela [38].

The initial mesh used for this problem is 45×30, and is shown in Fig. 14 (top). We used the magnitude of the density gradient as the error indicator. From $t = 0$ to

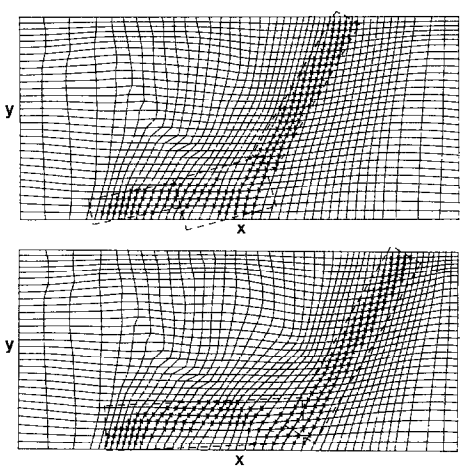

FIG. 15. Meshes of Example 4.5 at $t = 0.04$ (top), $t = 0.08$ (bottom).

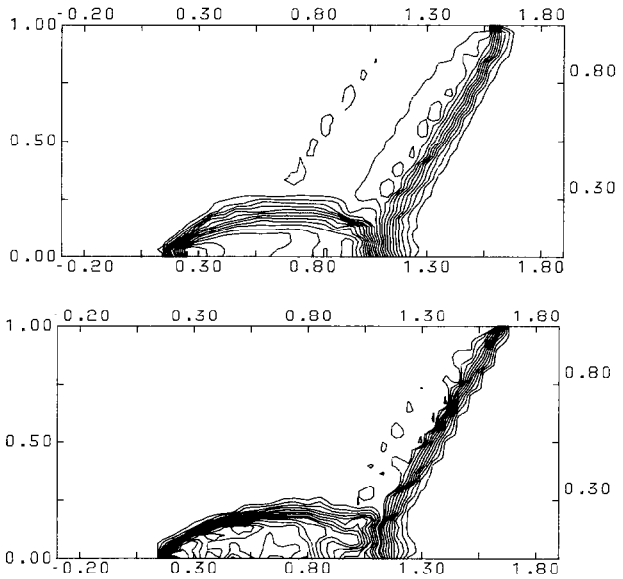

FIG. 16. Contour plots of density from the calculated solutions of Example 4.5 at $t = 0.08$ on a stationary mesh (top) and on a moving mesh (bottom) as shown in Figs. 14 and 15.

$t = 0.01$, one error cluster is formed that includes both the Mach 10 shock and the smaller reflected shock region. The resulting mesh at $t = 0.01$ is shown in Fig. 14 (center). This single error cluster is inefficient since it includes different structures of the solution with different propagation velocities. At $t = 0.02$, the clustering algorithm has recognized two different structures and has clustered the error nodes as shown in Fig. 14 (bottom). At $t = 0.04$, three different structures, the Mach 10 shock, the reflected shock, and a Mach stem region, are recognized by the clustering algorithm as shown in Fig. 15 (top). The nodes of the mesh are now able to move with the different velocities of these structures. However, by $t = 0.08$, it is evident from the mesh of Fig. 15 (bottom) that there are not enough nodes to resolve the continuing elongation of the reflected shock region. Mesh refinement is necessary to continue the effective computation of the solution for $t > 0.08$.

Contour plots of the density at $t = 0.08$ using the moving mesh and a uniform stationary mesh with the same number of nodes are compared in Fig. 16. The moving mesh reduces the numerical oscillations and provides finer resolution of the shocks and the first contact discontinuity. However, neither calculation was able to resolve the fine structures of the second Mach stem and contact discontinuity.

5. CONCLUDING REMARKS

We have described a general two-dimensional mesh moving technique that enables a mesh to follow the propagation of given error indicators. Mesh motion is

determined from the movement of clusters of nodes with significantly high error. This procedure was tested on hyperbolic problems having solutions with large gradients.

Even though mesh moving in two dimensions is difficult, we are encouraged by our initial results. The mesh moving algorithm was able to move with the wave and shock fronts of Examples 4.1 and 4.5, control the error of the rotating cone in Example 4.2, and handle the merging and separation of error regions in Example 4.4. The distortion of the mesh in Example 4.3 shows the need for static rezoning when such severe distortions occur. The elongation of the reflected shock region of Example 4.5 demonstrates the need for local mesh refinement procedures.

We are investigating ways to improve the efficiency, reliability, and robustness of the mesh moving algorithm. Possible improvements include (i) not clustering at every time-step and letting the mesh move at a constant velocity for several time-steps, (ii) efficiently testing for mesh tangling or distortion, and (iii) using a better solver for hyperbolic equations, such as the total variation diminishing schemes of Engquist and Osher [18], Osher and Chakravarthy [31], or van Leer [37]. These methods are based on the approximate solution of one-dimensional Riemann problems and they can be used, via operator splitting, with our mesh moving technique. Since the moving mesh is partially aligned with dominant dynamic phenomena, such as shock waves, we would expect these methods to perform significantly better than they do on stationary two-dimensional meshes. We also hope to show the flexibility of the mesh moving algorithm by implementing it with a finite element solver for parabolic problems. Finally, we intend to include local mesh refinement in the solution algorithm. This combination of mesh moving and refinement should enhance efficiency, accuracy, and robustness.

ACKNOWLEDGMENT

The authors thank James Hayes, Department of Mathematics, United States Military Academy, for the use of his graphics programs that are included in the system code and were used to plot many of the figures in this paper.

REFERENCES

1. S. AD JERID AND J. E. FLAHERTY, *SIAM J. Numer. Anal.*, in press.
2. S. ADJERID AND J. E. FLAHERTY, *Comput. Methods Appl. Mech. Engr.*, in press; Department of Computer Science, Tech. Rep., No. 85–21, Rensselaer Polytechnic Institute, 1985 (unpublished).
3. D. C. ARNEY, Ph.D. dissertation, Rensselaer Polytechnic Institute, 1985 (unpublished).
4. I. BABUSKA, J. CHANDRA, AND J. E. FLAHERTY, Eds., *Adaptive Computational Methods for Partial Differential Equations* (SIAM, Phaladelphia, 1983).
5. J. B. BELL AND G. R. SHUBIN, *J. Comput. Phys.* **52**, 569 (1983).
6. G. BEN-DOR AND I. I. GLASS, *J. Fluid Mech.* **92**, 459 (1979).
7. M. J. BERGER, PH.D., dissertation, Department of Computer Science, Rep. No. STAN-CS-82-924, Stanford Univ., 1982.

8. M. J. BERGER, "Data Structures for Adaptive Mesh Refinement," *Adaptive Computational Methods for Partial Differential Equations*, edited by I. Babuska, J. Chandra, and J. E. Flaherty (SIAM, Philadelphia, 1983), p. 237.
9. M. J. BERGER AND J. OLIGER, *J. Comput. Phys.* **53**, 484 (1984).
10. M. BIETERMAN AND I. BABUSKA, *Numer. Math.* **40**, 339 (1982).
11. M. BIETERMAN AND I. BABUSKA, *Numer. Math.* **40**, 373 (1982).
12. J. U. BRACKBILL AND J. S. SALTZMAN, *J. Comput. Phys.* **46**, 342 (1982).
13. J. M. COYLE, J. E. FLAHERTY, AND R. LUDWIG, *J. Comput. Phys.*, in press.
14. S. F. DAVIS AND J. E. FLAHERTY, *SIAM J. Sci. Stat. Comput.* **3**, 6 (1982).
15. D. A. DREW AND J. E. FLAHERTY, "Adaptive Finite Element Methods and the Numerical Solution Shear Band Problems," *Phase Transitions and Material Instabilities in Solids*, edited by M. Gurtin (Academic Press, Orlando, 1984), p. 37.
16. J. K. DUKOWICZ, *J. Comput. Phys.* **54**, 411 (1984).
17. H. A. DWYER, "Grid Adaption for Problems with Separation, Cell Reynolds Number, Shock-Boundary Layer Interaction, and Accuracy," AIAA Paper No. 83–0449, AIAA Twenty-First Acrospace Sciences Meeting, Reno, NV, 1983.
18. B. ENGQUIST AND S. OSHER, *Math. Comput.* **36**, 321 (1981).
19. J. E. FLAHERTY, J. M. COYLE, R. LUDWIG, AND S. F. DAVIS, "Adaptive Finite Element Methods for Parabolic Partial Differential Equations," *Adaptive Computational Methods for Partial Differential Equations*, edited by I. Babuska, J. Chandra, and J. E. Flaherty, (SIAM, Philadelphia, 1983), p. 144.
20. J. E. FLAHERTY AND P. K. MOORE, in *Proc. Conf. Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, Vol. 2, Centre for Structural Mech. and Engr. (Technical University of Lisbon, Lisbon, 1984), p. 139.
21. R. J. GELINAS, S. K. DOSS, AND K. MILLER, *J. Comput. Phys.* **40**, 202 (1981).
22. D. GANNON, Ph.D. dissertation, Department of Computer Science, Univ. of Illinois at Urbana-Champaign, 1980 (unpublished).
23. D. GOTTLIEB AND S. ORSZAG, *Numerical Analysis of Spectral Methods: Theory and Applications* (SIAM, Philadelphia, 1977).
24. A. HARTEN AND J. M. HYMAN, *J. Comput. Phys.* **50**, 235 (1983).
25. R. HINDMAN, *AIAA J.* **20**, 1359 (1982).
26. R. HINDMAN, Ph.D. dissertation, Iowa State Univ. 1980 (unpublished).
27. J. M. HYMAN, "Adaptive Moving Mesh Methods for Partial Differential Equations," Los Alamos National Laboratory Rep. LA-UR-82-3690, 1982 (unpublished).
28. A. LAPIDUS, *J. Comput. Phys.* **2**, 154 (1967).
29. K. MILLER, *SIAM J. Numer. Anal.* **18**, 1033 (1981).
30. K. MILLER AND R. N. MILLER, *SIAM J. Numer. Anal.* **18**, 1019 (1981).
31. S. OSHER AND S. CHAKRAVARTHY, *J. Comput. Phys.* **50**, 447 (1983).
32. M. RAI AND D. ANDERSON, "The Use of Adaptive Grids in Conjunction with Shock Capturing Methods," AIAA Paper 81-1012, 1981.
33. M. RAI AND D. ANDERSON, "Application of Adaptive Grids in Fluid Flow Problems with Asymptotic Solutions," AIAA Paper 81-0114, Nineteenth Aerospace Sciences Meeting, St. Louis, MO, 1981.
34. M. RAI AND D. ANDERSON, *J. Comput. Phys.* **43**, 327 (1981).
35. J. S. SALTZMAN AND J. BRACKBILL, "Applications and Generalizations of Variational Methods for Generating Adaptive Meshes," *Numerical Grid Generation*, edited by J. F. Thompson, (North–Holland, New York, 1982), p. 865.
36. J. F. THOMPSON, Ed., *Numerical Grid Generation* (North–Holland, New York, 1982).
37. B. VAN LEER, "Computational Methods for Ideal Compressible Flow," NASA Rep. No. 172180, ICASE, NASA Langley Research Center, 1983.
38. P. WOODWARD AND P. COLLELA, *J. Comput. Phys.* **54** (1984), 115.
39. O. C. ZIENKIEWICZ, D. W. KELLY, J. GAGO, AND I. BABUSKA, in *Proc. Math. Finite Elements and Applic.*, edited by J. R. Whiteman (Academic Press, London, 1982), p. 313.