INT. J. NUM. METH. HEAT FLUID FLOW, VOL. 2, 171–190 (1992)

ADAPTIVE GRID METHOD FOR UNSTEADY FLOW PROBLEMS

MICHAEL J. BOCKELIE
Computer Sciences Corporation, Hampton, VA 23666, USA
AND
PETER R. EISEMAN
Program Development Corporation, White Plains, NY 10601, USA

ABSTRACT

An adaptive grid solution method is described for computing the time accurate solution of an unsteady flow problem. The solution method consists of three parts: a grid point redistribution method; an unsteady Euler equation solver; and a temporal coupling routine that links the dynamic grid to the flow solver. The grid movement technique is a direct curve by curve method containing grid controls that generate a smooth grid that resolves the severe solution gradients and the sharp transitions in the solution gradients. By design, the temporal coupling procedure provides a grid that does not lag the solution in time. The adaptive solution method is tested by computing the unsteady inviscid solutions for a one-dimensional shock tube and a two-dimensional shock-vortex interaction. Quantitative comparisons are made between the adaptive solutions, theoretical solutions and numerical solutions computed on stationary grids. Test results demonstrate the good temporal tracking of the solution by the adaptive grid, and the ability of the adaptive method to capture an unsteady solution of comparable accuracy to that computed on a stationary grid containing significantly more grid points than used in the adaptive grid.

KEY WORDS Solution adaptive Time accurate Unsteady flow

INTRODUCTION

Solution strategies that use adaptive grids have become an increasingly important tool in computational fluid dynamics. A wide variety of methods have been proposed in recent years for solving steady and unsteady problems[1–15]. A distinguishing feature in many adaptive methods is the temporal coupling of the adaptive grid to the numerical flow solver. To compute a time accurate adaptive solution, one should use[5] an evolving grid that accurately tracks the severe solution behaviour in space and time in such a way that the solution will not escape its provided grid resolution in the course of a time step.

Solution adaptive methods can generally be classified as either static or dynamic based on the type of temporal coupling used to link the adaptive grid to the numerical flow solver. Dynamic adaptive methods continuously move the grid in space and time. These methods are typically implemented by directly computing the grid velocity or determining the new grid as part of the solution at the forward time step. This approach has the potential for producing a grid with a very tight solution tracking capability because the new grid provides resolution in the regions over which the severe solution behaviour occurs during the forward time step; that

0961–5539/92/020171–20$2.00
© 1992 Pineridge Press Ltd

Received May 1991

is, the grid does not lag the solution in time. However, dynamic methods can be difficult to implement and can lead to folded grids if the unknown grid velocity, or new grid, is not computed accurately.

In static adaptive methods the grid moves, or is adapted, only at discrete time levels. The new grid is determined separately from the solution using solution data at the current, and possibly previous, time level(s). In the flow solver, the time dependence of the grid is ignored, or treated with grid velocity terms computed with a simple backward difference in time. Static adaptive methods can make use of a broad range of reliable grid generation techniques and thus generally do not suffer from the grid singularity problems associated with the dynamic methods. However, static adaptive methods typically result in a grid that lags the solution in time because the new grid is formed using only historical solution data. Hence, these methods generally do not contain adequate temporal tracking to be applicable in time accurate solutions of unsteady problems. Accordingly, static adaptive methods are usually applied in time asymptotic solutions where only a few adaptations are needed.

In this paper we describe a temporal coupling method for linking an adaptive grid to a stationary grid flow solver that provides the temporal tracking ability of a dynamic adaptive method and the robustness and stability of a static adaptive method. The procedure treats the time integration of the solution as a series of initial value problems over short intervals in which: (1), a provisional solution is advanced on the existing grid; (2), a new grid is formed from the provisional solution data; (3), the solution data are interpolated from the old grid to the new grid; and (4), the solution is recomputed on the new grid. By successively repeating the process the solution and grid are advanced through time to the desired final time level.

The chief advantages of our procedure are accurate temporal tracking of the severe solution behaviour and flexibility. The good temporal tracking occurs because the new grid is formed from solution data forward in time and thus provides good resolution in the regions where it is needed during the forward time integration. The flexibility provided by our method comes from being able to decouple the grid generation from the flow solver being used. In particular, using the static remesh step to transfer the solution between grids avoids having to incorporate grid velocity terms into the flow solver, and thereby allows using a wide range of available flow solvers. To be cost effective, we compute the provisional solution on a coarse grid.

The temporal coupling method described here bears a philosophical similarity to one presented[8], but does not require the flow solver to have a grid velocity capability. However, if the flow solver had a grid velocity capability, our method would provide a means to efficiently and accurately compute the grid velocity. Although the temporal coupling is demonstrated only for inviscid flows computed on structured grids, the same coupling is demonstrated only for inviscid flows computed on structured grids, the same technique should work for viscous flows and for solutions computed on unstructured meshes.

In addition to the temporal coupling method a practical grid point redistribution scheme is described. The grid movement sceme is a direct curve by curve method that is a blend of techniques developed by ourselves[3,5,9,13,14] and other authors[1,6,7,10]. Here, particular emphasis is placed on a technique that improves the ability to adapt the grid to multiple solution features that merge and then separate during the course of computing the solution.

In the following sections we describe in order, the adaptive grid movement scheme, the unsteady Euler equation solver, the temporal coupling method, and last, results of numerical tests. The sections covering the grid movement scheme and the temporal coupling method contain detailed discussions of the respective techniques. The flow solver is described only briefly. The feasibility of our adaptive solution method is demonstrated on two problems. The results of a series of numerical tests for an unsteady one-dimensional shock tube are described to provide a benchmark and to demonstrate a basic ability to accurately track a solution through time. The ability to accurately perform multidimensional grid adaptation for time dependent problems is demonstrated by computing the unsteady two-dimensional inviscid flow field for a shock vortex interaction problem.

ADAPTIVE GRID

The adaptive grid method described here utilizes a surface grid lying on a *monitor surface* to identify regions where finer grid resolution is needed. The monitor surface, $\psi(x_i, y_i)$, is a smooth piecewise linear surface positioned over the physical domain. The monitor surface is typically defined as a linear combination of one or more of the variables that drives the physics of the problem. Hence, regions in which the solution has a sharp gradient, or a sharp transition between vastly different gradients, are represented in the monitor surface as, respectively, sharp gradients and sharp bends. The adaptive procedure repositions the nodes of the surface grid on the monitor surface to better resolve the geometric features of the monitor surface. At the completion of the node redistribution process the surface grid is projected down to the physical domain to yield the new grid. Repositioning the grid points to more accurately represent the physics of the governing problem should reduce the overall error in the numerical solution.

In two-dimensional problems, the nodes of the surface grid are redistributed on a curve by curve basis[5]. The order in which the curves are adapted is based on directional sweeps. The coordinate curves of the surface grid lying in one direction are operated on, followed by the same process for the curves in the opposite direction. Along each coordinate curve the mechanics of redistributing the grid points are the same as for a one-dimensional problem.

Equidistribution statement

Along each coordinate curve the grid points are repositioned to equally distribute a positive weight function over the curve. That is, the nodes are repositioned to satisfy the equidistribution statement:

$$w_{i+1/2}\Delta s_i = A\,\Delta\xi_i = \text{constant} \tag{1}$$

where $w_{i+1/2}=\frac{1}{2}(w_i+w_{i+1})$, $w_i=w(s_i)$, $\Delta s_i=s_{i+1}-s_i$, s_i is the arc length to node i along the current curve on the monitor surface, A is a constant to be determined, $\Delta\xi_i=1$, and the ξ_i are the uniformly spaced locations in computational space of the new grid points that satisfy the equidistribution condition. As can be seen in (1), the node redistribution is driven by the balancing of the weight function against the interval arc length. Notice that as the weight function increases (descreases) in value the associated interval on the monitor surface must shrink (expand).

Weight function

The basic form of w_i used in (1) is:

$$w_i = 1 + C|\kappa_{n_i}| \tag{2}$$

where C is a constant and κ_{n_i} is the normal curvature of the monitor surface at node i. The weight function in (2) will result in a surface grid in which, along each curve, the nodes have a uniform arc length spacing that is locally deviated from in regions of high normal curvature. When projected down to the physical domain, the points of the surface grid having a uniform arc length spacing will resolve the gradient regions and provide a uniform spacing in the physical domain in regions away from severe solution behaviour. The nodes of the surface grid in the regions of large κ_n will reside in the transition regions of the solution where the solution gradients are rapidly changing in value, and thereby help provide a smooth change in the grid cell size between the regions of high and low cell density (i.e., regions of high and low solution gradients, respectively). The normal curvature resolves the sharp bends in the monitor surface because it tracks the changes in the orientation of a plane tangent to the minotor surface[3]. The normal curvature is computed from:

$$\kappa_n = \hat{\mathbf{N}}\cdot\frac{d^2\mathbf{P}}{ds^2} \qquad \hat{\mathbf{N}} = \frac{\mathbf{B}}{\|\mathbf{B}\|} \qquad \mathbf{B} = \frac{\partial \mathbf{P}}{\partial \xi}\times\frac{\partial \mathbf{P}}{\partial \eta} \tag{3}$$

where $\mathbf{P}=(\mathbf{x},\psi)$ is the position vector for the nodes on the monitor surface, $\mathbf{x}=(x,y)$ is the

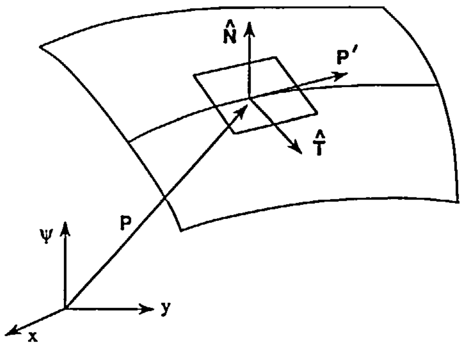

Figure 1 Orientation of vectors to compute curvature properties of curve on monitor surface

position vector of the node in the physical domain, $\hat{N}$ is the unit vector normal to both the curve and the surface tangent plane, and ξ and η are the coordinate directions of the curvilinear coordinate curves on the monitor surface (see *Figure 1*). Before assembling the weight function, a one-dimensional smoothing operation is applied to the κ_{n_i} to eliminate abrupt jumps that can occur in the computed values. The coefficient C in (2) is computed based on the prescribed percentage of the total number of grid points, f, that are to be attributed to curvature clustering[1]:

$$f = \left[\int_{s_1}^{s_n} C|\kappa_n|\,\mathrm{d}r\right] \Big/ \left[\int_{s_1}^{s_n} w(r)\,\mathrm{d}r\right] \tag{4}$$

The function for computing C includes a 'switch function' that dynamically adjusts the value of C based on the ratio of the arc length of the curve on the monitor surface to its shortest possible length; if the ratio is near one, then normal curvature clustering is not needed and $C \approx 0$[3]. Some simple guidelines for prescribing f are given in Reference 5. For problems in which the boundaries of the physical domain contain complex shapes that also need fine grid resolution (i.e., adaptation to the boundary geometry), the weight function in (2) can be modified to include the geodesic curvature of the coordinate curves on the monitor surface and thereby provide additional node concentration in regions where the boundaries contain sharp bends[3].

It should be noted that the weight function in (2) can result in a grid with very small grid cells. A grid in which the cells are too small can lead to prohibitively expensive solutions when using an explicit flow solver due to the Courant–Freidrichs–Lewy (CFL) constraint on the admissible time step. Techniques to include a minimum grid cell size control in the weight function for a curve by curve scheme have been described[5–7,14]. To maintain simplicity, we do not utilize a grid cell size control in this study because such controls typically require parameters that are based on numerical tests and thus require additional user interaction. In addition, in viscous flow problems containing thin boundary layers such a grid control may not be desirable. For problems in which the CFL constraint is too severe, it may be necessary to use a flow solver that employs an implicit time integration method when computing the solution on an adaptive grid.

Grid point redistribution algorithm

After determining the w_i, the new position vectors $\mathbf{Q}$ are computed from the old position vectors $\mathbf{P}$ using a piecewise linear transformation obtained from a series of integrations of (1) over portions of the curve[1,3,5]. Thus, on curve i the $\mathbf{Q}_{i,j}$ are computed as:

$$\mathbf{Q}_{i,j} = \mathbf{P}_{i,m} + \left[\frac{\xi_j - \xi(s_m)}{\xi(s_{m+1}) - \xi(s_m)}\right](\mathbf{P}_{i,m+1} - \mathbf{P}_{i,m}) \tag{5}$$

where the index $m=m(j)$ is the interval that brackets ξ_j, or $\xi(s_m)\leqslant\xi_j<\xi(s_{m+1})$. In the above, the $\xi(s_m)$ are the arbitrarily spaced locations in computational space of the old grid points. The value of $\xi(s_m)$ is obtained by first integrating (with a trapezoidal rule) (1) over the curve to eliminate A, then integrating only to point m and inserting boundary values for ξ:

$$\xi(s_m)=(n-1)\frac{F(s_m)}{F(s_n)} \tag{6}$$

where

$$F(s_m)=\int_{s_1}^{s_m} w(r)\,\mathrm{d}r=\sum_{k=1}^{m-1} w_{k+1/2}\Delta s_k \tag{7}$$

To obtain the positions of the new grid in the physical domain the cartesian coordinates are extracted from the new positions vectors in (5). With respect to implementing the above redistribution algorithm, note that (5)–(7) can be combined to obtain an algorithm that is mathematically equivalent but computationally more efficient (see Eiseman[3] for details).

Active passive phases

A grid smoothing operation is included in our grid movement scheme to alleviate the grid skewness and unsmooth change in node cell spacing that can occur with a curve by curve adaptive method when the adaptive data contains complex geometries or very sharp gradients. The grid smoothing is interwoven with the node redistribution by splitting the mode movement within each directional sweep into active and passive phases[5,14]. In the active phase, the nodes are redistributed along each curve in the current direction using the previously described method. In the passive phase, a smoothing operation is applied to the grid to remove any wiggles and abrupt changes in spacing created by the active phase. For a two-dimensional grid, the relaxation formula is:

$$\mathbf{R}_{i,j}=\frac{1}{2}\left[\frac{\mathbf{R}_{i-1,j}+\mathbf{R}_{i,j-1}+\mathbf{Q}_{i+1,j}+\mathbf{Q}_{i,j+1}}{4}\right]+\frac{1}{2}\mathbf{Q}_{i,j} \tag{8}$$

where **R** and **Q** are respectively the new and old position vector values at the grid point. Along the boundaries the relaxation scheme employs a ghost point for the neighbouring node that lies outside of the physical domain. The ghost point is computed by extending the curve transverse to the boundary, beyond the boundary along a trajectory with slope equal to that of the transverse curve coming into its first interior point, for a distance that is equal to the distance between the first interior point and the point where the extended transverse curve intersects the tangent vector of the boundary. At the four corner points of the grid the position vector is not relaxed. For problems in which the shape of the boundary is not to be modified while computing the solution, then an additional procedure is required to ensure that the grid point movement is tangent to the boundary. In this case, we define the boundary curve by a very fine piecewise linear representation that is never modified. After computing the new (relaxed) grid point location using (8), the (local) position vector is projected onto the fine representation of the boundary to obtain the final location. For further details on treating the boundaries see Reference 14.

The smoothing operation is typically applied 3 to 5 times to the grid in each passive phase, depending on the complexity of the problem at hand. It should be emphasized that the smoothing operation is applied only to the physical domain components of the position vector (i.e., **x**). The monitor surface values are only updated at the end of each passive phase, after all the smoothing is completed. Simultaneously updating the monitor surface values as the new grid positions are computed with (8) is too diffusive of the ψ values when multiple applications of the relaxation scheme is employed in each passive phase. The new ψ values are computed from bilinear interpolation of the ψ values at the four nodes that form the cell in the old grid that encompasses the new grid point. The required cell in the old grid is determined with a two-dimensional search procedure that is a combination of methods[9,16]. The parameters used to perform the bilinear interpolation are computed from the (x, y) coordinates of the new grid point and the four nodes that form the old grid cell using a standard Newton–Raphson method.

Vector function monitor surface

From experience we have learned that special care must be exercised in defining the monitor surface in applications where the grid is to track multiple solution features that merge during the course of computing the solution. In such cases, the simplest approach would be to treat the monitor surface as a scalar function defined as the linear combination of the different solution features to be tracked. However, this approach can result in very poor grid resolution in the region of merger due to the gradients of the tracked solution features cancelling each other. The gradient cancellation can be seen in the arc length computations. For a problem in which two solution features are being tracked, then the differential arc length for a curve on the monitor surface is

$$ds = \|d\mathbf{P}\| = (d\mathbf{x} \cdot d\mathbf{x} + d\psi^2)^{1/2} = (d\mathbf{x} \cdot d\mathbf{x} + [d\psi_1 + d\psi_2]^2)^{1/2} \tag{9}$$

That $d\psi_1$ can cancel $d\psi_2$ is easily seen in (9).

To accurately adapt the grid to multiple solution features that merge in time requires that the monitor surface be defined as a vector function rather than a scalar function. With the monitor surface defined as a vector function, the gradients of the respective components cannot cancel when the solution features merge. Thus good grid resolution is mainted in the region of merger. The vector function monitor surface is defined as an N dimension vector $\mathbf{\Psi} = (\psi_i(\mathbf{x}), \ldots, \psi_N(\mathbf{x}))$ where ψ_k is one of the N features of the solution that is to be tracked, or resolved. The ψ_k are scalar functions and are typically formed from a single variable or property of the solution. The monitor surface is positioned over the physical domain and the ψ_k are mutually orthogonal. The position vector for a point of the surface grid on the monitor surface is $\mathbf{P} = (\mathbf{x}, \psi_1(\mathbf{x}), \ldots, \psi_N(\mathbf{x}))$. Thus, for a problem being solved on a M dimensional physical domain in which N solution features are to be tracked, the scalar function monitor surface is a M dimensional surface embedded in a $M+1$ dimensional space where as the vector function monitor surface is a M dimensional surface embedded in a $M+N$ dimensional space. Repeating (9) for the vector function monitor surface yields:

$$ds = (d\mathbf{x} \cdot d\mathbf{x} + d\psi_1^2 + d\psi_2^2)^{1/2} \tag{10}$$

from which it can be seen that the contributions to the arc length of the individual solution features being tracked (i.e., $d\psi_i^2$), cannot cancel. *Figure 2* contains the scalar and vector function monitor surfaces for the case of two merging disturbances for a one-dimensional problem. For this example the scalar monitor surface is a one-dimensional surface in a two-dimensional space (i.e., planar curve) and the vector function monitor surface is a one-dimensional surface in a three-dimensional space (i.e., curve in space). Here, the distinction between the vector and scalar monitor surface is easily visualized. In addition, the improved grid resolution in the region of merger of the two solution features can be readily seen in the vector function monitor surface plot (*Figure 2d*). A similar comparison of scalar and vector function monitor surfaces for two-dimensional problems are contained in References 9 and 13.

The process for adapting a grid to a vector function monitor surface is essentially the same as for a scalar function monitor surface. To ensure that the curvature properties used in the weight function are uniquely defined, the surface grid is projected onto each component of the vector monitor surface, ψ_k, to compute the normal curvature for the respective component. Thus at each node there will be a normal curvature value computed for each of the ψ_k components of the monitor surface. The normal curvature at node i for the kth component of the monitor surface, $\kappa_{n_i}^{(k)}$, is computed using (3) where the position vector to be used is $\tilde{\mathbf{P}} = (\mathbf{x}, \psi_k(\mathbf{x}))$ and the associated arc length $\tilde{s}$ is computed along the curve projected onto component ψ_k, $\tilde{s} = \tilde{s}(\tilde{\mathbf{P}})$. The

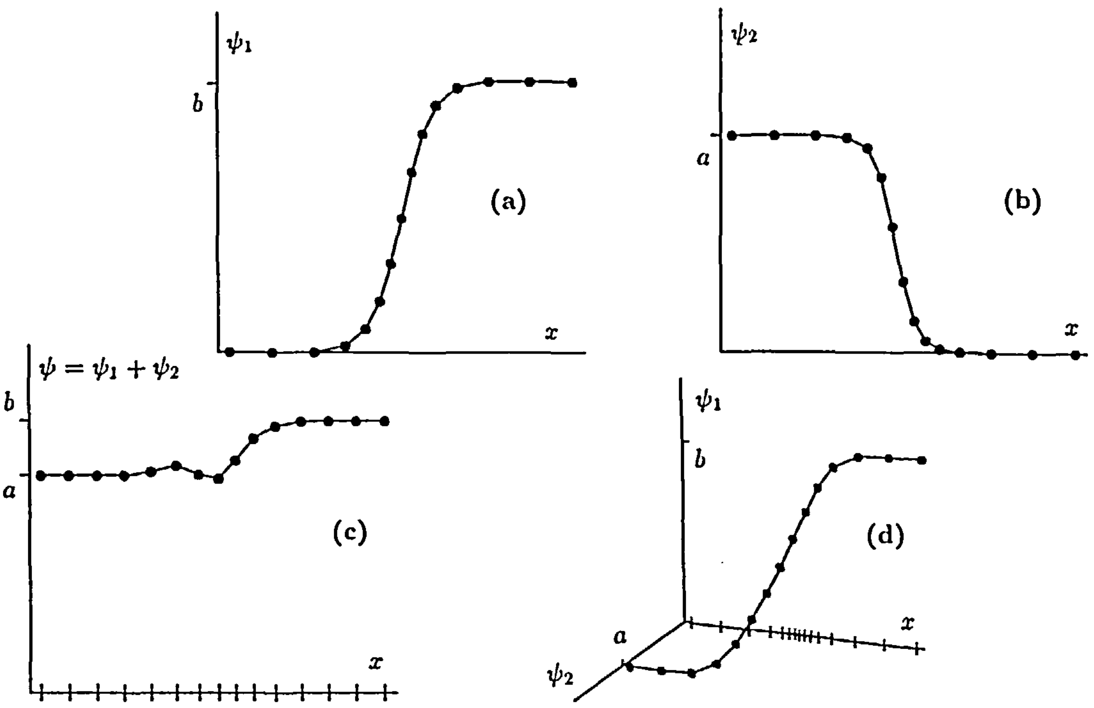

Figure 2 Comparison of scalar and vector monitor surface for 1D problem. Illustrated are two solution features to be tracked (a, b), the scalar (c) and vector (d) monitor surface. Resulting grid shown on x axis

weight function used in the equidistribution statement is:

$$w_i = 1 + \sum_{k=1}^{N} C_k |\kappa_{n_i}^{(k)}| \tag{11}$$

where C_k is the curvature coefficient for ψ_k. Before assembling the weight function a one-dimensional smoothing operation is applied to the $\kappa_{n_i}^{(k)}$ values. The dynamic curvature coefficient for each component of Ψ, C_k, is computed in an analogous manner to that for a scalar monitor surface, where the percentage of the grid points to be attributed to curvature attraction for the kth solution feature is f_k. Further details on treating the monitor surface as a vector function are contained in Reference 9.

Summary of grid movement scheme

Before starting the grid movement, the relaxation formula in (8) is applied three times to the $\psi(x)$ values (not the x positions) to ensure that the monitor surface is smooth. Along the boundaries, the ψ value of the ghost point that lies outside of the physical domain is set equal to the ψ value at the first interior point on the same transverse curve (i.e., reflection boundary condition). For a vector monitor surface, the ψ_k are smoothed individually.

After smoothing the monitor surface, the new grid is generated with two iteration cycles of the following:

(1) perform the *active* phase for each curve in the first direction:

(a) compute $\bar{s}_i$, s_i, κ_{n_i}, etc. at each node on the current curve;
(b) form w_i for each node;

(c) compute the $\mathbf{Q}_{i,j}$ locations that satisfy the equidistribution statement by redistributing the points on the monitor surface along the current curve;

(2) perform the *passive* phase by applying the smoothing operator to the grid to yield $\mathbf{R}_{i,j}$;
(3) repeat steps (1) and (2) for the opposite direction.

Numerical tests have demonstrated that the grid typically settles in to a final configuration within a few iterations. Thus the grid movement is stopped after only two global iteration cycles, rather than iterating until the grid has converged. The slightly improved grid resolution that can be obtained with a more precise placement of the grid points typically does not offset the additional expense of computing such a node placement. The CPU time for the above grid generation algorithm is about 120 μsec per grid point per global iteration cycle on a CRAY2.

THE EULER EQUATION SOLVER

The flow equation solver used in this study is an 'off the shelf' finite volume, shock capturing code for solving the unsteady two dimensional Euler equations for idealized compressible gas flow[9]. The solution is computed using a characteristic based method that captures crisp (almost) monotone shock profiles ($O(\Delta x^2)$). The solution is integrated in a time accurate manner with a classical two step modified Euler explicit time integration scheme that is nominally $O(\Delta t^2)$ for a linear problem. The flow solver is designed for use on a stationary (i.e., time independent) grid and has not been altered to include grid velocity terms. The CPU time for the flow solver is about 80 μsec per grid point per time step on a CRAY2.

TEMPORAL COUPLING

Our temporal coupling method treats the time integration of the solution from $T=0$ to $T=T_f$ as a series of initial value problems over *short* time intervals. Conceptually, the steps required to perform a single cycle of the process can be grouped into four stages. In the first stage, a provisional solution is advanced forward in time for a time interval τ. Next, the new adaptive grid for this time interval is generated using the data collected from the provisional solution stage. In the third stage the initial solution is transferred from the old grid to the new grid (i.e., a static remesh). In the last stage, the solution is recomputed on the new grid over the time interval τ. The process is repeated until the solution has been advanced to the desired final time level.

Each cycle of our temporal coupling algorithm begins at a time level T with an existing adaptive grid ($\mathbf{X}$) and solution ($\mathbf{q}(\mathbf{X}, T)$), and ends with a new adaptive grid and solution at a time level $T+\tau$. In keeping with our approach of treating the time integration as a series of initial value problems, we refer to $\mathbf{X}$ and $\mathbf{q}(\mathbf{X}, T)$ as the initial condition for the cycle starting at time T. Note that in general $\mathbf{X}$ and $\mathbf{q}(\mathbf{X}, T)$ are not the same as the initial condition of the physical problem that is specified at $T=0$. In addition, the new adaptive grid and solution determined at time level $T+\tau$ will become the initial condition for the next cycle of the algorithm, in which the grid and solution are advanced in time for another short time interval.

To be cost effective, the solution in the provisonal stage of the temporal coupling algorithm is computed on a coarse grid defined from the existing adaptive grid ($\mathbf{X}$). For many applications, an adequate estimate of the adaptive data needed to generate the new grid can be obtained in this manner. The larger grid cell size of the coarse grid allows using a larger time step based on the CFL constraint. The combination of the fewer grid points and larger time step makes the CPU time incurred in computing the provisional solution small in comparison to that used to recompute the solution on the full grid. In the two-dimensional shock vortex solutions described below, the ratio of the CPU time of the provisional solution step (STEP 1–6) to that of the full grid solution stage (STEP 9) is about 1:7. In three-dimensional problems, this ratio would be even smaller.

In the following discussion, the temporal coupling algorithm is divided into ten steps in order to provide a precise description of a single cycle of our algorithm.

STEP 1: choose a time interval τ

In some problems it is possible to specify the value of τ. However, it is usually simpler to specify the number of times steps, p, to be used in the forward time integration of the provisional solution. Thus $\tau=\sum_{k=1}^{p}\Delta t_k$ is the admissible time step determined by the flow solver for computing a time accurate solution. In the present study we (arbitrarily) set $p=10$. Although this may seem like a large value for p, the CFL constraint of the explicit time integration in the Euler equation solver results in the time interval τ being quite small. In the case that an implicit flow solver were being used, we would set the value of p to be much smaller.

STEP 2: obtain a coarse grid from the full grid

The coarse grid, XC, is obtained by deleting every second coordinate curve in each direction in X. Note that the metrics, jacobians, etc. of XC must be computed.

STEP 3: transfer initial condition to the coarse grid

The initial condition on the coarse grid, $\mathbf{q}(\mathrm{XC}, T)$, is computed from $\mathbf{q}(\mathrm{X}, T)$ using local bilinear interpolation. Note that the (i,j) locations of the points in X used to interpolate the values onto XC are known *a priori* due to the manner in which XC is defined. The localized coordinates used to perform the bilinear interpolation can be computed very efficiently using a standard Newton–Raphson method. (This interpolation step is required because in our flow solver the solution is located at the center of the grid cell. For flow solvers in which the solution and grid vertex point coincide, the solution on the coarse grid could be extracted directly from the full grid solution without interpolation.)

STEP 4: compute provisional solution over τ on the coarse grid

A provisional solution is computed up to $T+\tau$ by integrating $\mathbf{q}(\mathrm{XC}, T)$ forward in a time accurate manner. After each time step a monitor surface, ψ_i, is computed and stored in a running sum. A ψ_i is also formed from $\mathbf{q}(\mathrm{XC}, T)$ to enhance the temporal smoothness of the grid.

STEP 5: form a monitor surface over τ

The $p+1$ monitor surfaces computed in STEP 4 are averaged to yield a composite monitor surface on the coarse grid, $\psi^*(\mathrm{XC})$. Thus, $\psi^*=(1/(p+1))\sum_{i=1}^{p+1}\psi_i$.

STEP 6: transfer monitor surface to the full grid

The monitor surface on the full grid, $\psi^*(\mathrm{X})$, is computed from $\psi^*(\mathrm{XC})$ using local bilinear interpolation. Again, the (i,j) locations of the points in XC used to interpolate the values onto X are known *a priori* due to the manner in which XC is defined.

STEP 7: adapt the grid

The new grid, X*, is generated using two cycles of the curve by curve scheme described above. After generating X*, its metrics, etc. must also be computed.

STEP 8: transfer solution data to the new grid

A conservative rezoning method is used to interpolate the old initial condition $\mathbf{q}(\mathrm{X}, T)$ onto X* to determine the new initial condition $\mathbf{q}(\mathrm{X}^*, T)$. By using a conservative rezoning method the conservation properties of the solution will still be intact after the interpolation onto the new grid. Letting $\mathbf{q}$ represent the solution vector, then in each cell of the new grid X*, $\mathbf{q}(\mathrm{X}^*, T)$

is computed from $\mathbf{q}(\mathbf{X}, T)$ as:

$$\mathbf{q}^{\text{new}} = \frac{1}{A^{\text{new}}} \sum_m \iint_{A_m^0} \{\mathbf{q}_m^0 + \langle \nabla \mathbf{q}_m^0 \rangle \cdot \Delta \mathbf{r}\} \, dS \tag{12}$$

where $\mathbf{q}^{\text{new}}$ is $\mathbf{q}$ at the centre of the new grid cell, A^{new} is the area of the new grid cell, A_m^0 is the fractional area of the old grid cell m that lies in the new grid cell, $\mathbf{q}_m^0$ is $\mathbf{q}$ at the centre of the old grid cell m, $\Delta\mathbf{r}$ is the local position vector within the old grid cell m, and $\langle \nabla \mathbf{q}_m^0 \rangle$ is a volume averaged estimate of the gradient of $\mathbf{q}$ within cell m. The gradient contains a 'Van Leer' like limiter function so that the interpolation on to the new grid does not induce new extrema into the solution. The gradient terms are computed as described in Reference 17 and the summation of the old grid cell contributions to each new grid cell is computed[18]. The interpolation is second order accurate in regions where the flow field is smooth, but is only first order accurate where there is an abrupt change in the solution (e.g., a shock wave). If the flow solver being used had a grid velocity capability, then the solution interpolation performed in this step would be replaced with computing the grid velocity terms.

STEP 9: recompute solution over τ on the new grid

The new initial condition on the new grid, $\mathbf{q}(\mathbf{X}^*, T)$, is integrated forward in time to yield $\mathbf{q}(\mathbf{X}^*, T+\tau')$ where $\tau' = 0.95\,\tau$ to ensure the solution does not outrun its provided resolution.

STEP 10: go to STEP 1

The solution is marched forward through time to the desired final time level T_f by successively repeating the procedure.

To achieve additional temporal accuracy in the solution, a sub-iteration cycle can be performed on STEP 1–7 before advancing the solution to the next time level. The additional accuracy results because with each pass through the iteration cycle, the grid more accurately represents the evolving solution and hence a more accurate solution can be computed on it. However, the increased accuracy occurs at the expense of additional computational effort. Performing such an iteration cycle results[15] in a 'strong temporal coupling'. In fact, its limit represents an implicit coupling of the grid and the solution. In the present study we do not use the sub-iteration cycle. In previous numerical tests with the shock tube problem described below, the slight improvement in solution accuracy achieved with the sub-iteration cycle did not justify the additional expense[14].

It should be emphasized that if a high degree of accuracy is required in the solution, then the static remesh in STEP 8 should be performed with the second order conservative rezoning method. In previous studies, we have tested using a bilinear interpolation method and a first order conservative rezoning method for performing the static remesh[14]. Bilinear interpolation is an attractive interpolation method because it does not induce oscillations into the solution, is simple to implement in a computationally efficient manner for two dimensional problems, and its extension to three-dimensional problems should be straight forward. However, bilinear interpolation consistently resulted in an $O(1\%)$ error in the location of the captured shock waves. The first order conservative rezoning method correctly captured the shock locations, but was rather diffusive in regions away from the adaptive resolution. The second order conservative rezoning method overcomes the solution accuracy limitations observed with the simpler interpolation methods and thus is the method used in this study. However, the improved solution accuracy requires additional human effort. A conservative rezoning method is somewhat difficult to implement because the intersections of cell faces must be computed; its extension to three-dimensional problems could be complicated. In addition, the computer implementation of the rezoning algorithm must be very efficient because in the adaptive context the solution data is transferred between successive grids a large number of times. For three-dimensional problems, and two-dimensional problems containing complex geometries, it may be advisable to use either a higher order monotone interpolation method[19], or to employ a flow solver with a grid velocity capability.

One last note pertaining to the temporal coupling algorithm is how to treat the case $T=0$. This case requires special attention because the given grid may not properly resolve the given initial condition of the problem. To ensure we obtain a 'gentle start' on the problem, we generate an initial grid that is adapted to the given initial condition and (in some cases) reduce the value of τ, or p, on the first few cycles of the 10 step routine. The initial grid for the gentle start is generated using the sub-iteration cycle on STEP 1–7 described above for increasing the temporal accuracy of the grid. If a gentle start up cycle is not used then the first adaptation of the grid can be equivalent to an impulsive start and lead to severe grid generation difficulties.

Unsteady shock tube

A series of numerical tests have been performed for the unsteady shock tube problem described by Sod[20]. The key features of the solution are an expansion wave that propagates to the left and a shock wave, followed by a contact discontinuity, that propagates to the right. Results are presented for solutions computed on uniform stationary grids (100×2, 200×2, 400×2 cells) and on adaptive grids (100×2 cells). The adaptive grids are generated using the previously described two-dimensional adaptive method by applying the grid movement in only the axial direction. The monitor surface for the adaptive solutions is formed from the internal energy per unit mass because this variable undergoes a large change in value at all three waves in the solution. The time interval of the provisional solution stage (τ) is prescribed to control how often the grid is regenerated.

Illustrated in *Figures 3* and *4* are the internal energy per unit mass for solutions computed on an uniform stationary grid (100 cells) and an adaptive grid in which the grid is generated using 20 adaptations (i.e., $\tau=0.05\ T_f$ in the temporal coupling routine). Comparing *Figures 3* and *4*, the adaptive solution has much closer agreement with the theoretical solution. *Figure 5* illustrates the change in the grid positions through time. *Figures 4* and *5* demonstrate graphically the accurate temporal and spatial tracking of the three waves in the solution provided by the adaptive solution method. In *Figures 4* and *5*, the concentration of grid points at the three waves occur over relatively broad regions because in our temporal coupling routine the solution is computed on the new grid for several time steps. For this test case, the full grid solution is computed for about 25 times steps on the new grid.

Table 1 and *Figure 6* contain quantitative comparisons of some solution properties for the shock tube studies. The shock wave propagation speed contained in *Table 1* is computed from a least squares fit of the shock front location as a function of time. As can be seen from *Table 1*, the adaptive solution correctly captures the time dependent shock location, whereas in the

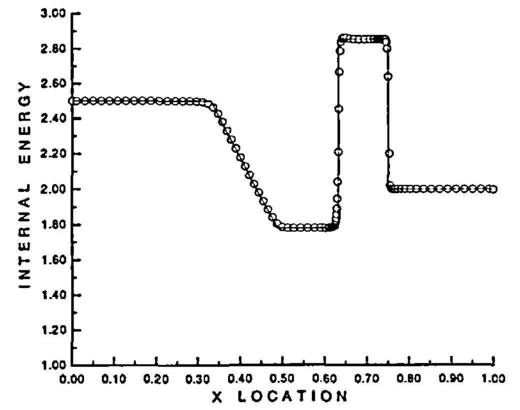

Figure 3 Uniform grid shock tube solution (100 cells)

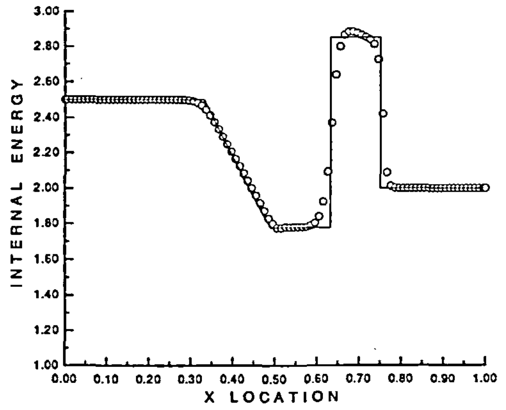

Figure 4 Adaptive grid shock tube solution (20 adaptations)

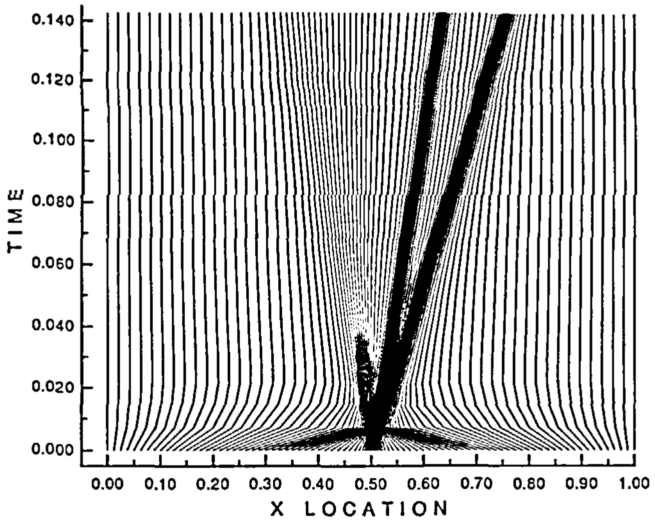

Figure 5 Time history of adaptive grid positions in shock tube solution

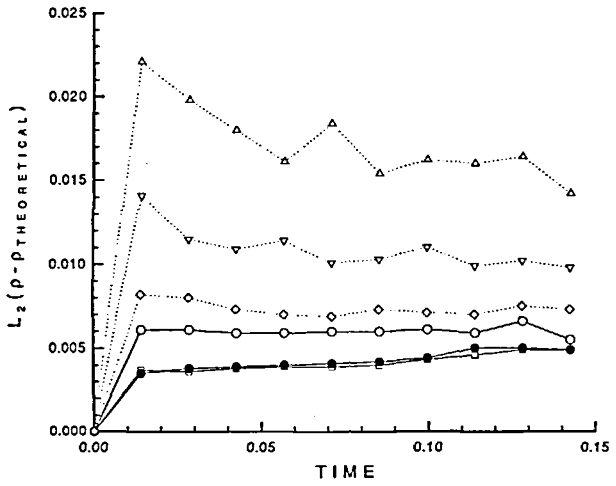

Figure 6 Time history of L_2 $(\rho-\rho_{\text{theoretical}})$ for shock tube solutions. Plotted are adaptive solutions computed with 20 (○), 100 (□), 200 (●) adaptations and stationary grid solutions computed on uniform grids with 100 (△), 200 (▽), 400 (◇) cells

stationary grid solutions the shock is correctly captured only on the 400cell grid. Illustrated in *Figure 6* is the time history of the error in the numerical solutions listed in *Table 1*. The solution error is computed (at eleven time levels) as the L_2 norm of the difference in the density for the theoretical and numerical solutions. From *Figure 6*, it can be seen that the adaptive solutions capture a more accurate solution than the stationary grid solutions. Also note in *Figure 6* that adapting the grid more frequently generally results in a more accurate solution, due to the tighter tracking of the solution by the adaptive grid. However, adapting the grid more frequently also increases the diffusion of the solution that occurs in the static remesh step; if the grid is adapted

Table 1 Comparison of shock tube solution properties

Solution	Grid cells	No. of grid adaptations	Shock speed†, M_s	M_s error* (%)
Exact	n/a	n/a	1.7522	0.00
Stationary grid	100	n/a	1.7636	0.65
	200	n/a	1.7539	0.10
	400	n/a	1.7520	−0.01
Adaptive grid	100	20	1.7520	−0.01
	100	100	1.7521	−0.01
	100	200	1.7521	−0.01

† Propagation speed computed from least squares fit of shock wave location as a function of time.
* Error $=(M_s - M_s|_{\text{theoretical}})/M_s|_{\text{theoretical}}$.

too many times the diffusion can offset the benefit of the improved solution tracking, as demonstrated by the adaptive solutions for the cases of 100 and 200 adaptations having essentially the same error.

Altogether, the shock tube results demonstrate the basic ability of our adaptive method to accurately track a solution through time using a grid with a modest number of grid points.

Shock vortex studies

A shock vortex interaction problem provides a good test of a computational method because the solution exhibits severe gradients, the positions of which change with time. From a physical viewpoint, the interaction provides an idealized model for a wide variety of problems[4,21,22].

The shock vortex problem modelled here consists of an initially planar shock wave marching toward, and eventually over, a simple solid core vortex (*Figure 7*). The flow field is assumed to lie within a solid walled channel. The initial flow field consists of a planar shock located at $x=0$ and a simple solid core vortex rotating counter clockwise centred at $(x, y)=(0.65, 0.0)$. To the left of the shock wave is the uniform (subsonic) flow field that would follow behind a propagating shock wave with a relative Mach number of $M_s=1.3$, if there were no upstream disturbance. Ahead of the shock, the initial velocity field of the vortex is $\mathbf{V}=(0, u_\theta(r))$ where $u_\theta(r)=u_0 r/r_c$ for $0 \leqslant r \leqslant r_c$, $u_\theta(r)=u_0 r_c/r$ for $r \geqslant r_c$ and $u_0=0.3$ is the maximum velocity in the vortex flow field, r is measured relative to the centre of the vortex, and $r_c=0.1$ is the size of the vortex core. The density and pressure of the initial flow field of the vortex are determined by assuming the vortex is a constant enthalpy flow[4,14]. Along the channel walls the flow is assumed to be tangent to the boundary. At the inlet and outlet the flow variables are extrapolated from the neighbouring cell in the interior.

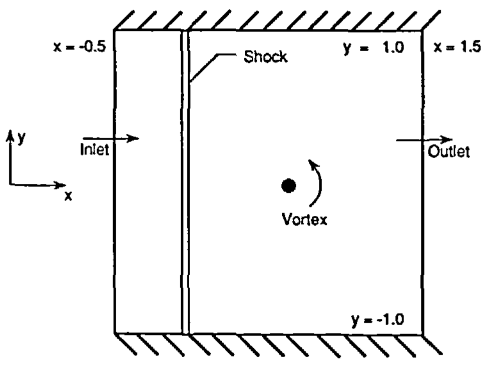

Figure 7 Shock problem definition

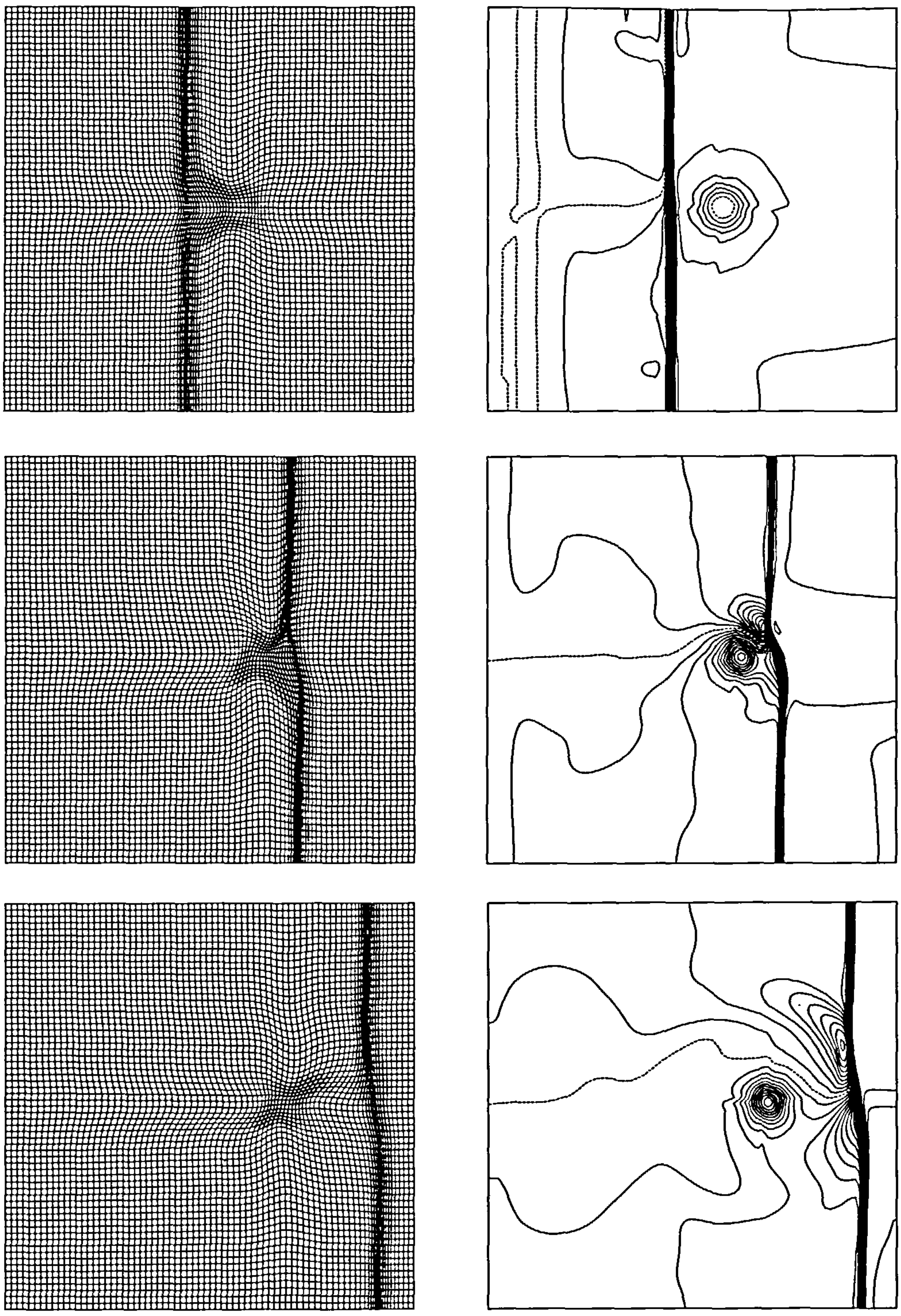

Figure 8 Adaptive grid shock vortex solution at (a) $t=0.30$, (b) $t=0.70$, (c) $t=1.00$

Illustrated in *Figures 8a–8c* are the grid and normalized pressure field ($\tilde{p}=p/p_{-\infty}$) for solutions computed on an 80×64 cell grid (80 cells in the axial direction) adapted to both the shock wave and the vortex. In the contour plots, $\Delta\tilde{p}=0.01$ with $\tilde{p}\in[0.50, 1.10]$, and the contours at $\tilde{p}=0.5$ and 1.00 plotted as broken lines.

From the pressure plots in *Figure 8* the key features of a shock vortex interaction can be discerned. Before the shock reaches the vortex core ($t=0.30$), the shock wave is still essentially planar and the pressure field behind the shock wave is slightly anti-symmetric, with the pressure being slightly higher in the upper half of the channel. After the shock has passed over the vortex core ($t=0.70$), the shock front is curved and the pressure contours behind the shock contain regions of sharp curvature indicating the presence of an acoustic wave. As the shock has moved further down the channel ($t=1.00$), the curvature of the shock front becomes less pronounced because the upstream disturbance of the vortex flow field decrease. The vortex, which can be identified by the concentric pressure contours, has been convected down stream by the flow field following the shock wave and there is a noticeable gap between the shock wave and the vortex due to the shock propagating at a faster speed than the vortex. The acoustic wave is now more visible, being roughly cylindrical in shape and approximately centred on the drifting vortex. In addition, it can be seen that the acoustic wave is butting into the shock wave just above and below the channel centreline.

In the grid plots, the concentrated band of grid lines that runs across the channel is due to the adaptation to the shock wave. The concentration of grid points near the channel centreline is due to the attraction to the vortex. The smooth nature of the grid in the high resolution areas can be seen in *Figure 9* which contains an enlargement of the region near the vortex after the interaction has occurred ($t=0.70$).

The adaptive grids are generated using a vector function monitor surface formed from the density field and the circulation about each grid point[9,13]. The density field clusters the grid points to the shock front because the density field is approximately constant everywhere except across the shock wave where a large jump in value occurs. The circulation is defined as $\Gamma=\int \mathbf{V}\cdot d\mathbf{l}$ and provides a means of concentrating grid points to the vortex core because Γ is approximately zero everywhere except within the vortex core.

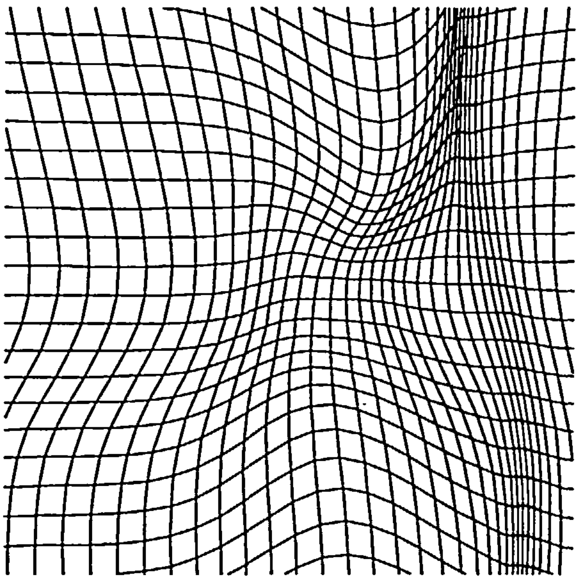
Figure 9 Enlarged view of grid near vortex ($t=0.70$)

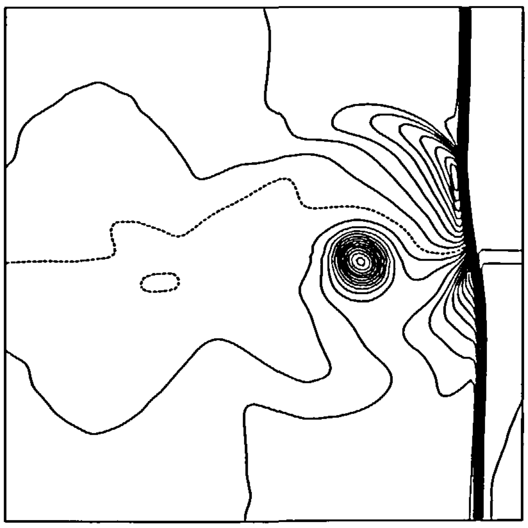
Figure 10 Shock vortex solution ($t=1.00$) using 300×120 cell stationary grid

To provide a comparison for the adaptive results, solutions have been computed on a stationary grid with fine resolution (300 × 120) and very fine resolution (500 × 120). Overall, there is a good agreement between the adaptive and stationary grid solutions. For brevity, *Figures 10* and *11* illustrate the respective stationary grid solutions only at the last time level.

Comparing *Figures 8*, *10* and *Figure 11*, it can be seen that the adaptive grid correctly captures the time dependent shape and location of the shock wave and the vortex. The adaptive solution captures a crisp shock wave that has bending and thickness that is comparable to that of the fine stationary grid solution even though the adaptive solution uses significantly fewer grids points.

The temporal tracking of the adaptive grid is further illustrated in *Figures 12*, *13* and *14*. In these Figures, the normalized pressure is plotted along the bottom wall, the channel centreline, and along the top wall at eleven time levels from $t=0$ to $t=1.00$ ($\Delta t=0.10$). The 'o' on the pressure plots indicates the location of the cell centres and the pressure profiles corresponding to the solutions illustrated in *Figure 8* are highlighted. From these plots the temporal evolution of the solution can be discerned. In addition, these plots provide a visual demonstration of the adaptive grid tracking the time dependent location of the severe solution behaviour.

The numerical shock width and relative shock location along the top and bottom channel walls at eleven time levels are plotted in *Figures 15* and *16*. To facilitate making quantitative comparisons, in the following we have assumed that the very fine grid solution is the 'exact' solution to the problem. In *Figure 15* the shock width is normalized by the shock width of the very find stationary grid solution. Plotted in *Figure 16* is the difference in the shock location relative to that of the very find stationary grid solution, normalized by the channel width. Comparing *Figures 15* and *16* again demonstrates the good agreement between the adaptive and stationary grid solutions. In *Figure 15* it can be seen that during the time that the shock is passing over the vortex core ($0.40 \leqslant t \leqslant 0.65$), the adaptive grid produces a thiner numerical shock than occurs on the fine stationary grid. In *Figure 16*, note that the time dependent shock location agrees with the shock location of the 500 × 120 cell solution to within ±4/100 of 1%.

Illustrated in *Figure 17* is the time history of the relative error in the density fields. The relative error is computed at specified time levels as the L_2 norm of the difference in the density of the

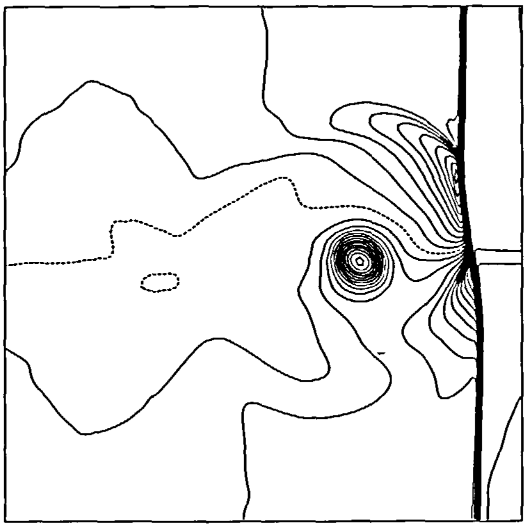

Figure 11 Shock vortex solution ($t=1.00$) using 500 × 120 cell stationary grid

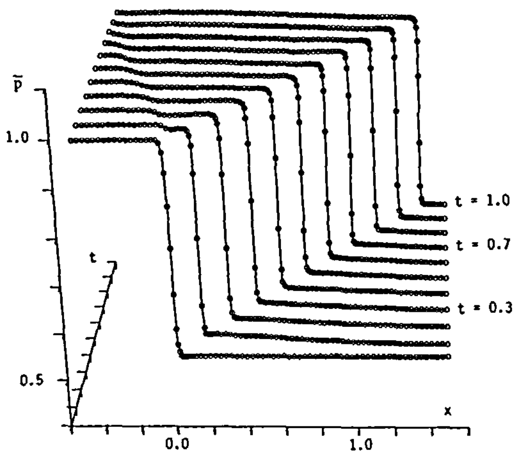

Figure 12 Time history of normalized pressure along bottom wall

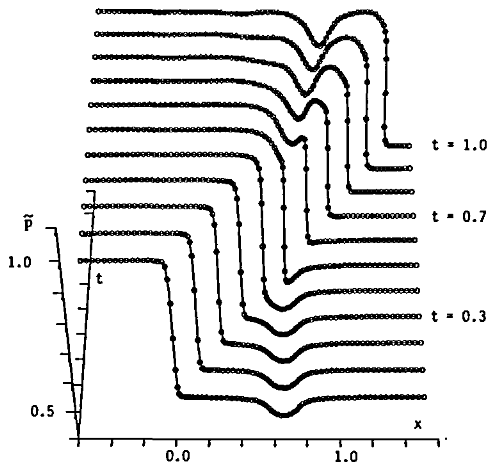

Figure 13 Time history of normalized pressure along *channel centreline. Cell centres approximated as average* location of cell centre immediately above and below channel centreline. Plot is viewed from above

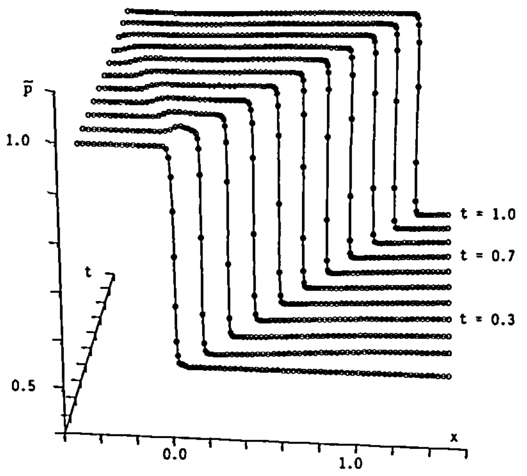

Figure 14 Time history of normalized pressure along top wall

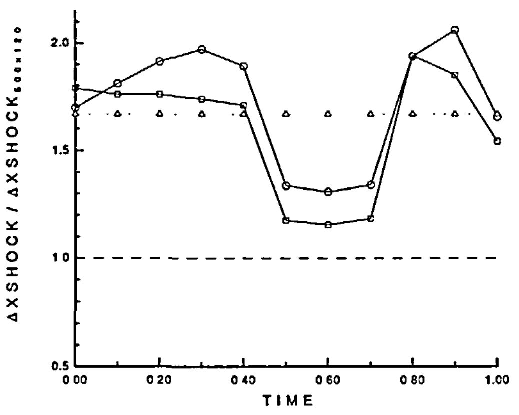

Figure 15 Numerical shock width normalized by shock width in 500 × 120 cells solution. Shock width is constant in the fine (△) and very fine (—) stationary grid solutions. Shock width varies in adaptive solution and is plotted only along the bottom (O) and top (□) wall

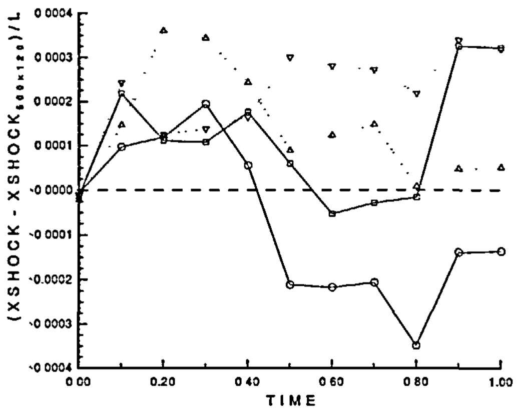

Figure 16 Shock location relative to shock location in 500 × 120 cells solution. Values normalized by channel width ($L=2$). Results plotted along, respectively, bottom and top wall for 300 × 120 cell solution (△, ▽) and adaptive solution (O, □)

very fine stationary grid and, respectively, the adaptive and fine stationary grid solutions. The L_2 values are determined using bilinear interpolation to transfer the density of the 500 × 120 cell solution onto the 80 × 64 adaptive grid and the 300 × 120 stationary grid. Comparing *Figure 15* and *17*, it can be seen that for the time interval when the shock passes over the vortex, the thinner shock captured by the adaptive solution results in a more accurate solution than occurs for the fine stationary grid solution, and approaches the accuracy achieved on the very fine grid. For the time intervals before and after the interaction, the error in the adaptive solution is slightly greater than that of the fine stationary grid. Over the full time period of the simulation, the error in the adaptive solution is roughly comparable to that of the fine grid solution.

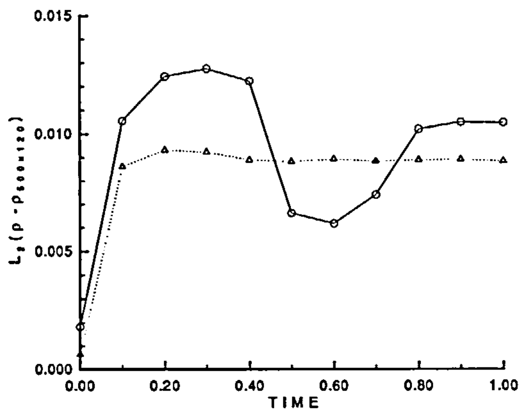

Figure 17 Time history of $L_2(\rho-\rho_{500\times120})$ for 30×120 cell solution (△) and adaptive solution (○)

Table 2 Comparison of shock vortex solution properties

Solution	Grid cells†	x_s^b*	x_s^t*	Time steps†	CPU†
500×120 Stationary	1.00	1.3288	1.2761	1.00	1.00
300×120 Stationary	0.60	1.3289	1.2767	0.66	0.40
80×64 Adaptive	0.09	1.3285	1.2767	0.79	0.18

* Shock wave location on bottom (b) and top (t) channel wall at $t=1.00$
† Number of grid cells, time steps, and CPU time (CPU) normalized by corresponding values for 500×120 cell solution

Listed in *Table 2* are the number of time steps and CPU time used to compute the adaptive and stationary grid solutions. In the temporal coupling routine we have used $p=10$, resulting in 42 sequences of the 10 step procedure. On average, the full grid solution is integrated for about 19 time steps on each adaptive grid. The number of time steps required for the adaptive solution is slightly greater than that of the fine grid solution and slightly smaller than that of the very fine grid solution; the adaptive solution requires such a large number of time steps due to the small grid cells created by the adaptive grid at the shock wave and the CFL constraint in the explicit time integration scheme. At present we achieve a little over 50% savings in CPU time as compared to the fine stationary grid solution which, over the full time period of the simulation, has a L_2 error that is roughly comparable to that of the adaptive solution.

In the adaptive solution, the percentage of the total CPU time spent in the provisional solution (STEP 1–6), grid generation (STEP 7), conservative rezoning (STEP 8) and full grid solution (STEP 9) stages are 6%, 7%, 46%, and 41% respectively. These percentages highlight two items. First, and most important, it can be seen that the CPU time spent in the (coarse grid) provisional solution stage and the grid generation stage is relatively small in comparison to the CPU time used in the full grid solution stage. Second, the conservature rezoning algorithm requires a relatively large portion of the solution time. The large amount of time consumed by the conservative rezoning method is due in part to the large number of computations that it requires and, to a lesser extent, an inefficient computer implementation of the rezoning algorithm. To provide some perspective, if the adaptive solution were computed using bilinear interpolation instead of conservative rezoning for the static remesh in STEP 8, then the relative CPU time to compute the solution would be about 0.10 and the percentage of CPU time spent in the provisional, grid generation, solution interpolation, and full grid solution stages would be 11%, 12%, 4%, and 73% respectively. However, as noted in the section describing the temporal coupling method, using bilinear interpolation for the static remesh step results in a noticeable loss in solution accuracy.

To summarize, the shock vortex tests demonstrate the ability of our adaptive solution method to capture an unsteady solution that is comparable in accuracy to a solution computed on a stationary grid containing significantly more grid points. In addition, these tests demonstrate the ability of our adaptive grid generator to track the time dependent location and shape of multiple solution features.

CONCLUSIONS

An adaptive grid method has been described that can be used to compute time accurate solutions of unsteady two-dimensional flow problems. A technique that treats the adaptive data as a vector function rather than a scalar function has been presented that improves the ability to adapt the grid to multiple solution features. A temporal coupling routine has been described for linking the adaptive grid and a stationary grid flow solver in such a way that the grid does not lag the solution in time. Using a coarse grid during the prediction stage has been demonstrated to provide sufficient information to generate the new grid.

The temporal coupling utilizes a static remesh to avoid having to modify the flow solver to include a grid velocity capability. The static remesh is performed with a second order conservative rezoning method in order to correctly capture the time dependent location of shock waves. The feasibility of the method has been demonstrated through numerical tests for a one-dimensional shock tube and a two-dimensional shock vortex problem.

Comparisons of the adaptive solutions with theoretical solutions and numerical solutions computed on stationary grids have demonstrated (a) the good temporal tracking of the solution by the adaptive grid, and (b) that our adaptive method can capture an unsteady solution to a fairly high degree of accuracy while using a grid containing only a modest number of points.

ACKNOWLEDGEMENTS

Portions of this project were performed while M.B. held a National Research Council Research Associateship at the NASA Langley Research Center. This project was partially supported under NASA and Air Force Grants NAG1-427 and AFOSR-86-0307. The computational work for this project was performed on the Numerical Aerodynamic Simulator (NAS) at the NASA Ames Research Center. The comments and support of R. E. Smith, NASA Langley Research Center, are gratefully acknowledged.

REFERENCES

1 Dwyer, H. A. Grid adaption for problems in fluid dynamics, *AIAA J.*, 22, 1705–1712 (1984)

2 Thompson, J. F. Grid generation techniques in computational fluid dynamics, *AIAA J.*, 22, 1505–1523 (1984)

3 Eiseman, P. R. Alternating direction grid generation, *AIAA J.*, 23, 551–560 (1985)

4 Srinivasan, G. R., McCroskey, W. J. and Baeder, J. D. Aerodynamics of two dimensional blade vortex interaction, *AIAA J.*, 24, 1569–1576 (1986)

5 Eiseman, P. R. Adaptive grid generation, *CMAME*, 64, 321–376 (1987)

6 Nakahashi, K. and Diewert, G. S. A self-adaptive grid method with application to airfoil flow, *AIAA J.*, 25, 513–520 (1987)

7 Abolhassani, J. S., Smith, R. E. and Tiwari, S. N. Grid adaptation for hypersonic flow, *AIAA Paper 87-1169CP*, (1987)

8 Blom, J. G., Sanz-Serna, J. M. and Verwer, J. G. On simple moving grid methods for one dimensional evolutionary partial differential equations, *J. Comp. Phys.* 74, 191–213 (1988)

9 Bockelie, M. J. Adaptive grid movement schemes and the numerical simulation of shock vortex interaction, *PhD Thesis*, Columbia University (1988)

10 Lee, K. D., Loellback, J. M. and Kim, M. S. Adaptation of structured grids for improved Navier–Stokes solutions, *AIAA Paper 90-0125* (1990)

11 Baines, M. J. The structure of the moving finite element method, *Finite Element Analysis in Fluids*, (Eds. T. J. Chung and G. R. Karr), University of Alabama in Huntsville Press (1989)

12 Benson, R. and McRae, D. A three dimensional dynamic solution adaptive mesh algorithm, *AIAA Paper 90-1566* (1990)
13 Bockelie, M. J. and Eiseman, P. R. A time accurate adaptive grid method and the numerical simulation of a shock vortex interaction, *NASA Langley TP-2998* (1990)
14 Bockelie, M. J., Eiseman, P. R. and Smith, R. E. An adaptive grid method for computing time accurate solutions on structured grids, *AIAAA-91-0144* (1991)
15 Palmeiro, B. and Dervieux, A. On weak and strong coupling between mesh adaptor and flow solvers, *12th Int. Conf. Num. Meth. Fluid Dynamics*, (*Lect. Notes Phys. 371*) (Ed. K. W. Morgan), Springer-Verlag, Berlin (1990)
16 Mastin, C. W. Fast interpolation schemes for moving grids, *Numerical Grid Generation in Computational Fluid Mechanics '88* (Eds. S. Sengupta *et al.*), Pineridge Press, Swansea (1988)
17 Duckowicz, J. K. Accurate conservative remapping (rezoning) for arbitrary lagrangian Eulerian computations, *SIAM J. Sci. Stat. Comput.* **8**, 305–321 (1987)
18 Ramshaw, J. D. Conservative rezoning algorithm for generalized two dimensional meshes, *J. Comp. Phys.* **59**, 193–199 (1985)
19 Williamson, D. L. and Rasch, P. J. Two dimensional semi-Lagrangian transport with shape preserving interpolation, *M Weath Rev.* **117**, 102–129 (1989)
20 Sod, G. A. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws, *J. Comp. Phys.* **27**, 1–31 (1978)
21 Pao, S. P. and Salas, M. D. A numerical study of two dimensional shock vortex interaction, *AIAA Paper 81-1205* (1981)
22 Ribner, H. S. Cylindrical sound wave generated by shock vortex interaction, *AIAA J.* **23**, 1708–1715 (1985)